Unity Pro
I/O Management
Block Library

12/2015

33002531.16

www.schneider-electric.com

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

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

Table of Contents

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

This document describes the functions and function blocks in the I/O Management library.

Validity Note

This document is valid for Unity Pro 11.0 or later.

Related Documents

Title of Documentation	Reference Number
Unity Pro Program Languages and structure, Reference Manual	35006144 (English), 35006145 (French), 35006146 (German), 35013361 (Italian), 35006147 (Spanish), 35013362 (Chinese)
Unity Pro System Bits and Words, Reference Manual	EIO0000002135 (English), EIO0000002136 (French), EIO0000002137 (German), EIO0000002138 (Italian), EIO0000002139 (Spanish), EIO0000002140 (Chinese)

You can download these technical publications and other technical information from our website at http://download.schneider-electric.com

Part I
General

Overview

This section contains general information concerning the I/O Management library.

NOTE: For a detailed description of the system objects (%S and %SW), refer to *Unity Pro System Bits and Words, Reference Manual*.

What Is in This Part?

This part contains the following chapters:

Chapter 1
Block Types and their Applications

Overview

This chapter describes the different block types and their applications.

What Is in This Chapter?

This chapter contains the following topics:

Block Types

Block Types

Block Types

Different block types are used in Unity Pro. The general term for the block types is FFB.

There are the following types of block:

- Elementary Function (EF)
- Elementary Function Block (EFB)
- Derived Function Block (DFB)
- Procedure

NOTE: Motion Function Blocks are not available on the Quantum platform.

Elementary Function

Elementary functions (EF) have no internal status and one output only. If the input values are the same, the output value is the same for the executions of the function, for example the addition of two values gives the same result at every execution.

An elementary function is represented in the graphical languages (FBD and LD) as a block frame with inputs and an output. The inputs are represented on the left and the outputs on the right of the block frame. The name of the function, that is the function type, is shown in the center of the block frame.

The number of inputs can be increased with some elementary functions.

⚠ CAUTION
UNEXPECTED BEHAVIOR OF EQUIPMENT For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF. **Failure to follow these instructions can result in injury or equipment damage.**

NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools → Program → Languages → Common**.

Elementary Function Block

Elementary function blocks (EFB) have an internal status. If the inputs have the same values, the value on the outputs can have another value during the individual executions. For example, with a counter, the value on the output is incremented.

An elementary function block is represented in the graphical languages (FBD and LD) as a block frame with inputs and outputs. The inputs are represented on the left and the outputs on the right of the block frame. The name of the function block, that is the function block type, is shown in the center of the block frame. The instance name is displayed above the block frame.

Derived Function Block

Derived function blocks (DFBs) have the same properties as elementary function blocks. They are created by the user in the programming languages FBD, LD, IL and/or ST.

Procedure

Procedures are functions with several outputs. They have no internal state.

The only difference from elementary functions is that procedures can have more than one output and they support variables of the `VAR_IN_OUT` data type.

Procedures do not return a value.

Procedures are a supplement to IEC 61131-3 and must be enabled explicitly.

There is no visual difference between procedures and elementary functions.

⚠ CAUTION

UNEXPECTED BEHAVIOR OF EQUIPMENT

For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF.

Failure to follow these instructions can result in injury or equipment damage.

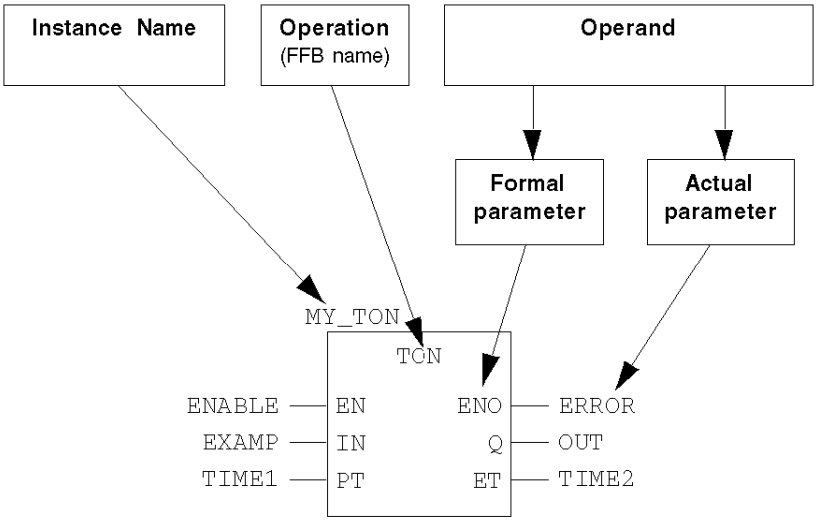
NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools → Program → Languages → Common**.

FFB Structure

Structure

Each FFB is made up of an operation (name of the FFB), the operands are required for the operation (formal and actual parameters) and an instance name for elementary/derived function blocks.

Call of a function block in the FBD programming language:



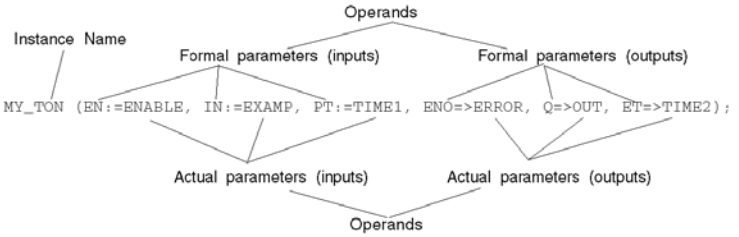
⚠ CAUTION

UNEXPECTED APPLICATION BEHAVIOR

Do not call several times the same block instance within a PLC cycle

Failure to follow these instructions can result in injury or equipment damage.

Formal call of a function block in the ST programming language:



Operation

The operation determines which function is to be executed with the FFB, e.g. shift register, conversion operations.

Operand

The operand specifies what the operation is to be executed with. With FFBs, this consists of formal and actual parameters.

Formal/actual parameters

Inputs and outputs are required to transfer values to or from an FFB. These are called formal parameters.

Objects are linked to formal parameters; these objects contain the current process states. They are called actual parameters.

At program runtime, the values from the process are transferred to the FFB via the actual parameters and then output again after processing.

The data type of the actual parameters must match the data type of the input/output (formal parameters). The only exceptions are generic inputs/outputs whose data type is determined by the actual parameter. If the actual parameters consist of literals, a suitable data type is selected for the function block.

FFB Call in IL/ST

In text languages IL and ST, FFBs can be called in formal and in informal form. For detail, refer to chapter *Programming Language (see Unity Pro, Program Languages and Structure, Reference Manual)*.

Example of a formal function call:

```
out:=LIMIT (MN:=0, IN:=var1, MX:=5);
```

Example of an informal function call:

```
out:=LIMIT (0, var1, 5);
```

NOTE: The use of `EN` and `ENO` is only possible for formal calls.

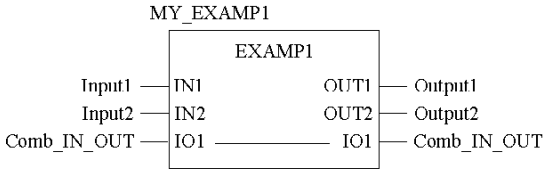
VAR_IN_OUT variable

FFBs are often used to read a variable at an input (input variables), to process it and to output the altered values of the **same** variable (output variables).

This special type of input/output variable is also called a VAR_IN_OUT variable.

The input and output variable are linked in the graphic languages (FBD and LD) using a line showing that they belong together.

Function block with VAR_IN_OUT variable in FBD:



Function block with VAR_IN_OUT variable in ST:

```
MY_EXAMP1 (IN1:=Input1, IN2:=Input2, IO1:=Comb_IN_OUT,
           OUT1=>Output1, OUT2=>Output2);
```

The following points must be considered when using FFBs with VAR_IN_OUT variables:

- The VAR_IN_OUT inputs must be assigned a variable.
- Literals or constants cannot be assigned to VAR_IN_OUT inputs/outputs.

The following additional limitations apply to the graphic languages (FBD and LD):

- When using graphic connections, VAR_IN_OUT outputs can only be connected with VAR_IN_OUT inputs.
- Only one graphical link can be connected to a VAR_IN_OUT input/output.
- Different variables/variable components can be connected to the VAR_IN_OUT input and the VAR_IN_OUT output. In this case the value of the variables/variable component on the input is copied to the output variables/variable component.
- No negations can be used on VAR_IN_OUT inputs/outputs.
- A combination of variable/address and graphic connections is not possible for VAR_IN_OUT outputs.

EN and ENO

Description

An EN input and an ENO output can be configured for the FFBs.

If the value of EN is equal to "0" when the FFB is invoked, the algorithms defined by the FFB are not executed and ENO is set to "0".

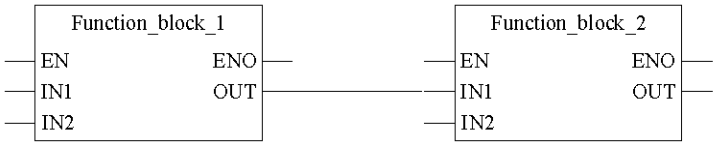
If the value of EN is equal to "1" when the FFB is invoked, the algorithms defined by the FFB will be executed. After the algorithms have been executed successfully, the value of ENO is set to "1". If certain error conditions are detected when executing these algorithms, ENO is set to "0".

If the EN pin is not assigned a value, when the FFB is invoked, the algorithm defined by the FFB is executed (same as if EN equals to "1"), Please refer to *Maintain output links on disabled EF (see Unity Pro, Operating Modes)*.

If the algorithms are executed successfully, then value of ENO is set to "1", else ENO is set to "0".

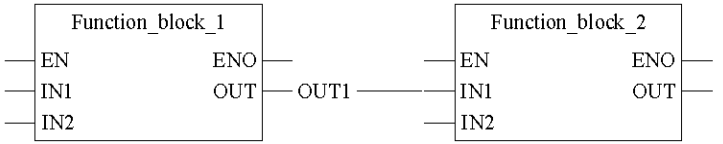
If ENO is set to "0" (caused by EN=0 or a detected error condition during execution or unsuccessful algorithm execution):

- Function blocks
  - EN/ENO handling with function blocks that (only) have one link as an output parameter:



If EN from FunctionBlock_1 is set to "0", the output connection OUT from Function-Block_1 retains the status it had in the last correctly executed cycle.

  - EN/ENO handling with function blocks that have one variable and one link as output parameters:



If EN from FunctionBlock_1 is set to "0", the output connection OUT from Function-Block_1 retains the status it had in the last correctly executed cycle. The variable OUT1 on the same pin, either retains its previous status or can be changed externally without influencing the connection. The variable and the link are saved independently of each other.

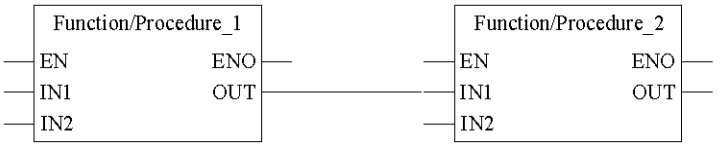
- Functions/Procedures

⚠ CAUTION
UNEXPECTED BEHAVIOR OF EQUIPMENT For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF. **Failure to follow these instructions can result in injury or equipment damage.**

NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools → Program → Languages → Common**.

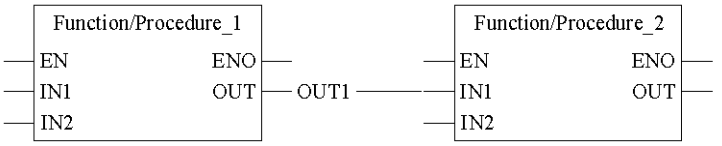
As defined in IEC61131-3, the outputs from deactivated functions (EN-input set to "0") is undefined. (The same applies to procedures.)
Here is an explanation of the output status in this case:

- EN/ENO handling with functions/procedures that (only) have one link as an output parameter:



If EN from Function/Procedure_1 is set to "0", the output connection OUT from Function/Procedure_1 is also set to "0".

- EN/ENO handling with function blocks that have one variable and one link as output parameters:



If EN from Function/Procedure_1 is set to "0", the output connection OUT from Function/Procedure_1 is also set to "0". The variable OUT1 on the same pin, either retains its previous status or can be changed externally without influencing the connection. The variable and the link are saved independently of each other.

The output behavior of the FFBs does not depend on whether the FFBs are called up without EN/ENO or with EN=1.

Conditional/Unconditional FFB Call

"Unconditional" or "conditional" calls are possible with each FFB. The condition is realized by pre-linking the input EN.

- EN connected
  conditional calls (the FFB is only processed if EN = 1)
- EN shown, hidden, and marked TRUE, or shown and not occupied
  unconditional calls (FFB is processed independent from EN)

NOTE: For disabled function blocks (EN = 0) with an internal time function (e.g. DELAY), time seems to keep running, since it is calculated with the help of a system clock and is therefore independent of the program cycle and the release of the block.

⚠ CAUTION
UNEXPECTED APPLICATION EQUIPMENT
Do not disable function blocks with internal time function during their operation.
Failure to follow these instructions can result in injury or equipment damage.

Note for IL and ST

The use of EN and ENO is only possible in the text languages for a formal FFB call, e.g.

```
MY_BLOCK (EN:=enable, IN1:=var1, IN2:=var2,
ENO=>error, OUT1=>result1, OUT2=>result2);
```

Assigning the variables to ENO must be done with the operator =>.

With an informal call, EN and ENO cannot be used.

Chapter 2
Block Availability on the Various Hardware Platforms

Block Availability on the Various Hardware Platforms

Introduction

Not all blocks are available on all hardware platforms. The blocks available on your hardware platform can be found in the following tables.

NOTE: The functions and function blocks in this library are not defined in IEC61131-3.

Analog I/O Configuration

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
I_FILTER	EF	-	-	+	-	-
I_SET	EFB	-	+	+	-	-
O_FILTER	EF	-	-	+	-	-
O_SET	EFB	-	+	+	-	-
+ Yes – No						

Analog I/O Scaling

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
I_NORM	EF	-	+	+	-	-
I_NORM_WARN	EFB	-	+	+	-	-
I_PHYS	EF	-	+	+	-	-
I_PHYS_WARN	EFB	-	+	+	-	-
I_RAW	EF	-	+	+	-	-
I_RAWSIM	Procedure	-	+	+	-	-
I_SCALE	EFB	-	+	+	-	-
I_SCALE_WARN	EFB	-	+	+	-	-
O_NORM	Procedure	-	+	+	-	-
+ Yes – No						

Block name	Block type	M340	M580	Quantum	Momentum	Premium
O_NORM_WARN	EFB	-	+	+	-	-
O_PHYS	Procedure	-	+	+	-	-
O_PHYS_WARN	EFB	-	+	+	-	-
O_RAW	Procedure	-	+	+	-	-
O_SCALE	Procedure	-	+	+	-	-
O_SCALE_WARN	EFB	-	+	+	-	-

+ Yes
– No

Explicit exchange

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
READ_PARAM	Procedure	+	+(2)	-	-	+
READ_PARAM_MX	EFB	-	+(3)	-	-	-
READ_STS	Procedure	+	+(2)	-	-	+
READ_STS_QX	EFB	-	-	+(1)	-	-
READ_STS_MX	EFB	-	+(4)	-	-	-
READ_TOPO_ADDR	EF	+	+(2)	-	-	+
RESTORE_PARAM	Procedure	+	+(2)	-	-	+
RESTORE_PARAM_MX	EFB	-	+(3)	-	-	-
SAVE_PARAM	Procedure	+	+(2)	-	-	+
SAVE_PARAM_MX	EFB	-	+(3)	-	-	+
TRF_RECIPE	Procedure	-	-	-	-	+
WRITE_CMD	Procedure	+	+(2)	-	-	+
WRITE_CMD_QX	EFB	-	-	+(1)	-	-
WRITE_CMD_MX	EFB	-	+(4)	-	-	-
WRITE_PARAM	Procedure	+	+(2)	-	-	+

+ Yes
– No
(1) M340 Ethernet RIO drop addressed from a Quantum platform
(2) Local rack with topological addressing (IODDT) vision.
(3) Local rack with Device DDT vision.
(4) Local rack or Ethernet RIO drop addressed from an M580 platform with Device DDT vision.

Block name	Block type	M340	M580	Quantum	Momentum	Premium
WRITE_PARAM_MX	EFB	-	+(3)	-	-	-

+ Yes
– No
(1) M340 Ethernet RIO drop addressed from a Quantum platform
(2) Local rack with topological addressing (IODDT) vision.
(3) Local rack with Device DDT vision.
(4) Local rack or Ethernet RIO drop addressed from an M580 platform with Device DDT vision.

Immediate I/O

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
IMIO_IN	EFB	-	-	+	-	-
IMIO_OUT	EFB	-	-	+	-	-
IU_ERIO	EFB	-	+	+	-	-

+ Yes
– No

Quantum I/O Configuration

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
ACI030	EFB	-	+	+	-	-
ACI040	EFB	-	+	+	-	-
ACO020	EFB	-	+	+	-	-
ACO130	EFB	-	+	+	-	-
AII330	EFB	-	+	+	-	-
AII33010	EFB	-	+	+	-	-
AIO330	EFB	-	+	+	-	-
AMM090	EFB	-	+	+	-	-
ARI030	EFB	-	+	+	-	-
ATI030	EFB	-	+	+	-	-
AVI030	EFB	-	+	+	-	-
AVO020	EFB	-	+	+	-	-
DROP	EFB	-	+	+	-	-

+ Yes
– No
(1) This EFB is reserved for internal use. Please contact your local sales agency and/or support for more information.

Block name	Block type	M340	M580	Quantum	Momentum	Premium
ERT_854_10	EFB	-	+	+	-	-
ERT_854_20	EFB	-	+	+	-	-
ERT_854_30	EFB	-	+(1)	+(1)	-	-
QUANTUM	EFB	-	-	+	-	-
XBE	EFB	-	-	+	-	-
XDROP	EFB	-	+	+	-	-

+ Yes
– No
(1) This EFB is reserved for internal use. Please contact your local sales agency and/or support for more information.

Simulation

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
WRITE_INPUT_DINT	EB	+	+	+	+	+
WRITE_INPUT_EBOOL	EB	+	+	+	+	+
WRITE_INPUT_EBOOL_16	Procedure	+	+	+	+	+
WRITE_INPUT_INT	EB	+	+	+	+	+
WRITE_INPUT_REAL	EB	+	+	+	+	+
WRITE_INPUT_UINT	EB	+	+	+	+	+
WRITE_INPUT_UDINT	EB	+	+	+	+	+

+ Yes
– No

Chapter 3
Operating Analog Modules

Introduction

This block library contains EFBs for the operation of analog modules.

NOTE: This block library's EFBs are available in every application. It is not possible to use these platform specific EFBs on a PLC platform for which they were not intended.

The EFBs for the analog modules can be found in the following families:

- Quantum I/O configuration
- Analog I/O configuration
- Analog I/O scaling

What Is in This Chapter?

This chapter contains the following topics:

Editing Analog Values

Introduction

This block library contains EFBs for the operation of analog modules. The EFBs are designed in such a way that it enables the FBD program configuration to be largely independent of the hardware module used. Hardware-dependent EFBs (e.g. family: `Quantum IO Configuration`) are used to evaluate project-specific information on the PLC and to process it in the hardware independent data structures `ANL_IN` and `ANL_OUT`. Hardware independent EFBs work with these data structures (e.g. family: `Analog I/O Scaling`), that read, scale and convert raw values from the input words (%IWx) into `REAL` values. This means that changes in direct addresses or changes in input/output parameters can be detected automatically by the EFBs.

Division into sections

As the detection of configuration data only occurs once after loading, it is advisable to divide the EFBs in the ANA_I/O library into at least two sections.

Division into at least two sections is recommended.

- Scaling section
- Configuration section

This division into a configuration section and several scaling sections can lead to a reduction of the CPU load, as the configuration part (configuration section) must only execute once, (after a cold restart or a warm restart). The scaling sections are usually executed continuously.

Scaling and Configuration Sections

Scaling section

Scaling sections are used for actual analog value processing.

Configuration section

The configuration section is used to configure the analog I/O modules and controls the data exchange between analog EFBs, the State RAM and the configuration data.

The configuration section should be called `CfgAnaIo` to ensure compatibility with future Unity Pro versions.

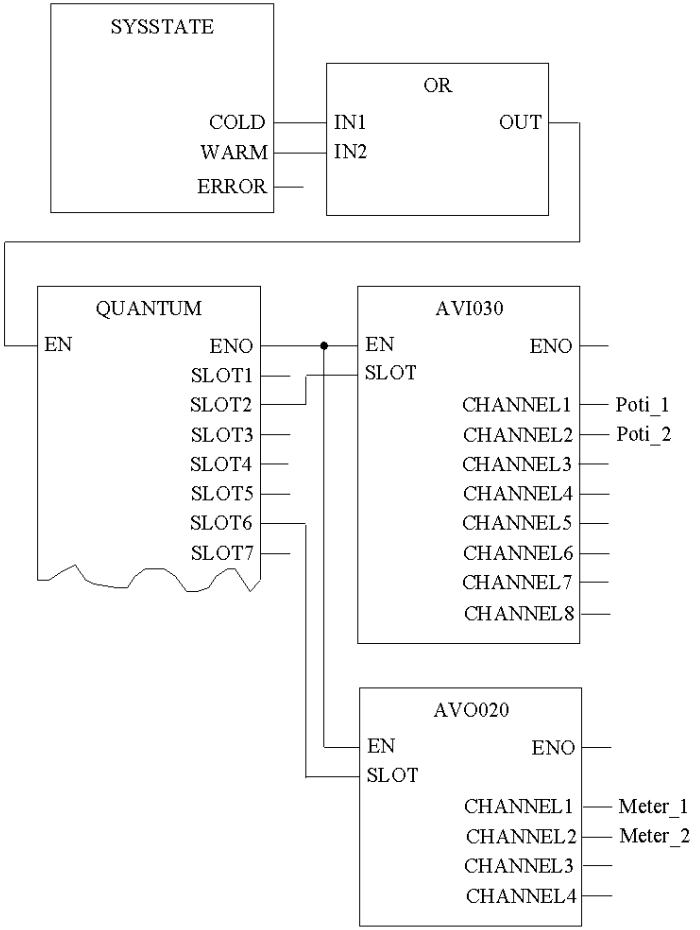
Two options are available to control a configuration section:

- using the `EN` inputs of the individual EFBs
- by enabling or disabling the configuration section

Controlling the Configuration Section

Control of the configuration section is possible through the `EN` inputs of this section's individual EFBs. The EFBs are enabled through the `SYSSTATE` EFB which has `COLD` or `WARM` outputs that are set to 1 for one cycle after either a cold or a warmstart.

Example of a configuration section `CfgAnaIo`



Configuration EFBs

Introduction

Configuration data from analog input/output modules is accessed using EFBs from the `Quantum I/O Configuration` block library family.

Procedure

Then place a single `QUANTUM` EFB in the configuration section `CfgAnaIo`.

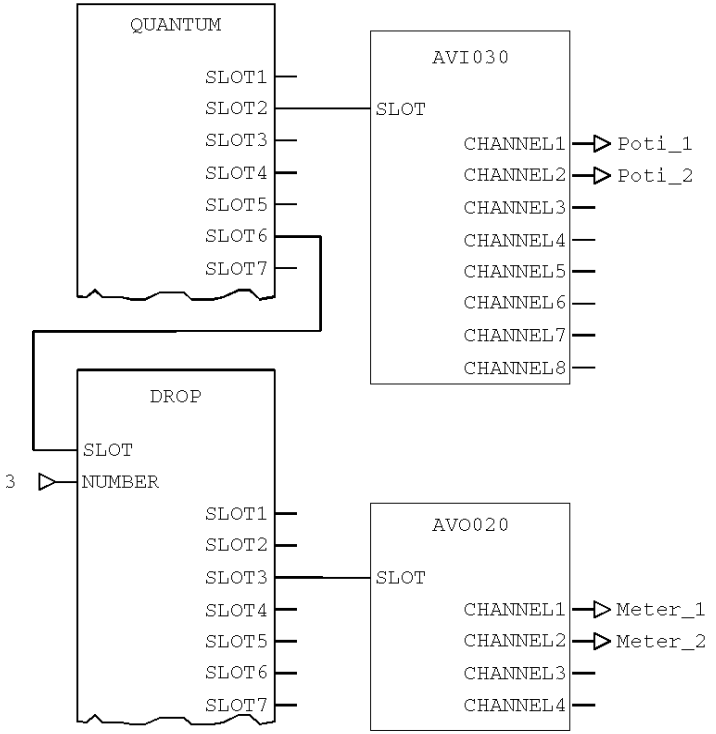
The configuration data from remote I/O (RIO), distributed I/O (DIO) or network option I/O (NOM) is accessed via `DROP` EFBs. The `DROP` EFB is applicable to all three I/O station types. If RIO or NOM is used, connect the `DROP` EFB to the slot on the RIO communication module or the NOM module. If DIO is used, connect the `DROP` EFB to the slot on the CPU. Each I/O station has its own number. Specify this number at the `NUMBER` input of the `DROP` EFB.

Each analog module has its own analog I/O EFB. Connect the desired analog I/O EFB to the corresponding slot on the `QUANTUM` or `DROP` EFB. The analog I/O EFB provides a variable of data type `ANL_IN` or `ANL_OUT`as an output. These values can be further processed using the scaling EFBs in the scaling sections. To do this, they are connected with the relevant scaling EFBs using Unlocated variables.

NOTE: Do not specify Literals at the `SLOT` inputs of the configuration EFBs. `SLOT` inputs must be connected to `SLOT` outputs.

Example of a configuration section

Example of a configuration section `CfgAnaIo`



The EFB's mode of functioning can be found in the table below.

EFB	Function mode
`QUANTUM`	The EFB is used to edit the configuration data of a primary rack for transfer by the analog in/out EFBs.
`DROP`	The EFB is used to edit the configuration data of an I/O rack for transfer by the analog in/out EFBs.
`AVI030`	Quantum module AVI 030 00 configuration. The EFB is used to edit the configuration data of the Quantum module AVI 030 00 for continued processing by the scaling EFBs. The %IW references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

EFB	Function mode
AVO020	Quantum module AVO 020 00 configuration. The EFB is used to edit the configuration data of the Quantum module AVO 020 00 for continued processing by the scaling EFBs. The %MW references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

Procedure for expanding the local module rack using the XBE module (Quantum)

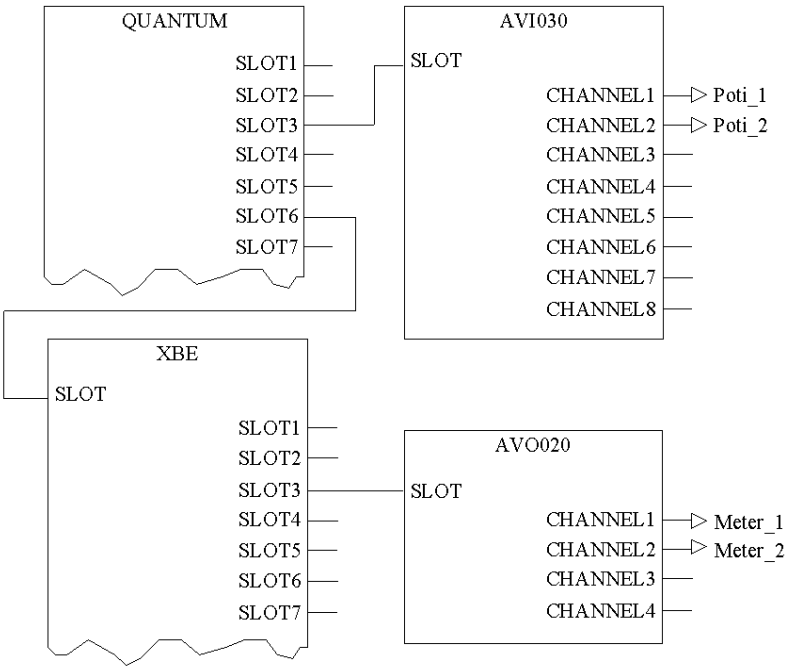
Place a single QUANTUM EFB in the configuration section (CfgAnaIo).

The module rack expander configuration data is accessed via the XBE EFB. Connect the XBE EFB to the slot on the XBE module.

NOTE: Do not specify Literals at the SLOT inputs of the configuration EFBs. SLOT inputs must be connected to SLOT outputs.

Procedure for expanding the local module rack using the XBE module

Example of a configuration section CfgAnaIo



The EFB's mode of functioning can be found in the table below.

EFB	Function mode
QUANTUM	The EFB is used to edit the configuration data of a primary rack for transfer by the analog in/out EFBs.
XBE	The EFB is used to edit the configuration data of a module rack expander for transfer by the analog in/out EFBs.
AVI030	See *Example of a configuration section, page 34*
AVO020	See *Example of a configuration section, page 34*

Procedure for Remote I/O (RIO, DIO)

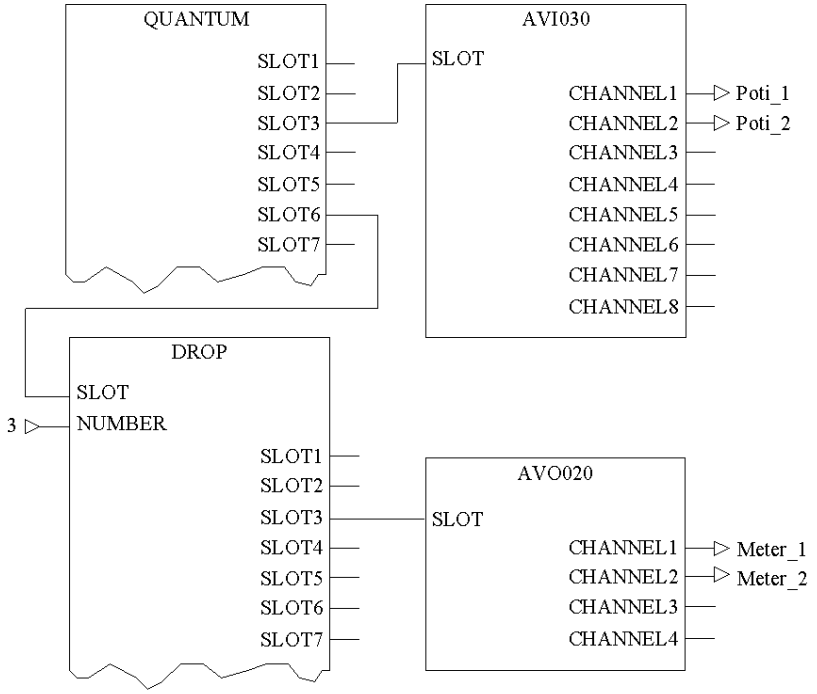
Place a single QUANTUM EFB in the configuration section (CfgAnaIo).

The configuration data from remote I/O (RIO) and distributed I/O (DIO) is accessed via DROP EFB. The DROP EFB is applicable to all three I/O station types. If RIO is used, connect the DROP EFB to the slot on the RIO communication module. If DIO is used, connect the DROP EFB to the slot on the CPU or NOM module. Each I/O station has its own address. Specify this number at the NUMBER input of the DROP EFB.

NOTE: Do not specify Literals at the SLOT inputs of the configuration EFBs. SLOT inputs must be connected to SLOT outputs.

Procedure for Remote I/O (RIO or NOM)

Example of a configuration section `CfgAnaIo`



The EFB's mode of functioning can be found in the table below

EFB	Function mode
`QUANTUM`	The EFB is used to edit the configuration data of a primary rack for transfer by the analog in/out EFBs.
`DROP`	The EFB is used to edit the configuration data of an I/O rack for transfer by the analog in/out EFBs.
`AVI030`	See *Example of a configuration section, page 34*
`AVO020`	See *Example of a configuration section, page 34*

Procedure for expanding the remote module rack using the XBE module

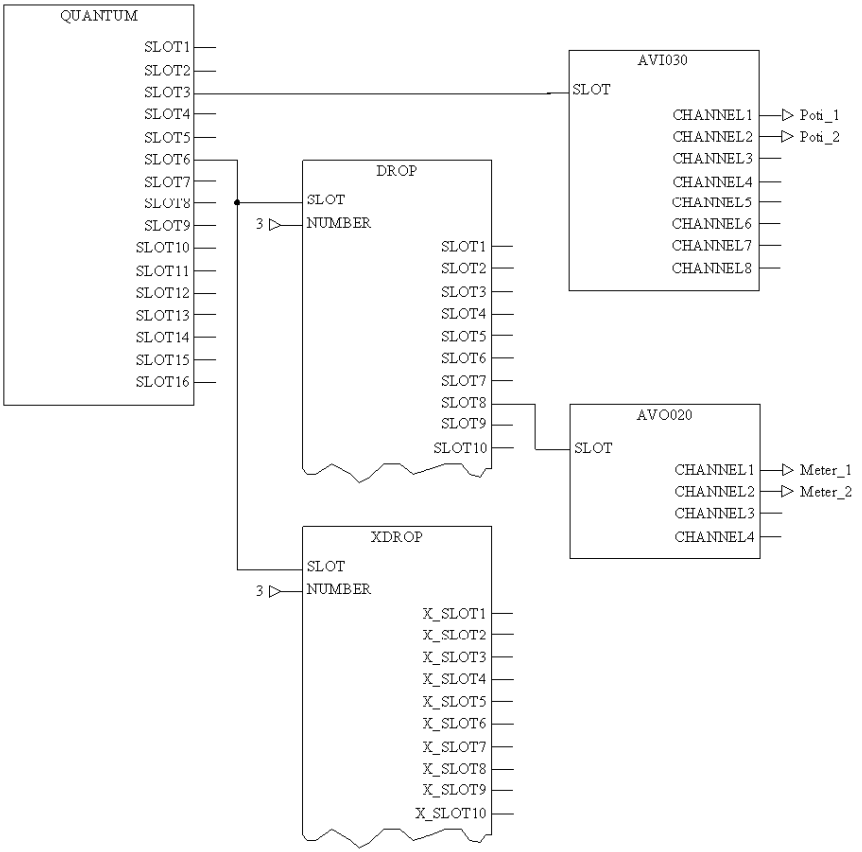
Place a single QUANTUM EFB in the configuration section (CfgAnaIo).

The remote module rack expander configuration data is accessed via the XDROP *(see page 291)* EFB. Connect the SLOT input of the XDROP EFB with the SLOT input of the DROP EFB. At the NUMBER input of the XDROP EFBs, enter the **same** number as at the NUMBER input of the DROP EFB.

NOTE: Do not specify Literals at the SLOT inputs of the configuration EFBs. SLOT inputs must be connected to SLOT outputs.

Procedure for expanding the remote module rack using the XBE module

Example of a configuration section CfgAnaIo



The EFB's mode of functioning can be found in the table below.

EFB	Function mode
QUANTUM	The EFB is used to edit the configuration data of a primary rack for transfer by the analog in/out EFBs.
XDROP	The EFB is used to edit the configuration data of a remote module rack expansion for transfer by the analog in/out EFBs.
AVI030	See *Example of a configuration section, page 34*
AVO020	See *Example of a configuration section, page 34*

Scaling EFBs

Introduction

The analog values are scaled using the EFBs of the `Analog IO Configuration` block library family in the scaling sections.

The analog I/O EFBs operate hardware independently with the data types `ANL_IN` and `ANL_OUT`.

Scaling EFBs available

The following scaling EFBs are available:

- `I_RAW`, `I_RAWSIM`, `O_RAW`:
  Raw value, no scaling
- `I_NORM`, `I_NORM_WARN`, `O_NORM`, `O_NORM_WARN`:
  Normalization, representation in a range from 0.0 to 1.0
- `I_PHYS`, `I_PHYS_WARN`, `O_PHYS`, `O_PHYS_WARN`:
  Physical, physical range
- `I_SCALE`, `I_SCALE_WARN`, `O_SCALE`, `O_SCALE_WARN`:
  Scaled, representation in a user-defined range from `MN` to `MX`

Handy tips

Please take note of the following tips about using scaling EFBs:

- When using these EFBs, the messages in **Tools** → **Diagnostic viewer** should definitely be observed. That is where execution errors for these EFBs are recorded.
- For tasks not requiring the physical units and/or only scaling by 0-100% the `NORM` EFB are preferred to the `PHYS` or `SCALE` EFBs.
- If scaling is required, the `PHYS` EFB should be used. `PHYS` EFBs do not work without information about the physical units however, for these the `SCALE` EFBs are used.
- `SCALE` EFBs can also be used if physical scaling is not required or is not possible.
- `SCALE` EFBs do not work in conjunction with input/output modules which provide direct physical values (e.g. decimal values). This applies, for example, to temperature or resistance modules not set to raw values (see parameter dialog box in the IO map).
- EFBs with `WARN` cannot be used for all input/output modules. The descriptions of the individual EFBs explains which modules they can be used with.
- "Open circuit" is categorized as an error rather than a warning. "Open circuit" generates an online error message which can be accessed using **Tools** → **Diagnostics viewer** and sets the output `ENO` of the `WARN` EFBs to "0".
- The `RAW` EFB is not usually required. It merely represents a simple way to make additional use of the raw values.

Application example for Quantum

Introduction

To precisely monitor the output values, it is advisable to implement the scaling with two EFBs. The first EFB (scaling EFBs) scales the analog value and the second EFB monitors the scaled value for ranges preset by the process. In the following process, either the original Y output of the scaling EFB or the limited OUT output of the Limiter EFB can be used.

Application example

A simple example shows how the EFBs can be used.

The example assumes a boiler with a capacity of 350 liters. The input voltage ranges from 0.0 Volt for 0 liters to 10.0 Volt for 1000 liters. A PI controller should guarantee a volume between 200 and 300 liters. The Limiter EFB detects violations in this range and will limit the output.

Given values:

BoilerMn: 0

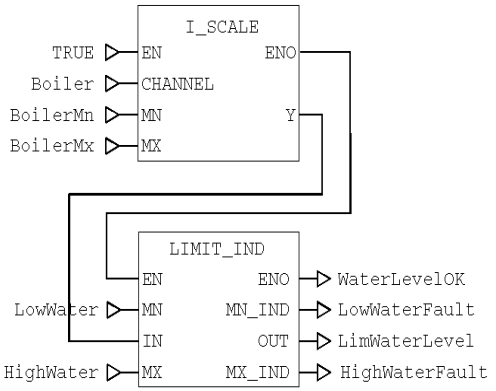
BoilerMx: 1 000

LowWater: 199

HighWater: 301

Boiler is an unlocated variable of the ANL_IN type and is linked to an AVI030 EFB.

Application example



Part II
Analog I/O Configuration

Overview

This section describes the elementary functions and elementary function blocks from the family `Analog I/O configuration`.

What Is in This Part?

This part contains the following chapters:

Chapter 4
I_FILTER: Linearization for analog-inputs

Introduction

This chapter describes the `I_FILTER` block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

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

3 different adjustments are available:

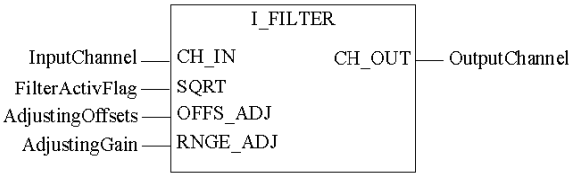
- Linearization with square root (standardized range)
- Correction of the "Offset" (zero offset compensation)
- Correction of "Range" (gain)

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

`EN` and `ENO` can be configured as additional parameters.

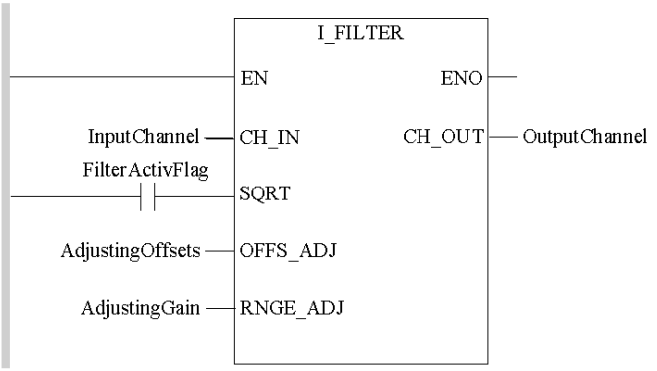
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputChannel
I_FILTER FilterActivFlag,AdjustingOffsets,AdjustingGain
ST OutputChannel
```

Representation in ST

Representation:

```
OutputChannel := I_FILTER (InputChannel, FilterActivFlag,
 AdjustingOffsets, AdjustingGain) ;
```

Parameter description

Description of input parameters:

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

Description of output parameters:

Parameter	Data type	Meaning
CH_OUT	ANL_IN	Output channel

Runtime error

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

NOTE: For a list of all block error codes and values, see*Analog I/O Configuration, page 314*.

Detailed description

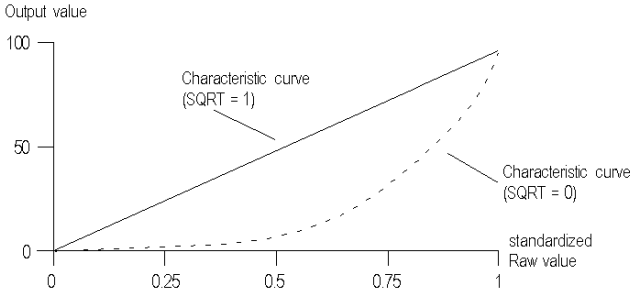
Linearization with square root (standardized range)

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 parameter can be used to modify (adjust) the calculated offset value of the output.

Correction of the automatically set value (OFFS_ADJ = 0) is not normally necessary.

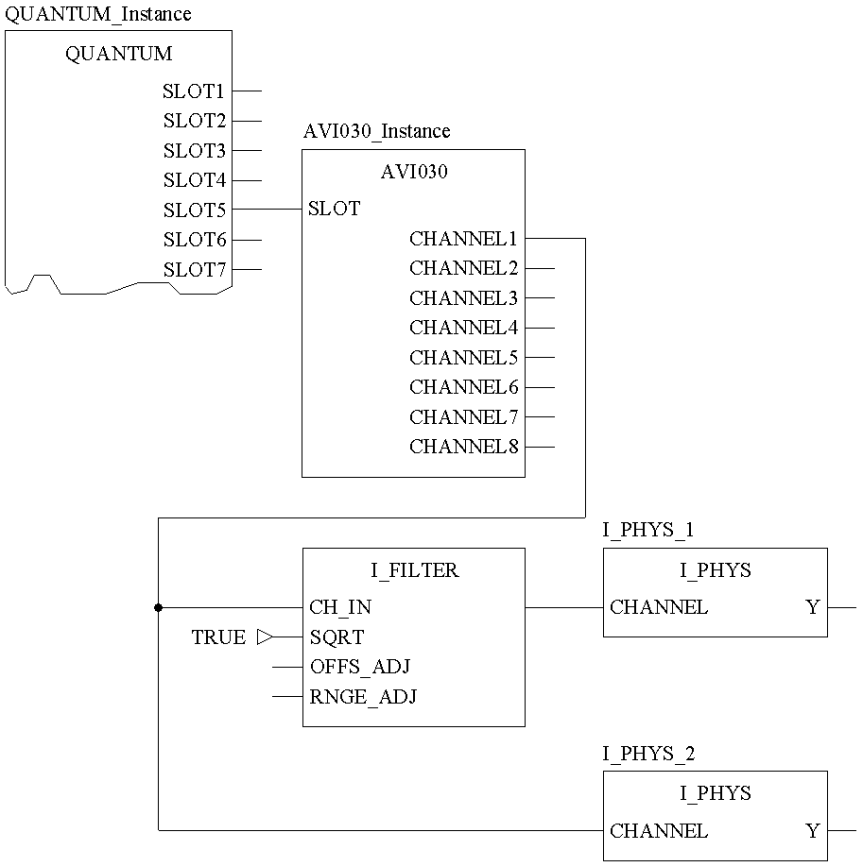
Correction of "Range" (gain)

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

Correction of the automatically set value (RNGE_ADJ = 0) is not normally necessary.

Example

Structure with I_FILTER



The inputs OFFS_ADJ and RNGE_ADJ of the I_FILTER function block are not used. They are set to "0" by default.

The following values apply for function block I_PHYS (I_PHYS_1):

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

Input values (AVI030 10 V)	Output values (I_PHYS)
10 V	10.0

The following values apply for I_PHYS (I_PHYS_2) function block:

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

Chapter 5
I_SET: Set information from analog input channels

Introduction

This chapter describes the block `I_SET`.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The function block sets the information for the analog input channels (ANL_IN).

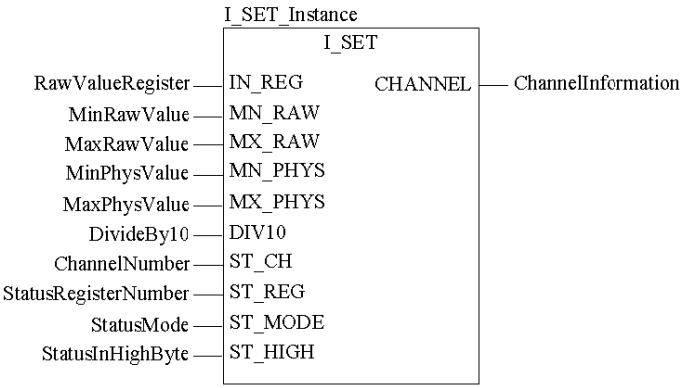
This block enables all scaling blocks of this library to be used.

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.

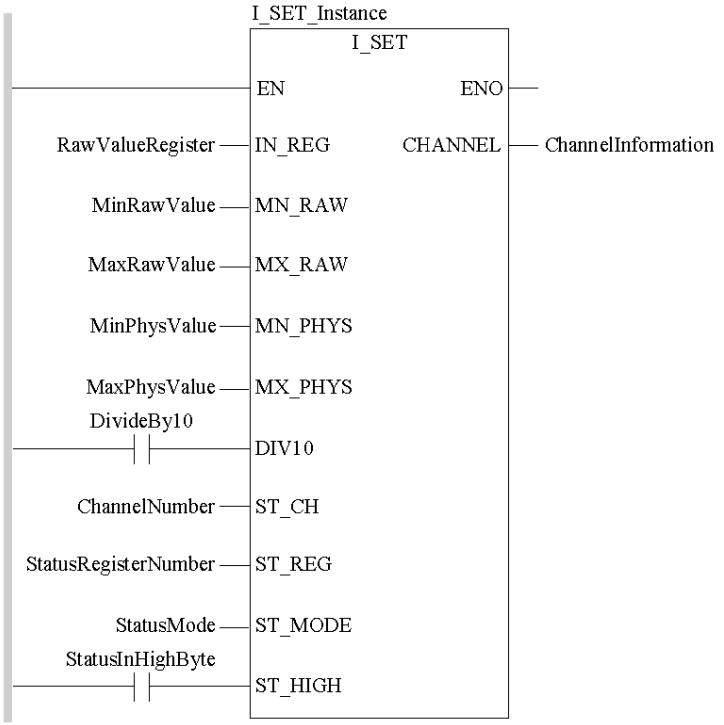
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL I_SET_Instance (IN_REG:=RawValueRegister,
    MN_RAW:=MinRawValue, MX_RAW:=MaxRawValue,
    MN_PHYS:=MinPhysValue, MX_PHYS:=MaxPhysValue,
    DIV10:=DivideBy10, ST_CH:=ChannelNumber,
    ST_REG:=StatusRegisterNumber, ST_MODE:=StatusMode,
    ST_HIGH:=StatusInHighByte, CHANNEL=>ChannelInformation)
```

Representation in ST

Representation:

```
I_SET_Instance (IN_REG:=RawValueRegister,
    MN_RAW:=MinRawValue, MX_RAW:=MaxRawValue,
    MN_PHYS:=MinPhysValue, MX_PHYS:=MaxPhysValue,
    DIV10:=DivideBy10, ST_CH:=ChannelNumber,
    ST_REG:=StatusRegisterNumber, ST_MODE:=StatusMode,
    ST_HIGH:=StatusInHighByte,
    CHANNEL=>ChannelInformation) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
IN_REG	UINT	Number of the raw value register (%IW). For example: 1 for %IW1
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) No constants are allowed
MX_PHYS	INT	greatest input value (e.g. +10 V as 10) No constants are allowed
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 (%IW) For example: 9 for %IW9
ST_MODE	UINT	Status mode (e.g. 1=AVI_STATUS_MODE)
ST_HIGH	BOOL	Status byte found in high byte of the register

Description of the output parameters:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	channel information to be described

NOTE: If the EFB is needed, the data type can be change in the data editor as ANL_IN.

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 (%IW).
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 (%IW).
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

Detailed description

Area of application

The function block can be used in three areas:

1. Raw value scaling, with the blocks: 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 (%IW).

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 used to avoid floating point values in the range 0.2 V ... 1 V. In this range, the settings MN_PHYS=2, MX_PHYS=10 and DIV10=1 are to be made.

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, page 58*.

Evaluation of error information

For the evaluation of error information the inputs ST_CH, ST_REG, 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 (%IW), 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	Not supported
5	DAU204_STATUS_MODE	Not supported
6	ADU205_STATUS_MODE	Not supported
7	AMM090_STATUS_MODE	AMM090
8	ADU214_STATUS_MODE	Not supported

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

Chapter 6
O_FILTER: Linearization for analog outputs

Introduction

This chapter describes the O_FILTER block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

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

3 different adjustments are available:

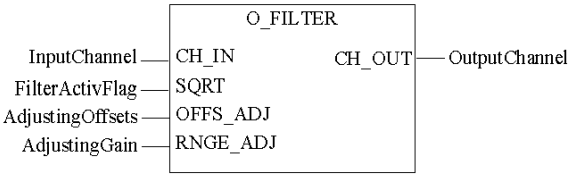
- Linearization with square root (standardized range)
- Correction of the "Offset" (zero offset compensation)
- Correction of "Range" (gain)

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

`EN` and `ENO` can be configured as additional parameters.

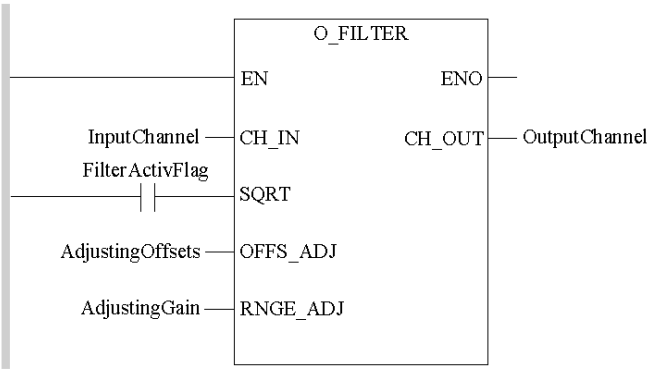
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputChannel
O_FILTER FilterActivFlag, AdjustingOffsets, AdjustingGain
ST OutputChannel
```

Representation in ST

Representation:

```
OutputChannel := O_FILTER (InputChannel, FilterActivFlag,
    AdjustingOffsets, AdjustingGain) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
CH_IN	ANL_OUT	Input channel
SQRT	BOOL	Square root filter 1: Filter active 0: Filter inactive
OFFS_ADJ	INT	Adjusting offset
RNGE_ADJ	INT	Adjusting gain

Description of output parameters:

Parameter	Data type	Meaning
CH_OUT	ANL_OUT	Output channel

Runtime error

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

NOTE: For a list of all block error codes and values, see*Analog I/O Configuration, page 314*.

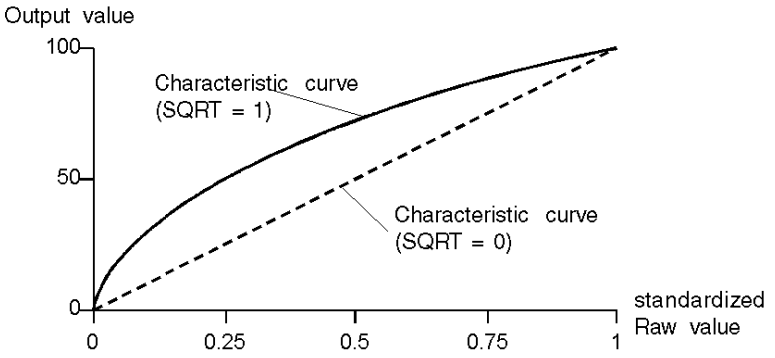
Detailed description

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)

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

Correction of the automatically set value (OFFS_ADJ = 0) is not normally necessary.

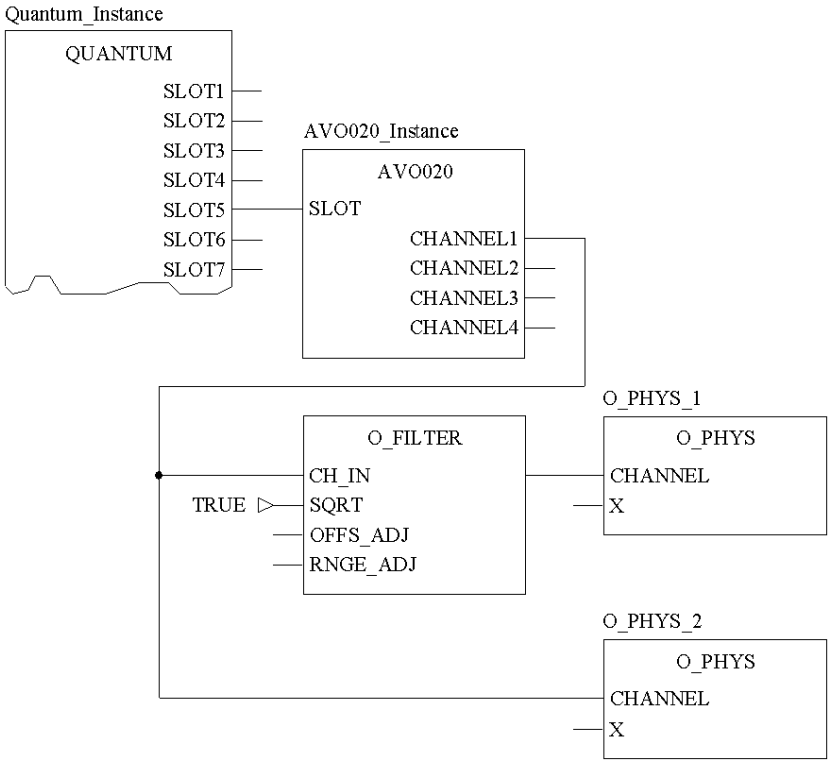
Correction of "Range" (gain)

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

Correction of the automatically set value (RNGE_ADJ = 0) is not normally necessary.

Example

Structure with O_FILTER



The inputs `OFFS_ADJ` and `RNGE_ADJ` of the `O_FILTER` function are not used. They are set to "0"by default.

The following values apply for `O_PHYS` (O_PHYS_1) function block:

Input values (AVO020 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 0_PHYS (O_PHYS_2) function block:

Input values (AVO020 10 V)	**Output values (0_PHYS)**
0 V	0.0
2.5 V	2.5
5 V	5.0
10 V	10.0

Chapter 7
O_SET: Set information from analog output channels

Introduction

This chapter describes the block O_SET.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The function block sets the information for the analog output channels (ANL_OUT).

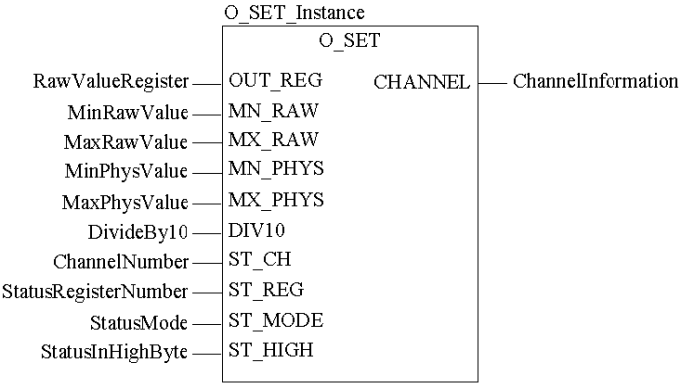
This block enables all scaling blocks of this library to be used.

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.

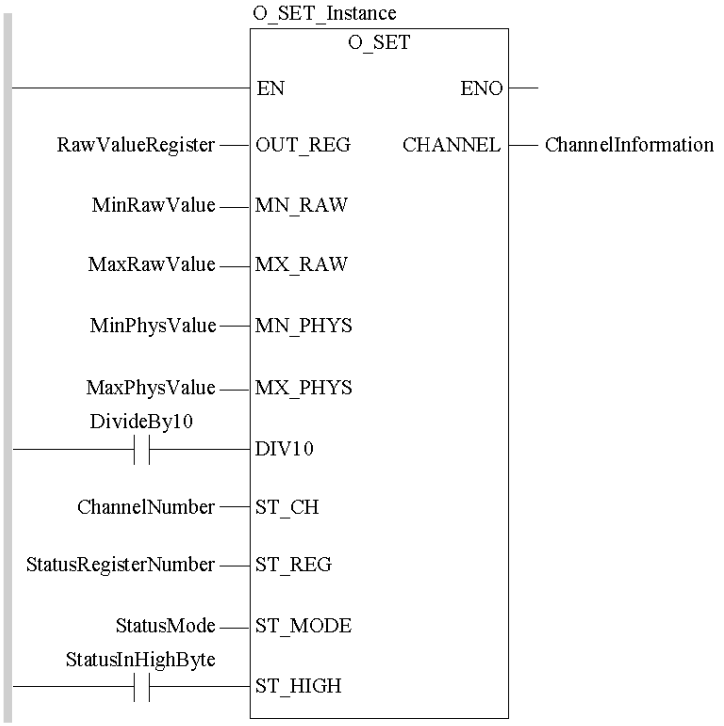
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL O_SET_Instance (OUT_REG:=RawValueRegister,
    MN_RAW:=MinRawValue, MX_RAW:=MaxRawValue,
    MN_PHYS:=MinPhysValue, MX_PHYS:=MaxPhysValue,
    DIV10:=DivideBy10, ST_CH:=ChannelNumber,
    ST_REG:=StatusRegisterNumber, ST_MODE:=StatusMode,
    ST_HIGH:=StatusInHighByte, CHANNEL=>ChannelInformation)
```

Representation in ST

Representation:

```
O_SET_Instance (OUT_REG:=RawValueRegister,
    MN_RAW:=MinRawValue, MX_RAW:=MaxRawValue,
    MN_PHYS:=MinPhysValue, MX_PHYS:=MaxPhysValue,
    DIV10:=DivideBy10, ST_CH:=ChannelNumber,
    ST_REG:=StatusRegisterNumber, ST_MODE:=StatusMode,
    ST_HIGH:=StatusInHighByte,
    CHANNEL=>ChannelInformation) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
OUT_REG	UINT	Number of the raw value register (%MW)
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 (%IW)
ST_MODE	UINT	Status mode (e.g. 3=ACO_STATUS_MODE)
ST_HIGH	BOOL	Status byte found in high byte of the register

Description of the output parameters:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	channel information to be described

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 (%MW).
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 (%MW).
E_EFB_USER_ERROR_3 with parameter MN_RAW	MN_RAW ≥ MX_RAW

Error message	Meaning
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

Detailed description

Area of application

The function block can be used in three areas:

1. Raw value scaling, with the blocks: 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 (%MW).

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 used to avoid floating point values in the range 0.2 V ... 1 V. For this area MN_PHYS=2, MX_PHYS=10 and DIV10=TRUE.

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, page 74*.

Evaluation of error information

For the evaluation of error information the inputs ST_CH, ST_REG, 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 (%IW), 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	Not supported
5	DAU204_STATUS_MODE	Not supported
6	ADU205_STATUS_MODE	Not supported
7	AMM090_STATUS_MODE	AMM090
8	ADU214_STATUS_MODE	Not supported

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

Part III
Analog I/O Scaling

Overview

This section describes the elementary functions and elementary function blocks from the family `Analog I/O scaling`.

What Is in This Part?

This part contains the following chapters:

Chapter 8
I_NORM: Standardized analog input

Description

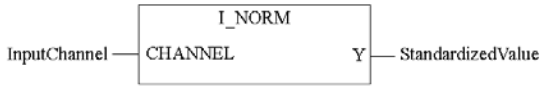
Function description

The function converts data from the 16 bit integer format into the 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 configured as additional parameters.

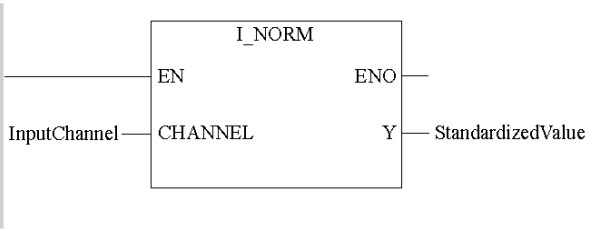
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputChannel
I_NORM
ST StandardizedValue
```

Representation in ST

Representation:

```
StandardizedValue := I_NORM (InputChannel) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`InputChannel`	`ANL_IN`	Input channel

Description of output parameters:

Parameter	Data type	Meaning
`StandardizedValue`	`REAL`	Standardized value

Runtime error

An error message is returned

- 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: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

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

Chapter 9
I_NORM_WARN: Standardized analog-input with warning status

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)

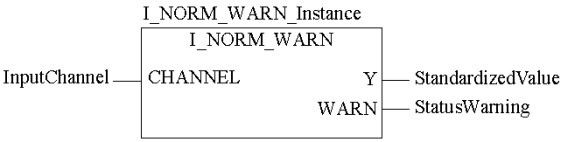
In addition, the function block indicates at the WARN output whether a status warning has occurred in the connected analog input EFB.

EN and ENO can be configured as additional parameters.

NOTE: The I_NORM_WARN function can only be used with the analog input EFBs AII330, AMM090 and AVI030, which are able to generate status information with warnings. Please use the I_NORM function for all other input EFBs

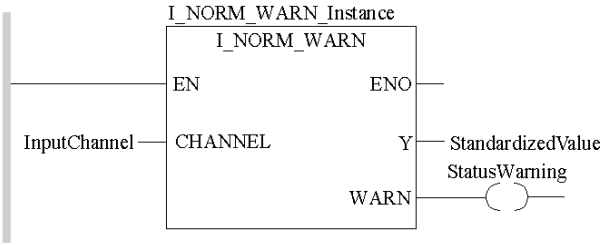
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL I_NORM_WARN_Instance (CHANNEL:=InputChannel,
    Y=>StandardizedValue, WARN=>StatusWarning)
```

Representation in ST

Representation:

```
I_NORM_WARN_Instance (CHANNEL:=InputChannel,
    Y=>StandardizedValue, WARN=>StatusWarning) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input channel

Description of output parameters:

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

Runtime error

An error message is returned

- 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. 6 Volt instead of 0 ... 5 Volt)
- 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.

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Chapter 10
I_PHYS: Physical analog input

Description

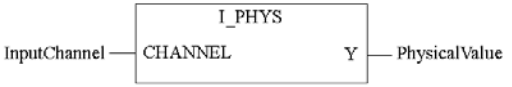
Function description

The function returns analog input values (voltage, current or temperature) as physical values in REAL floating-point format.

EN and ENO can be configured as additional parameters.

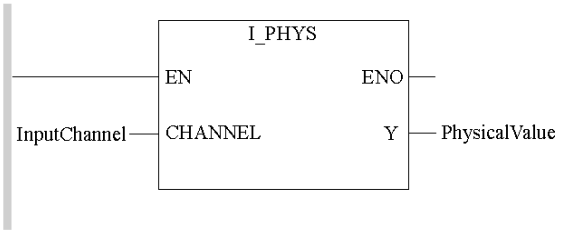
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputChannel
I_PHYS
ST PhysicalValue
```

Representation in ST

Representation:

```
PhysicalValue := I_PHYS (InputChannel) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`InputChannel`	`ANL_IN`	Input channel

Description of output parameters:

Parameter	Data type	Meaning
`PhysicalValue`	`REAL`	Physical value

Runtime error

An error message is returned

- 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: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

NOTE: To evaluate the status information for the I/O module, use the `I_PHYS_WARN` function block.

Chapter 11
I_PHYS_WARN: Physical analog input with warning status

Description

Function description

The function block returns analog input values (voltage, current or temperature) as physical values in REAL floating-point format.

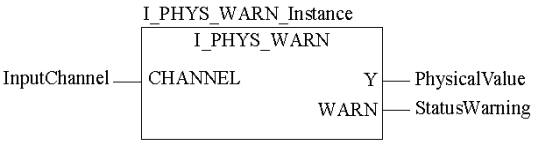
In addition, the function block indicates at the WARN output whether a status warning has occurred in the connected analog input EFB.

EN and ENO can be configured as additional parameters.

NOTE: The I_PHYS_WARN function can only be used with the analog input EFBs, AII330, AMM090 and AVI030, which are able to generate status information with warnings. Please use the I_PHYS function for all other input EFBs.

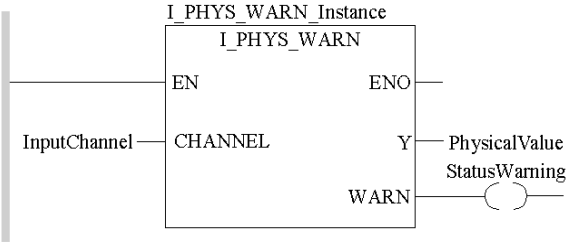
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL I_PHYS_WARN_Instance (CHANNEL:=InputChannel,
    Y=>PhysicalValue, WARN=>StatusWarning)
```

Representation in ST

Representation:

```
I_PHYS_WARN_Instance (CHANNEL:=InputChannel,
    Y=>PhysicalValue, WARN=>StatusWarning) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
CHANNEL	ANL_IN	Input channel

Description of output parameters:

Parameter	Data type	Meaning
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

An error message is returned

- 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. -1 Volt instead of 0 ... 5 Volt).
- with an input value overflow (outside the warning range, e.g. 6 Volt instead of 0 ... 5 Volt).
- 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.

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Chapter 12
I_RAW: Raw value analog input

Description

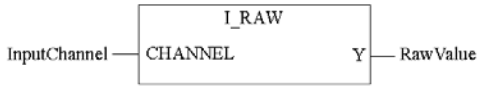
Function description

The function provides analog input values as raw values of the WORD data type.

EN and ENO can be configured as additional parameters.

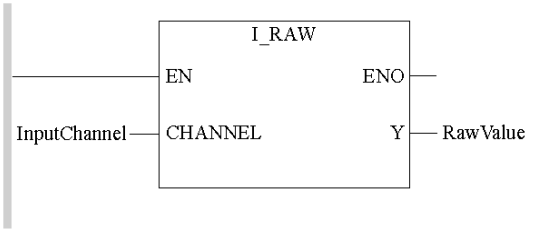
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputChannel
I_RAW
ST RawValue
```

Representation in ST

Representation:

```
RawValue := I_RAW (InputChannel) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`InputChannel`	`ANL_IN`	Input channel

Description of output parameters:

Parameter	Data type	Meaning
`RawValue`	`WORD`	Raw value

Runtime error

An error message is returned

- 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.

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Chapter 13
I_RAWSIM: Simulated raw value analog input

Description

Function description

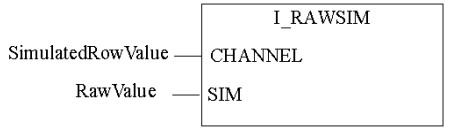
The procedure provides the option to simulate raw value analog inputs.

NOTE: Specify the processing sequence of functions and function blocks so that `I_RAWSIM` is processed before all other functions and function blocks that read the simulated raw value. Link the `ENO`output of `I_RAWSIM` to the `EN` inputs of all functions and function blocks that read the simulated raw value.

As additional parameters, `EN` and `ENO` are projected.

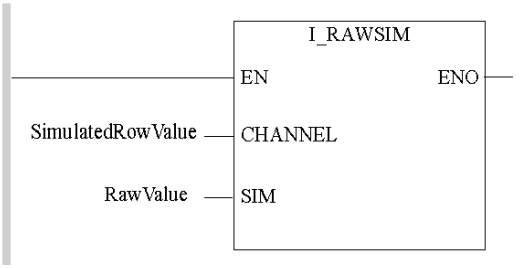
Representation in FBD

Representation:



Representation in LD

Appearance:



Representation in IL

Appearance:

```
I_RAWSIM (CHANNEL:=SimulatedRowValue, SIM:=RawValue)
```

Representation in ST

Appearance:

```
I_RAWSIM (CHANNEL:=SimulatedRowValue, SIM:=RawValue) ;
```

Parameter description

Description of input parameters:

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

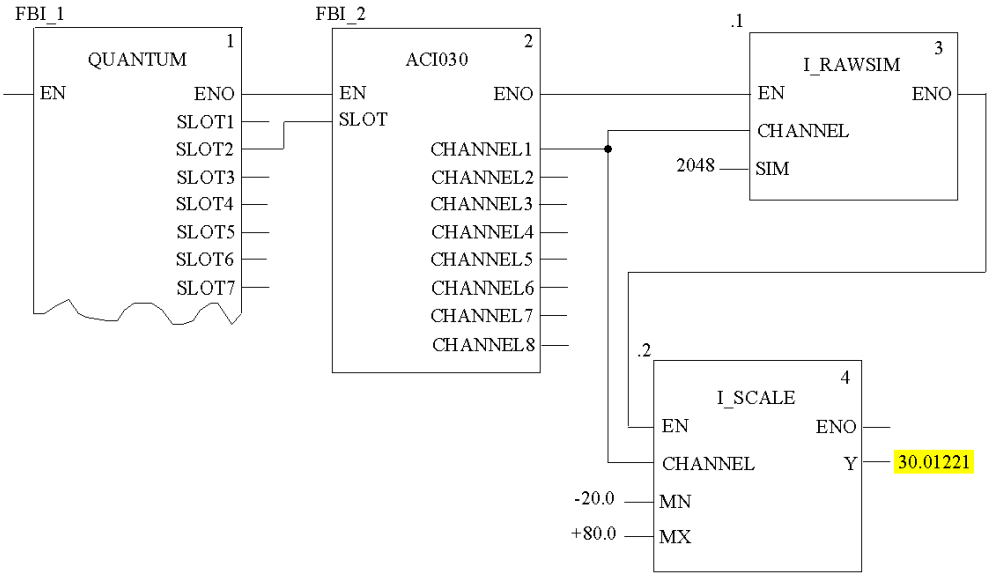
Example

The example shows a configuration with an analog input module (140 ACI 030) with a resolution of 12 bits, which corresponds with a value range (raw value) of 0...4096.

Using the scaling block I_SCALE this value range (raw value) is scaled to a physical value range of -20° C...+80° C.

The procedureI_RAWSIM provides the option to simulate and scale a raw value (e.g. 2048) of the analog input module (30.01221), without actually basing this value on the module.

Example configuration



Runtime error

An error message is returned

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

NOTE: For a list of all error codes and values of the module, see *Analog I/O Scaling, page 315*.

Chapter 14
I_SCALE: Scaled analog input

Description

Function description

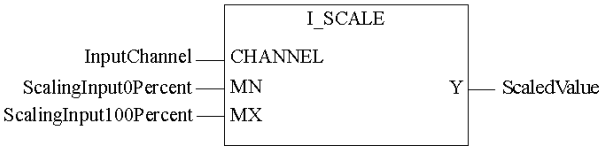
The function converts data from the 16 bit integer format into the REAL floating-point format. The scaling inputs MN and MX predefine the value range for the output. MN corresponds to 0 percent and MX to 100 percent. The integer input value is displayed in the floating-point range. If there are warning 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)

EN and ENO can be configured as additional parameters.

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

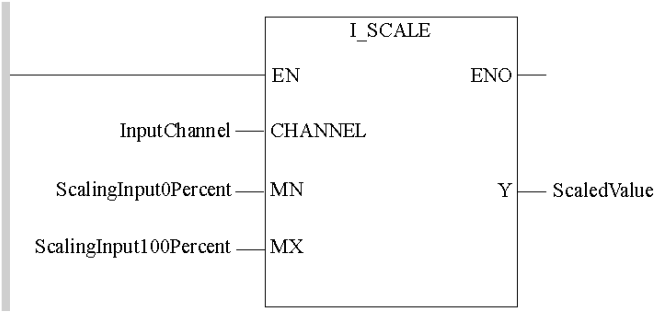
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputChannel
I_SCALE ScalingInput0Percent,ScalingInput100Percent
ST ScaledValue
```

Representation in ST

Representation:

```
ScaledValue := I_SCALE (InputChannel,
    ScalingInput0Percent, ScalingInput100Percent) ;
```

Parameter description

Description of input parameters:

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

Description of output parameters:

Parameter	Data type	Meaning
OUT	REAL	Scaled value

Runtime error

An error message is returned

- 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: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

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

Chapter 15
I_SCALE_WARN: Scaled analog input with warnings status

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)

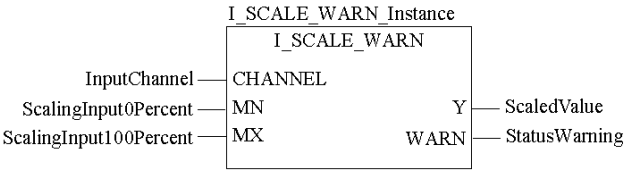
In addition, the function block indicates at the WARN output whether a status warning has occurred in the connected analog input EFB.

EN and ENO can be configured as additional parameters.

NOTE: The I_SCALE_WARN function cannot be used to scale temperature measurements. Please use the I_PHYS_WARN function to scale temperature measurements.
The I_SCALE_WARN function can only be used with the analog input EFBs All330, AMM090 and AVI030, which are able to generate status information with warnings. Please use the I_SCALE function for all other input EFBs.

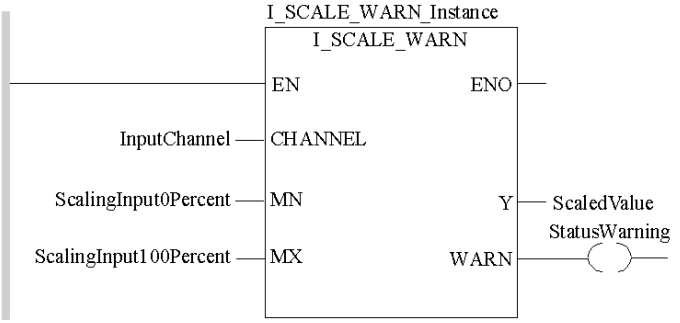
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL I_SCALE_WARN_Instance (CHANNEL:=InputChannel,
    MN:=ScalingInput0Percent, MX:=ScalingInput100Percent,
    Y=>ScaledValue, WARN=>StatusWarning)
```

Representation in ST

Representation:

```
I_SCALE_WARN_Instance (CHANNEL:=InputChannel,
    MN:=ScalingInput0Percent, MX:=ScalingInput100Percent,
    Y=>ScaledValue, WARN=>StatusWarning) ;
```

Parameter description

Description of input parameters:

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

Description of output parameters:

Parameter	Data type	Meaning
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

An error message is returned to the diagnostic buffer

- 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. -1 Volt instead of 0 ... 5 Volt).
- with an input value overflow (outside the warning range, e.g. 6 Volt instead of 0 ... 5 Volt).
- if the connected analog input EFB is not AII330, AMM090 or AVI030.

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Chapter 16
O_NORM: Standardized analog output

Description

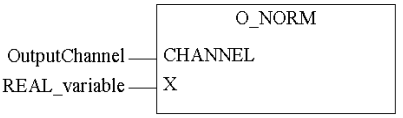
Function description

The procedure returns values from the REAL floating point format 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 configured as additional parameters.

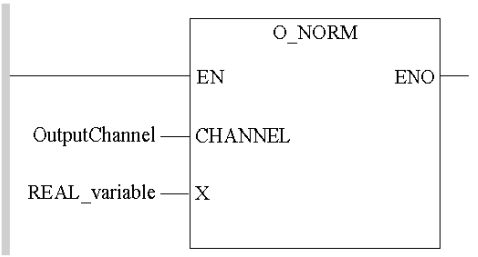
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD OutputChannel
O_NORM REAL_variable
```

Representation in ST

Representation:

```
O_NORM (OutputChannel, REAL_variable);
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`OutputChannel`	`ANL_OUT`	Output channel
`REAL_variable`	`REAL`	Standardized value

Runtime error

An error message is returned

- 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: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

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

Chapter 17
O_NORM_WARN: Standardized analog output with warning status

Description

Function description

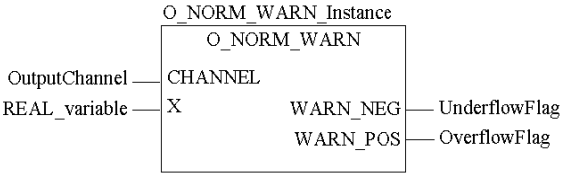
The function block returns values from the REAL floating point format as analog values in 16 bit integer format. The floating point value in the range of 0.0 to 1.0 is displayed onto the configured integer output value.

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.

EN and ENO can be configured as additional parameters.

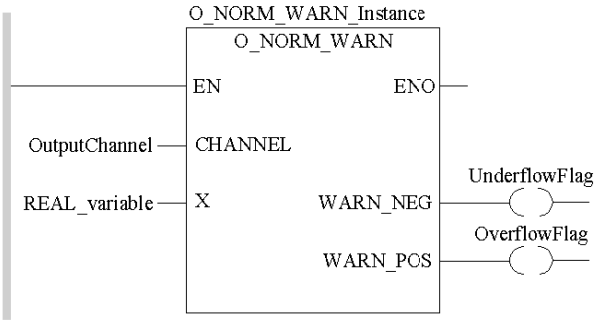
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL O_NORM_WARN_Instance (CHANNEL:=OutputChannel,
    X:=REAL_variable, WARN_NEG=>UnderflowFlag,
    WARN_POS=>OverflowFlag)
```

Representation in ST

Representation:

```
O_NORM_WARN_Instance (CHANNEL:=OutputChannel,
    X:=REAL_variable, WARN_NEG=>UnderflowFlag,
    WARN_POS=>OverflowFlag) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
CHANNEL	ANL_OUT	Output channel
X	REAL	Standardized value

Description of output parameters:

Parameter	Data type	Meaning
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

An error message is returned

- 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, e.g. -0.1 V instead of 0 ... 1.0 V) 1.0 Volt)
- with an output value overflow (arithmetic) (outside the warning range, e.g. 1.1 instead of 0 ... 1.0 V) 1.0 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_NORM` procedure.

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Chapter 18
O_PHYS: Physical analog output

Description

Function description

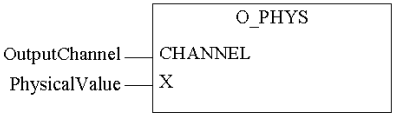
The procedure provides analog input values (voltage, current or temperature) as physical values in REAL floating-point format.

The function block is used for output modules with configuration information.

EN and ENO can be configured as additional parameters.

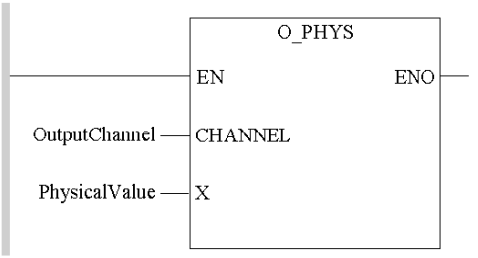
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD OutputChannel
O_PHYS PhysicalValue
```

Representation in ST

Representation:

```
O_PHYS (OutputChannel, PhysicalValue);
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`OutputChannel`	`ANL_OUT`	Output channel
`PhysicalValue`	`REAL`	Physical value

Runtime error

An error message is returned

- 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: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

NOTE: To evaluate the status information for the I/O module, use the `O_PHYS_WARN` *(see page 109)*function block.

Chapter 19
O_PHYS_WARN: Physical analog output with warning-status

Description

Function description

The function block provides analog input values (voltage, current or temperature) as physical values in REAL floating-point format.

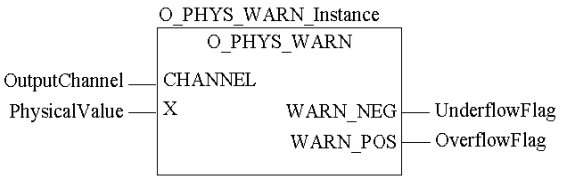
The function block is used for output modules with configuration information.

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.

EN and ENO can be configured as additional parameters.

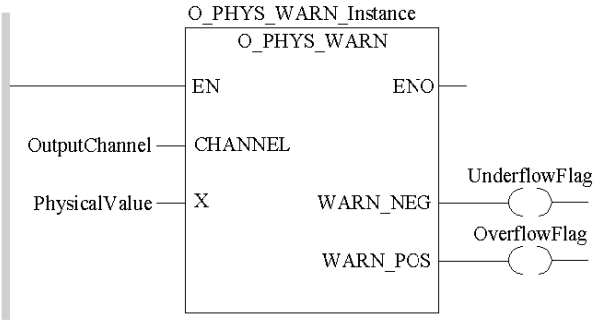
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL O_PHYS_WARN_Instance (CHANNEL:=OutputChannel,
    X:=PhysicalValue, WARN_NEG=>UnderflowFlag,
    WARN_POS=>OverflowFlag)
```

Representation in ST

Representation:

```
O_PHYS_WARN_Instance (CHANNEL:=OutputChannel,
    X:=PhysicalValue, WARN_NEG=>UnderflowFlag,
    WARN_POS=>OverflowFlag) ;
```

Parameter description

Description of input parameters:

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

Description of output parameters:

Parameter	Data type	Meaning
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

An error message is returned

- 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 procedure.

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Chapter 20
O_RAW: Raw value analog output

Description

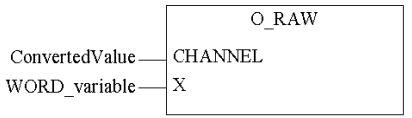
Function description

The procedure returns raw values of the WORD data type as analog output values.

EN and ENO can be configured as additional parameters.

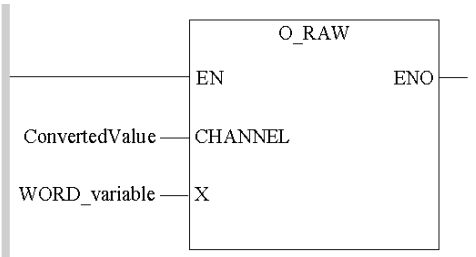
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD ConvertedValue
O_RAW WORD_variable
```

Representation in ST

Representation:

```
O_RAW (ConvertedValue, WORD_variable);
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
ConvertedValue	ANL_OUT	Output value
WORD_variable	WORD	Raw value

Runtime error

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

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Chapter 21
O_SCALE: Scaled analog output

Description

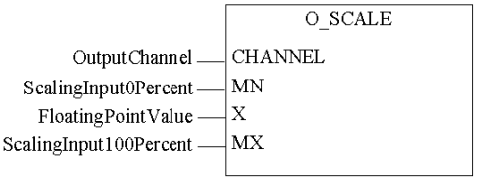
Function description

The procedure converts values from the REAL floating point format into 16 bit integer format. The scaling inputs `ScalingInput0Percent` and `ScalingInput100Percent` define the value range for the analog output. Accordingly, `ScalingInput0Percent` corresponds to 0 percent and `ScalingInput100Percent` to 100 percent of the output range (e.g. -10 ...10 V).

EN and ENO can be configured as additional parameters.

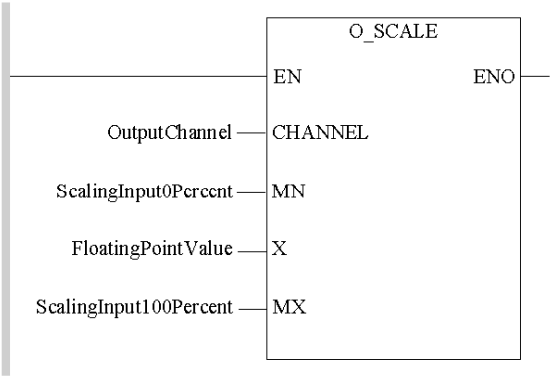
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD OutputChannel
O_SCALE ScalingInput0Percent, FloatingPointValue,
        ScalingInput100Percent
```

Representation in ST

Representation:

```
O_SCALE (OutputChannel, ScalingInput0Percent,
         FloatingPointValue, ScalingInput100Percent);
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`OutputChannel`	`ANL_OUT`	Output channel
`ScalingInput0Percent`	`REAL`	Scaling input, 0 percent
`FloatingPointValue`	`REAL`	Floating-point value
`ScalingInput100Percent`	`REAL`	Scaling input, 100 percent

Runtime error

An error message is returned

- if the output channel is not configured. In this case, please check the connected I/O module EFB.
- if the values of `ScalingInput0Percent` and `ScalingInput100Percent` 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: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

NOTE: To evaluate the status information for the module, use the `O_SCALE_WARN` function block.

Chapter 22
O_SCALE_WARN: Scaled analog output with warnings status

Description

Function description

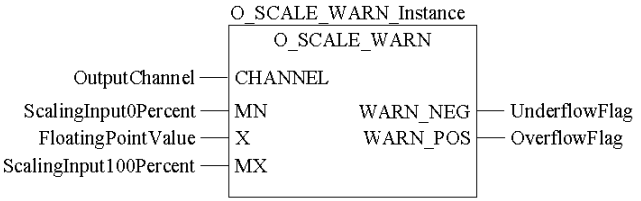
The function block converts values from the REAL floating point format into 16 bit integer format. The scaling inputs MN and MX predefine the value range for the analog output. MN corresponds to 0 percent and MX to 100 percent of the output range (e.g. -10 ...10 V).

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.

EN and ENO can be configured as additional parameters.

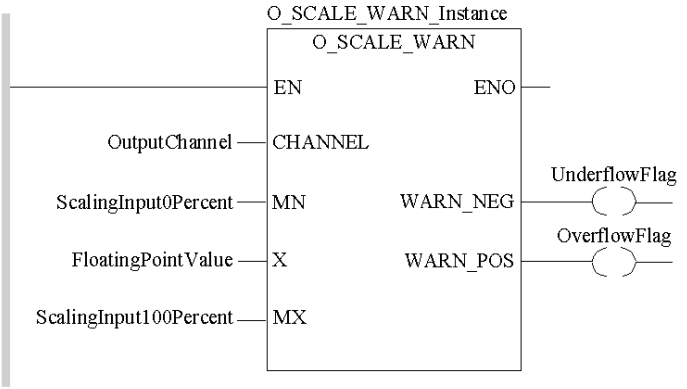
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL O_SCALE_WARN_Instance (CHANNEL:=OutputChannel,
    MN:=ScalingInput0Percent, X:=FloatingPointValue,
    MX:=ScalingInput100Percent,  WARN_NEG=>UnderflowFlag,
    WARN_POS=>OverflowFlag)
```

Representation in ST

Representation:

```
O_SCALE_WARN_Instance (CHANNEL:=OutputChannel,
    MN:=ScalingInput0Percent, X:=FloatingPointValue,
    MX:=ScalingInput100Percent,  WARN_NEG=>UnderflowFlag,
    WARN_POS=>OverflowFlag) ;
```

Parameter description

Description of input parameters:

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

Description of output parameters:

Parameter	Data type	Meaning
WARN_NEG	BOOL	0: no output value underflow at the closed analog output EFB 1: output value underflow at the closed analog output EFB X < MN
WARN_POS	BOOL	0: no output value overflow at the closed analog output EFB 1: output value overflow at the closed analog output EFB X > MX

Runtime error

An error message is returned

- 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 procedure.

NOTE: For a list of all block error codes and values, see*Analog I/O Scaling, page 315*.

Part IV
Explicit Exchange

Overview

This section describes the elementary functions and elementary function blocks from the family `Explicit exchange`.

What Is in This Part?

This part contains the following chapters:

Chapter 23
Operation and management of explicit exchanges

Subject of this Chapter

This chapter describes the operation and management of explicit exchanges.

What Is in This Chapter?

This chapter contains the following topics:

Explicit Exchanges: General

Introduction

Explicit exchanges are exchanges performed at the request of the user program by using the following functions:

- READ_STS (read status words)
- WRITE_CMD (write command words)
- READ_PARAM (read adjustment parameters)
- WRITE_PARAM (write adjustment parameters)
- SAVE_PARAM (save adjustment parameters)
- RESTORE_PARAM (restore adjustment parameters)
- READ_TOPO_ADDR (read topological address)
- TRF_RECIPE (recipe transfer)

These exchanges apply to a set of objects (status, commands, or parameters) of a single channel. Their parameter is an IODDT-type variable.

For a Quantum Device DDT-type variable, use:

- READ_STS_QX (for Quantum to read Modicon M340 drop status words)
- WRITE_CMD_QX (for Quantum to send command to Modicon M340 drop)

For a Modicon M580 Device DDT-type variable, use:

- READ_STS_MX (for M580 to read (e)X80 local rack or Ethernet RIO module status words)
- WRITE_CMD_MX (for M580 to send command to (e)X80 local rack or Ethernet RIO module)
- READ_PARAM_MX (read parameters of a local rack module)
- WRITE_PARAM_MX (write parameters to a local rack module)
- SAVE_PARAM_MX (save parameters of a local rack module)
- RESTORE_PARAM_MX (restore parameters of a local rack module)

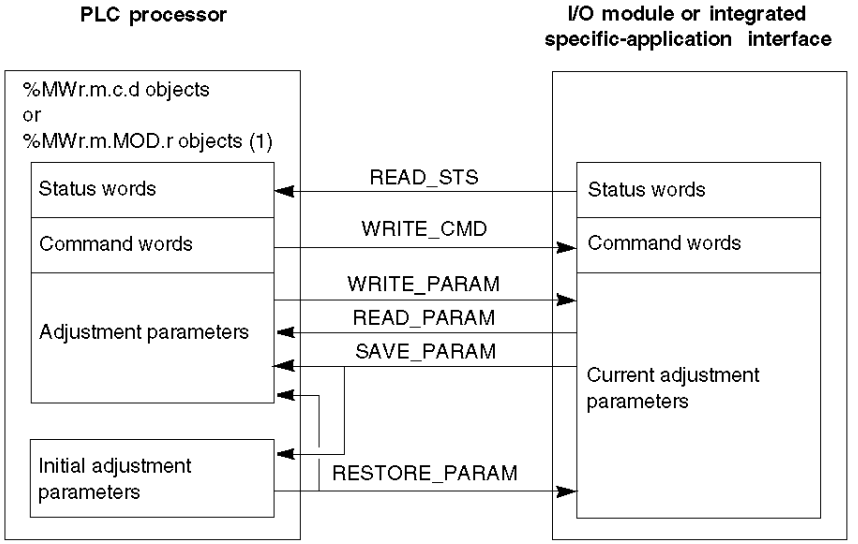
NOTE: These objects are not essential to the programming of a specific-application function, but they do provide additional information (e.g.: terminal faulty, module missing, etc.) as well as additional commands for advanced programming of specific-application functions (for further information on application-specific explicit exchange objects, a chapter dedicated to language objects is featured in each specific-application manual).

NOTE: The following functions also perform explicit exchanges, their operation being specific to the specific-application concerned (parameters and management of the different exchanges). The functions are detailed in each specific-application manual concerned, and a brief overview is provided in each library:

- DETAIL_OBJECT *(see Unity Pro, Drive control, Block Library)*
- SMOVE *(see Unity Pro, Drive control, Block Library)*
- XMOVE *(see Unity Pro, Drive control, Block Library)*
- MOD_CAM *(see Unity Pro, Drive control, Block Library)*
- MOD_PARAM *(see Unity Pro, Drive control, Block Library)*
- MOD_TRACK *(see Unity Pro, Drive control, Block Library)*

General principle for using explicit instructions

The illustration below shows the different types of explicit exchanges that can take place between the PLC processor and the module (or integrated interface):



(1) Only with the READ_STS instruction.

Managing exchanges

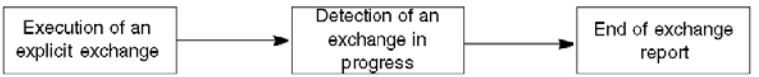
It may prove beneficial during an explicit exchange to check its execution, so as to ensure, for example, that the data read is only taken into account once the exchange has been correctly performed.

For this, two types of information are available:

- detection of an exchange in progress
- end of exchange report

NOTE: You can't send two explicit exchanges at the same time to the channels managed by the same Logical Node LN0 ch0&1; LN1 ch 2&3; LN2 ch 4&5; LN3 ch 6&7.The Logical Node can only process one request, the other request will generate an error.

The following diagram illustrates the principle for managing exchanges:



IODDT or %CHr.m.c logical channel

An IODDT associated with a %CHr.m.c channel or the representation of a %CHr.m.c channel is a general syntax for updating all objects of the same type associated with this channel or group of channels, by way of explicit instructions.

Example: READ_STS(IODDT_Var)or READ_STS(%CH1.2.3) (read status words of channel 3 of module 2 located in rack 1).

NOTE: For a group of channels, any channel of the group can be used, but the exchange will apply to all channels of the group (Analog and Discrete) READ_STS(%CH3.2.8)will have the same effect as READ_STS(%CH3.2.9).

Limit regarding the Fipio bus

The number of explicit exchange functions that can be activated simultaneously on the Fipio bus is limited to 24.

An exchange request addressed to the Fipio bus may take several cycles of the master task, thus making it necessary to manage the words of the exchange management parameters for all explicit variable exchanges.

Management of Exchanges and Reports with Explicit Objects

At a Glance

When data is exchanged between the PLC memory and the module, the module may require several task cycles to acknowledge this information. IODDTs use two words to manage exchanges:

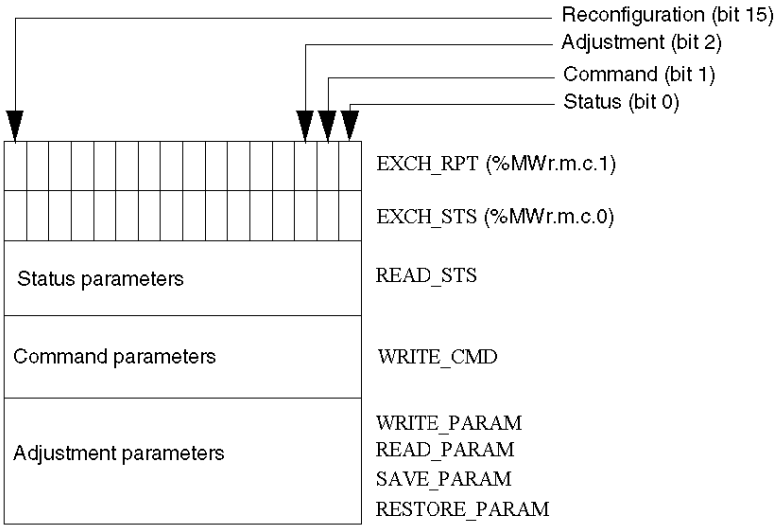
- EXCH_STS (%MWr.m.c.0): exchange in progress
- EXCH_RPT (%MWr.m.c.1): report

NOTE:
Depending on the localization of the module, the management of the explicit exchanges (%MW0.0.MOD.0.0 for example) will not be detected by the application:

- For in-rack modules, explicit exchanges are done immediately on the local PLC Bus and are finished before the end of the execution task. So, the READ_STS, for example, is finished when the %MW0.0.mod.0.0 bit is checked by the application.
- For remote bus (Fipio for example), explicit exchanges are not synchronous with the execution task, so the detection is possible by the application.

Illustration

The illustration below shows the different significant bits for managing exchanges:



Description of Significant Bits

Each bit of the words EXCH_STS (%MWr.m.c.0) and EXCH_RPT (%MWr.m.c.1) is associated with a type of parameter:

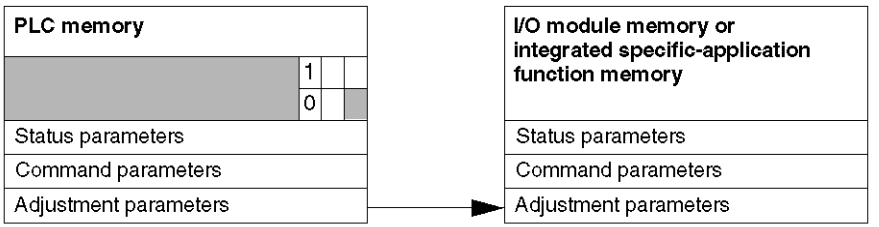
- Rank 0 bits are associated with the status parameters:
  - The STS_IN_PROGR bit (%MWr.m.c.0.0) indicates whether a read request for the status words is in progress.
  - The STS_ERR bit (%MWr.m.c.1.0) specifies whether a read request for the status words is accepted by the module channel.
- Rank 1 bits are associated with the command parameters:
  - The CMD_IN_PROGR bit (%MWr.m.c.0.1) indicates whether command parameters are being sent to the module channel.
  - The CMD_ERR bit (%MWr.m.c.1.1) specifies whether the command parameters are accepted by the module channel.
- Rank 2 bits are associated with the adjustment parameters:
  - The ADJ_IN_PROGR bit (%MWr.m.c.0.2) indicates whether the adjustment parameters are being exchanged with the module channel (via WRITE_PARAM, READ_PARAM, SAVE_PARAM, RESTORE_PARAM).
  - The ADJ_ERR bit (%MWr.m.c.1.2) specifies whether the adjustment parameters are accepted by the module. If the exchange is correctly executed, the bit is set to 0.
- Rank 15 bits indicate a reconfiguration on channel **c** of the module from the console (modification of the configuration parameters + cold start-up of the channel).
- The *r*, *m* and *c* bits indicates the following elements:
  - the **r** bit represents the rack number.
  - The **m** bit represents the position of the module in the rack.
  - The **c** bit represents the channel number in the module.

NOTE: r represents the rack number, **m** the position of the module in the rack, while **c** represents the channel number in the module.

NOTE: Exchange and report words also exist at module level EXCH_STS (%MWr.m.MOD) and EXCH_RPT (%MWr.m.MOD.1) as per IODDT type T_GEN_MOD.

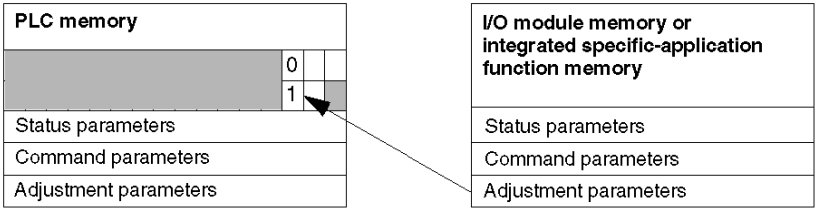
Example

Phase 1: Sending data by using the WRITE_PARAM instruction



When the instruction is scanned by the PLC, the **Exchange in progress** bit is set to 1 in %MWr.m.c.

Phase 2: Analysis of the data by the I/O module and report.



When the data is exchanged between the PLC memory and the module, acknowledgement by the module is managed by the ADJ_ERR bit (%MWr.m.c.1.2).

This bit makes the following reports:

- **0:** correct exchange
- **1:** incorrect exchange)

NOTE: There is no adjustment parameter at module level.

Execution Indicators for an Explicit Exchange: EXCH_STS

The table below shows the control bits of the explicit exchanges: EXCH_STS (%MWr.m.c.0)

Standard Symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Reading of channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
RECONF_IN_PROGR	BOOL	R	Reconfiguration of the module in progress	%MWr.m.c.0.15

NOTE: If the module is not present or is disconnected, explicit exchange objects (READ_STS for example) are not sent to the module (STS_IN_PROG (%MWr.m.c.0.0) = 0), but the words are refreshed.

Explicit Exchange Report: EXCH_RPT

The table below shows the report bits: EXCH_RPT (%MWr.m.c.1)

Standard Symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Error detected while reading channel status words (1 = detected error)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Error detected during a command parameter exchange (1 = detected error)	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Error dectected during an adjust parameter exchange (1 = detected error)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Error detected during reconfiguration of the channel (1 = detected error)	%MWr.m.c.1.15

Counting Module Use

The following table describes the steps realized between a couting module and the system after a power-on.

Step	Action
1	Power on.
2	The system sends the configuration parameters.
3	The system sends the adjust parameters by WRITE_PARAM method. **Note:** When the operation is finished, the bit %MWr.m.c.0.2 switches to 0.

If, in the begining of your application, you use a WRITE_PARAM command, wait until the bit %MWr.m.c.0.2 switches to 0.

Chapter 24
READ_PARAM: Reading the adjustment parameters

Description

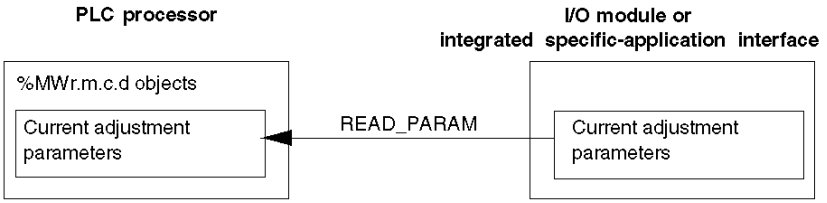
Description of the function

The READ_PARAM function is used to read the adjustment parameters of a module or integrated interface, by performing an explicit exchange with the processor memory.

The additional parameters EN and ENO may also be configured.

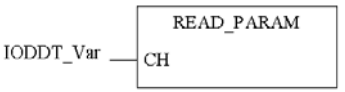
Operation

Operation flow diagram:



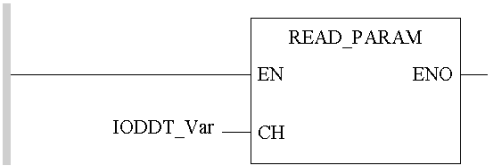
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
READ_PARAM
```

ST representation

Representation:

```
READ_PARAM(IODDT_Var);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	IODDT-type variable corresponding to the channel or module whose adjustment parameters the user wishes to read. **Example e**: `IODDT_Var` of type `T_COUNT_ACQ`

Chapter 25
READ_PARAM_MX: Reading the Parameters on Local Rack

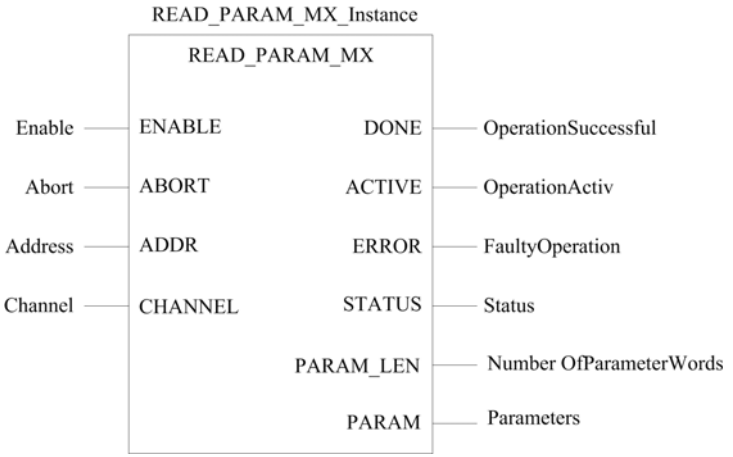
Description

Description of the function

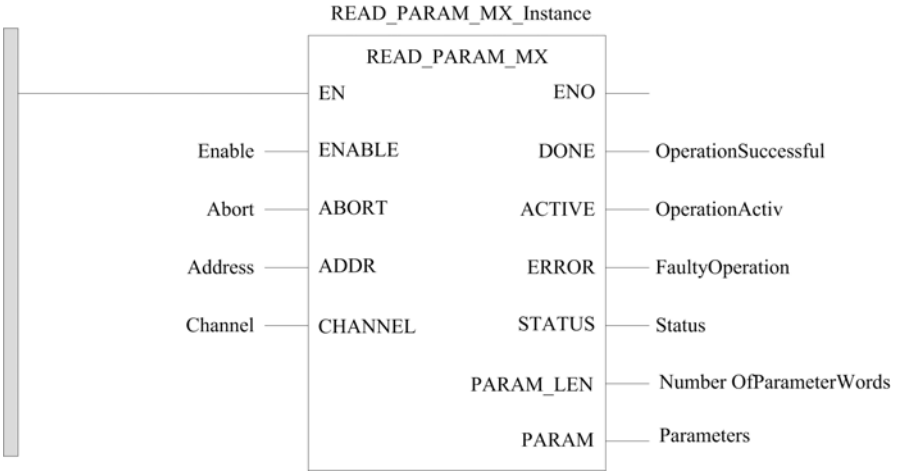
The READ_PARAM_MX function is used to read the parameters of a module declared in Device DDT in the local rack. It performs an explicit exchange *(see page 124)* with the Modicon M580 CPU memory.

The additional parameters EN and ENO can also be configured.

FBD representation



LD representation



IL representation

Representation:

```
CAL READ_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, DONE=>OperationSuccessful, ACTIVE=>OperationActiv,
ERROR=>FaultyOperation, STATUS=>Status, PARAM_LEN=>NumberOfParameterWords,
PARAM=>Parameters)
```

ST representation

Representation:

```
READ_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, DONE=>OperationSuccessful, ACTIVE=>OperationActiv,
ERROR=>FaultyOperation, STATUS=>Status, PARAM_LEN=>NumberOfParameterWords,
PARAM=>Parameters);
```

Parameter Description

The following table describes the input parameters:

Parameter	Data Type	Significance
Enable	BOOL	Set to 1 to trigger the operation
Abort	BOOL	Set to 1 to abort the currently active operation
Address	ANY ARRAY OF INT	Array that contains the address result of ADDMX function (1) and identifies the rack which the module belongs to.
Channel	STRING	Identifies the channel to work on; it contains rack, slot and channel numbers (r.s.c)
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Data Type	Significance
OperaionSucessful	BOOL	Operation terminated successfully
OperationActiv	BOOL	Operation active
FaultyOperation	BOOL	Operation terminated unsuccessfully
Status	WORD	Detected error identifier *(see page 323)*
NumberOfParameterWords	INT	Number of parameter registers/words to read
Parameters	ANY	Parameter registers/words of the channel. Because, they are module-specific, refer to the module manual for a detailed description. A device DDT-type variable may be used

Chapter 26
READ_STS: Reading the status parameters

Description

Description of the function

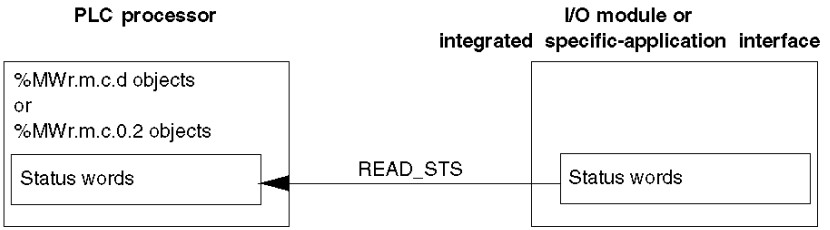
The READ_STS function is used to read the status words of a module or integrated interface, by performing an explicit exchange with the processor memory.

These status words contain data on the operating mode of the module, and may be used to perform program-based diagnostics.

The additional parameters EN and ENO may also be configured.

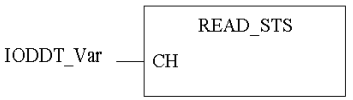
Operation

Operation flow diagram:



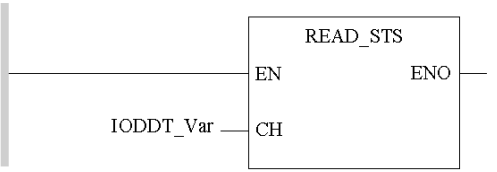
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
READ_STS
```

ST representation

Representation:

```
READ_STS(IODDT_Var);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	IODDT-type variable corresponding to the channel or module whose status words the user wishes to read. **Example**: `IODDT_Var` of type `T_COUNT_ACQ`

Chapter 27
READ_STS_QX: Reading the Status Parameters on EIO Bus

Description

Description of the function

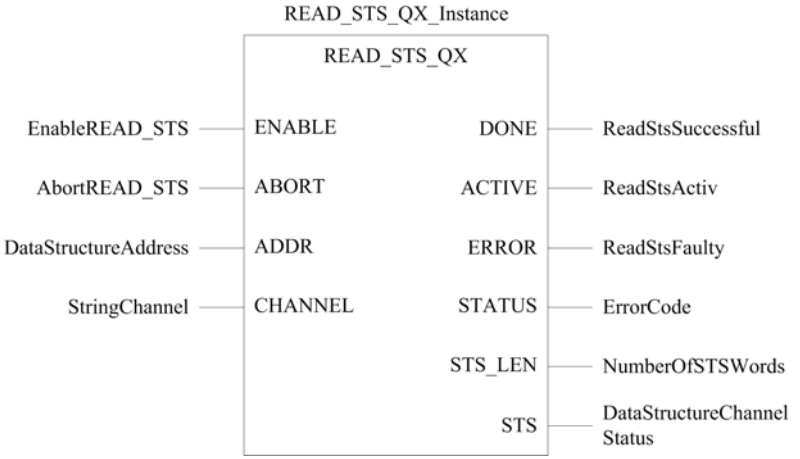
The READ_STS_QX function block is used to read the status words of a Modicon M340 Ethernet I/O module, by performing an explicit exchange with the processor memory.

These status words contain data on the operating mode of the module, and may be used to perform program-based diagnostics.

The additional parameters EN and ENO may also be configured.

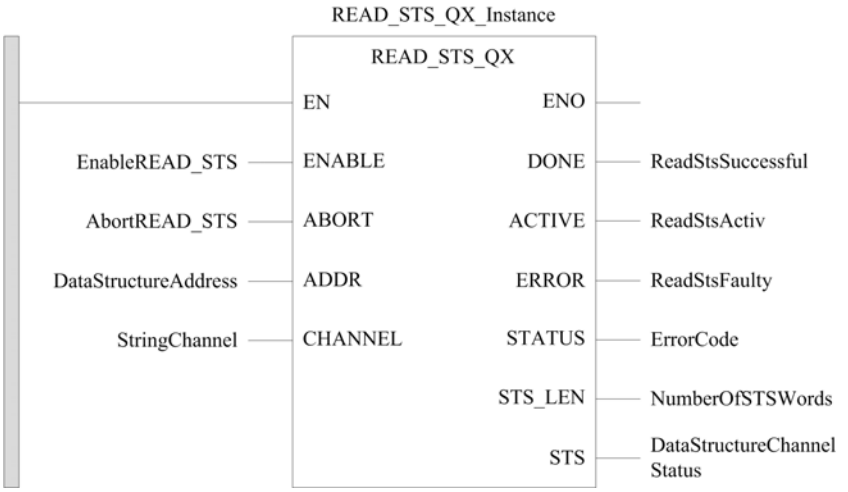
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
CAL READ_STS_QX_Instance (ENABLE:=EnableREAD_STS, ABORT:=AbortREAD_STS,
ADDR:=DataStructureAddress, CHANNEL:=StringChannel, DONE=>ReadStsSuc-
cessful, ACTIVE=>ReadStsActiv, ERROR=>ReadStsFaulty, STATUS=>ErrorCode,
STS_LEN=>NumberOfSTSWords, STS=>DataStructureChannelStatus)
```

ST representation

Representation:

```
READ_STS_QX_Instance (ENABLE:=EnableREAD_STS, ABORT:=AbortREAD_STS,
ADDR:=DataStructureAddress, CHANNEL:=StringChannel, DONE=>ReadStsSuc-
cessful, ACTIVE=>ReadStsActiv, ERROR=>ReadStsFaulty, STATUS=>ErrorCode,
STS_LEN=>NumberOfSTSWords, STS=>DataStructureChannelStatus);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
EnableREAD_STS	BOOL	Set to 1 to read the status words.
AbortREAD_STS	BOOL	Set to 1 to abort the current operation.
DataStructureAddress	ANY_ARRAY_INT	Identifies the Modicon M340 drop which the module belongs to, result of ADDMX *(see Unity Pro, Communication, Block Library)* function (1).
StringChannel	STRING	Identifies the channel to work upon. Contains the rack, slot and channel numbers as follows: 'r.s.c'. If the channel is omitted, then the module server is addressed.
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Type	Comment
ReadStsSuccessful	BOOL	Operation completed indication. Set to 1 when the execution of the operation is completed successfully.
ReadStsActiv	BOOL	Operation in progress indication. Set to 1 when the execution of the operation is in progress.
ReadStsFaulty	BOOL	Set to 1 if an error is detected by the function block.
ErrorCode	WORD	Code providing the detected error identification *(see page 323)*.
NumberOfSTSWords	INT	Number of status words read.
DataStructure-ChannelStatus	ANY	Channel status word. Number of status words to read. Instance of a DDT applicable to a specific IO channel status. The status word description is provided in each EIO device manual.

Chapter 28
READ_STS_MX: Reading the Status Parameters on EIO Bus

Description

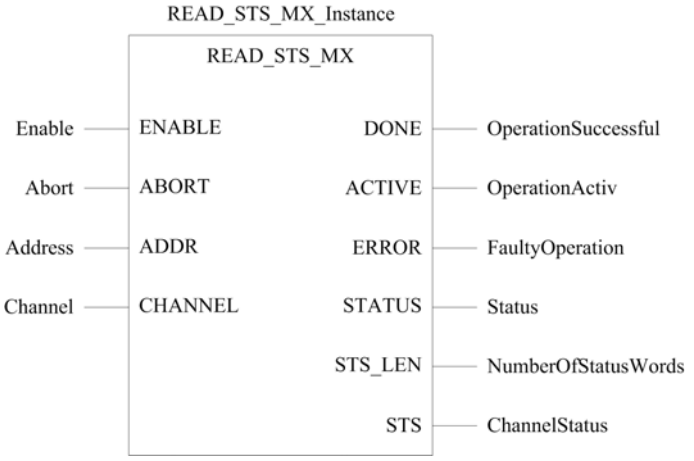
Description of the function

The READ_STS_MX function block is used to read the status words of an (e)X80 local rack or Ethernet RIO module, by performing an explicit exchange with the CPU memory.

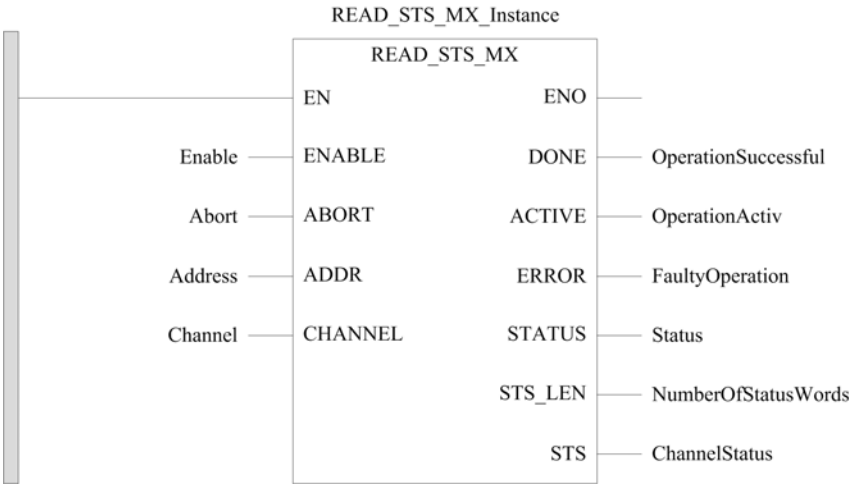
These status words contain data on the operating mode of the module, and may be used to perform program-based diagnostics.

The additional parameters EN and ENO may also be configured.

FBD representation



LD representation



IL representation

```
CAL READ_STS_MX_Instance (ENABLE:=EnableREAD_STS, ABORT:=AbortREAD_STS,
ADDR:=DataStructureAddress, CHANNEL:=StringChannel, DONE=>ReadStsSuc-
cessful, ACTIVE=>ReadStsActiv, ERROR=>ReadStsFaulty, STATUS=>ErrorCode,
STS_LEN=>NumberOfSTSWords, STS=>DataStructureChannelStatus)
```

ST representation

```
READ_STS_MX_Instance (ENABLE:=EnableREAD_STS, ABORT:=AbortREAD_STS,
ADDR:=DataStructureAddress, CHANNEL:=StringChannel, DONE=>ReadStsSuc-
cessful, ACTIVE=>ReadStsActiv, ERROR=>ReadStsFaulty, STATUS=>ErrorCode,
STS_LEN=>NumberOfSTSWords, STS=>DataStructureChannelStatus);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
EnableREAD_STS	BOOL	Set to 1 to enable the operation.
AbortREAD_STS	BOOL	Set to 1 to abort the currently active operation.
DataStructureAddress	ANY_ARRAY_INT	Identifies the (e)X80 drop which the module belongs to, result of ADDMX *(see Unity Pro, Communication, Block Library)* function (1).
StringChannel	STRING	Identifies the channel to work upon. Contains the rack, slot and channel numbers as follows: 'r.s.c'. If the channel is omitted, then the module server is addressed.
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Type	Comment
ReadStsSuccessful	BOOL	Operation completed indication. Set to 1 when the execution of the operation is completed successfully.
ReadStsActiv	BOOL	Operation in progress indication. Set to 1 when the execution of the operation is in progress.
ReadStsFaulty	BOOL	Set to 1 if an error is detected by the function block.
ErrorCode	WORD	Code providing the detected error identification *(see page 323)*.
NumberOfSTSWords	INT	Number of status registers/words read.
DataStructureChannelStatus	ANY	Channel status register/word. A DDDT-type variable may be used. The status register/word description is provided in each EIO device manual.

Chapter 29
READ_TOPO_ADDR: Reading the topological address

Description

Description of the function

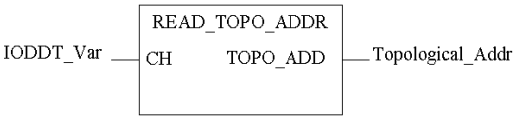
The READ_TOPO_ADDR function extracts from an IODDT-type variable the topological address associated with it. This address is stored in a table.

This function applies to all channel-type IODDTs which can be used in a DFB.

The additional parameters EN and ENO may also be configured.

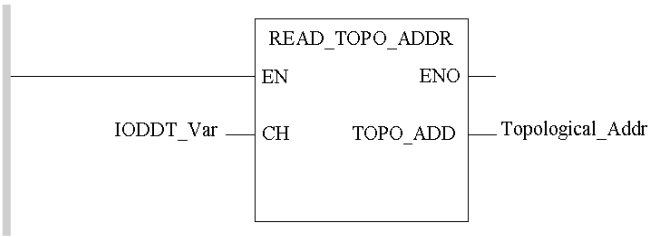
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
READ_TOPO_ADDR
ST Topological_Addr
```

ST representation

Representation:

```
Topological_Addr := READ_TOPO_ADDR(IODDT_Var);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	Channel-type IODDT variable whose physical location (topological address) you wish to ascertain. **Examples**: `IODDT_Var` of type `T_COUNT_STD`.

The following table describes the output parameters:

Parameter	Type	Comment
`Topological_Addr`	TOPO_ADDR_TYPE	Table of 5 integers representing the topological address of the `IODDT_Var` variable. `Topological_Addr[0]`: bus number (0 if the channel is located on an in-rack module). `Topological_Addr[1]`: device number (0 if the channel is located on an in-rack module). `Topological_Addr[2]`: rack number. `Topological_Addr[3]`: module number. `Topological_Addr[4]`: channel number.

Chapter 30
RESTORE_PARAM: Restoring the adjustment parameters

Description

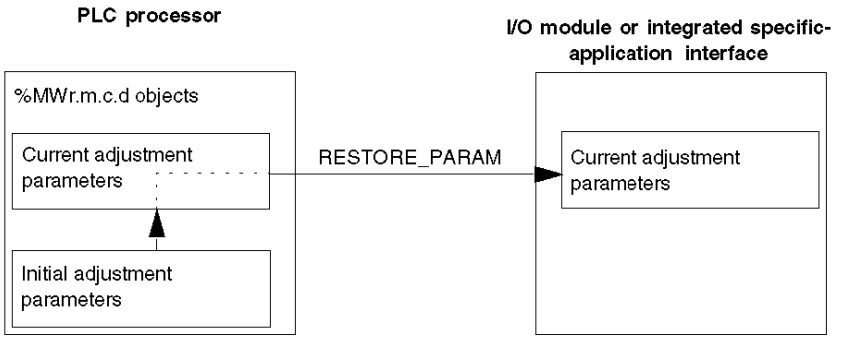
Description of the function

The RESTORE_PARAM function is used to restore in the module or the integrated interface the initial adjustment parameters written during configuration or at the time of the last save (SAVE_PARAM function).

The additional parameters EN and ENO may also be configured.

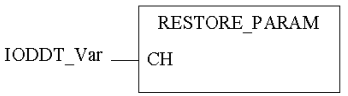
Operation

Operation flow diagram:



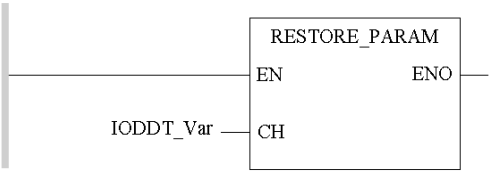
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
RESTORE_PARAM
```

ST representation

Representation:

```
RESTORE_PARAM(IODDT_Var);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	IODDT-type variable corresponding to the channel or module whose adjustment parameters the user wishes to restore. **Example**: `IODDT_Var` of type `T_COUNT_ACQ`

Chapter 31
RESTORE_PARAM_MX: Restoring the Parameters in Local Rack

Representation

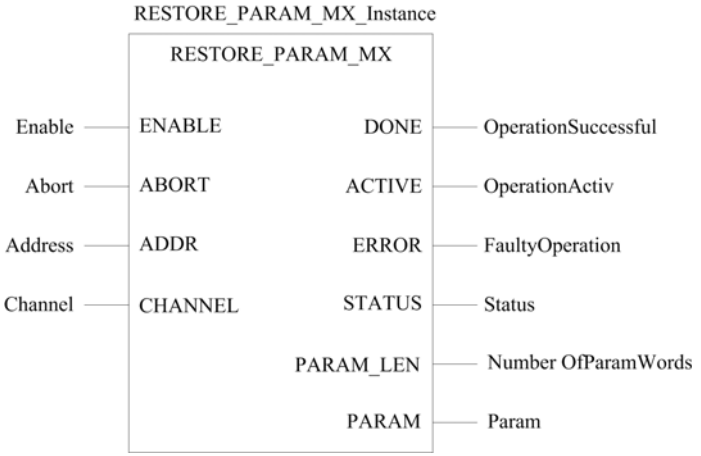
Description

The RESTORE_PARAM_MX function is used to restore the new parameters of a module declared in Device DDT in the local rack. It performs an explicit exchange *(see page 124)* with the Modicon M580 CPU memory.

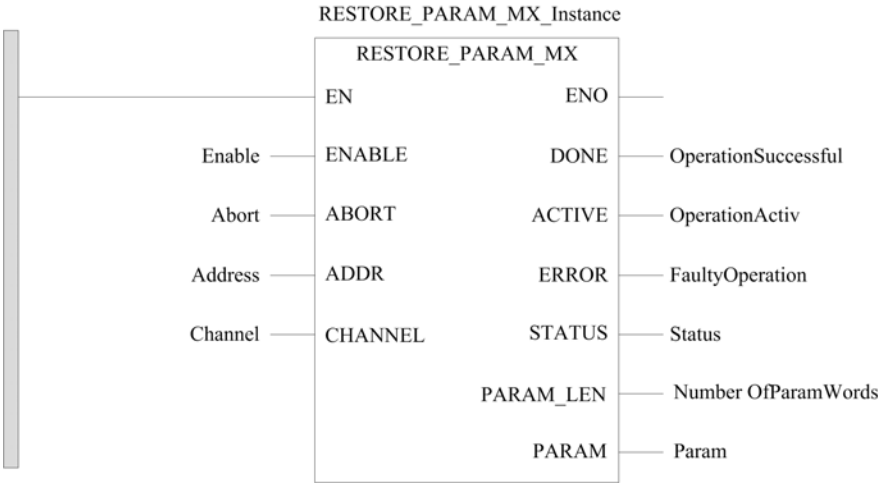
These parameters restore the values saved using SAVE_PARAM_MX.

The additional parameters EN and ENO can also be configured.

FBD Representation



LD Representation



IL representation

Representation:

```
CAL RESTORE_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, DONE=>OperationSuccessful, ACTIVE=>OperationActiv,
ERROR=>FaultyOperation, STATUS=>Status, PARAM_LEN=>NumberOfParamWords,
PARAM=>Param)
```

ST representation

Representation:

```
RESTORE_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, DONE=>OperationSuccessful, ACTIVE=>OperationActiv,
ERROR=>FaultyOperation, STATUS=>Status, PARAM_LEN=>NumberOfParamWords,
PARAM=>Param);
```

Parameter Description

The following table describes the input parameters:

Parameter	Data Type	Significance
Enable	BOOL	Set to 1 to trigger the operation
Abort	BOOL	Set to 1 to abort the currently active operation
Address	ANY ARRAY OF INT	Array that contains the address result of ADDMX function (1) and identifies the rack which the module belongs to.
Channel	STRING	Identifies the channel to work on; it contains rack, slot and channel numbers (r.s.c)
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Data Type	Significance
OperationSuccessful	BOOL	Operation terminated successfully
OperationActiv	BOOL	Operation active
FaultyOperation	BOOL	Operation terminated unsuccessfully
Status	WORD	Detected error identifier *(see page 323)*
NumberOfParamWords	INT	Number of parameter registers/words to read
Param	ANY	Parameter registers/words of the channel. Because, they are module-specific, refer to the module manual for a detailed description. A device DDT-type variable may be used

Chapter 32
SAVE_PARAM: Saving the adjustment parameters

Description

Description of the function

The SAVE_PARAM function is used when a modification is made to the adjustment parameters of a module or integrated interface, as a means of saving these new parameters - thus replacing the initial parameters - by performing an explicit exchange with the processor memory.

These parameters replace the initial values defined using the configuration editor (or by the most recent save).

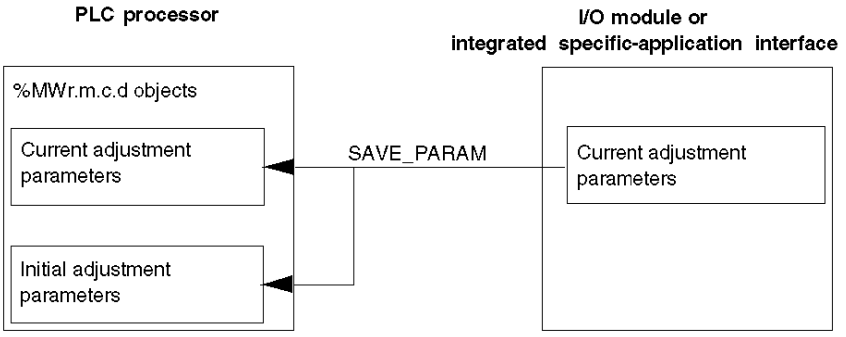
NOTE:

- For Premium this function only works if the application is stored in non-write protected RAM. On cold start-up, the current (unsaved) parameters are replaced with the initial parameters.
- For Modicon M340, the SAVE_PARAM does both current and initial parameter adjustment in memory application RAM (as in other PLCs) but on cold start (after application restore), the current parameter are replaced by the last adjusted initial values only if a save to memory card function (Backup Save or %S66 rising edge) has been done before.

The additional parameters EN and ENO may also be configured.

Operation

Operation flow diagram:

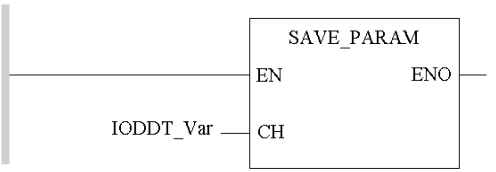


FBD representation

Representation:

LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
SAVE_PARAM
```

ST representation

Representation:

```
SAVE_PARAM(IODDT_Var);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	IODDT-type variable corresponding to the channel or module whose adjustment parameters the user wishes to save. **Example**: `IODDT_Var` of type `T_COUNT_ACQ`

Chapter 33
SAVE_PARAM_MX: Saving the Parameters in Local Rack

SAVE_PARAM_MX

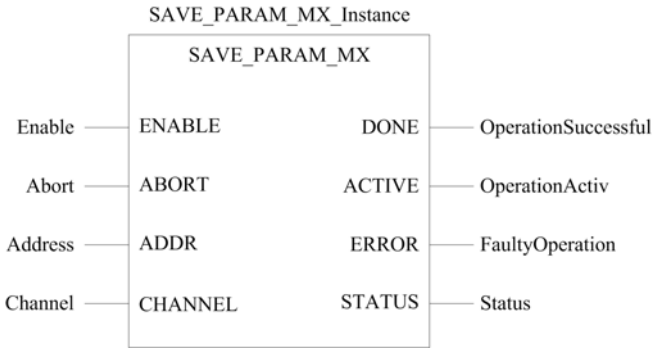
Description

The SAVE_PARAM_MX function is used to save the new parameters of a module declared in Device DDT in the local rack. It performs an explicit exchange *(see page 124)* with the Modicon M580 CPU memory.

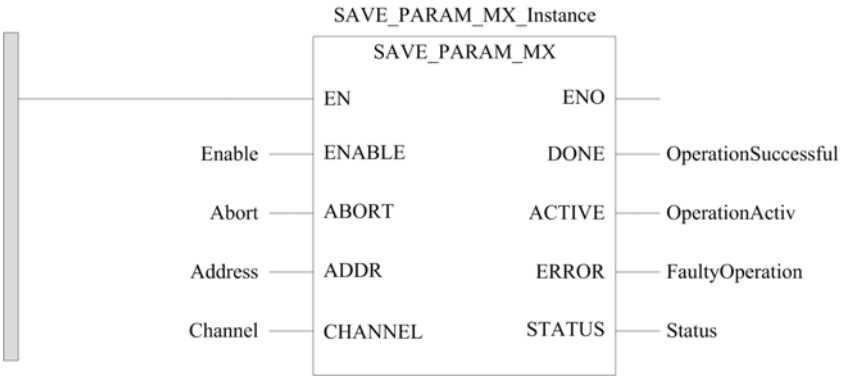
These parameters replace the initial values.

The additional parameters EN and ENO can also be configured.

FBD representation



LD representation



IL representation

Representation:

```
CAL SAVE_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, DONE=>OperationSuccessful, ACTIVE=>OperationActiv,
ERROR=>FaultyOperation, STATUS=>Status)
```

IL representation

Representation:

```
SAVE_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, DONE=>OperationSuccessful, ACTIVE=>OperationActiv,
ERROR=>FaultyOperation, STATUS=>Status);
```

Parameter Description

The following table describes the input parameters:

Parameter	Data Type	Significance
Enable	BOOL	Set to 1 to trigger the operation
Abort	BOOL	Set to 1 to abort the currently active operation
Address	ANY ARRAY OF INT	Array that contains the address result of ADDMX function (1) and identifies the rack which the module belongs to.
Channel	STRING	Identifies the channel to work on; it contains rack, slot and channel numbers (r.s.c)
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Data Type	Significance
OperationSuccessful	BOOL	Operation terminated successfully
OperationActiv	BOOL	Operation active
FaultyOperation	BOOL	Operation terminated unsuccessfully
Status	WORD	Detected error identifier *(see page 323)*

Chapter 34
TRF_RECIPE: Recipe transfer

Description

Description of the function

The TRF_RECIPE function is used to perform transfers and commands with the Cam and SERCOS specific-application modules.

For a SERCOS module (TSX CSY 84) *(see Premium and Atrium using Unity Pro, Motion control for SERCOS® motion, User manual)*, this service can be used to:

- read or write the speed controller parameters. TRF_RECIPE with the "Real axis" function,
- read or write the Cam profiles and launch the execution of special functions.

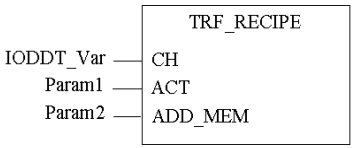
For an electronic cam module (TSX CCY 1128) *(see Premium and Atrium Using Unity Pro, Electronic Cam Module, User Manual)*, this service can be used to:

- transfer the contents of the current recipe to a memory field,
- transfer a recipe from a memory field to the %MW field containing the current recipe, then transfer it to the module.

The additional parameters EN and ENO may also be configured.

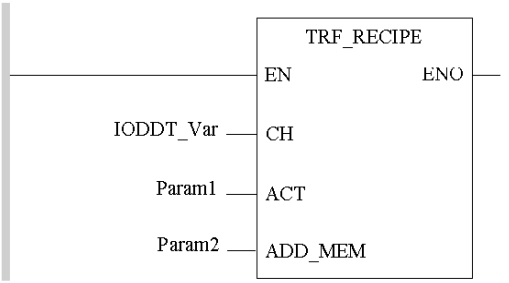
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
TRF_RECIPE Param1, Param2
```

ST representation

Representation:

```
TRF_RECIPE(IODDT_Var, Param1, Param2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	IODDT-type variable corresponding to the channel or module on which the function is to be performed. **Example**: `IODDT_Var` of type `T_CCY_MEASURE`
`Param1`	`INT`	The first function parameter; this depends on the module.
`Param2`	`INT`	The second function parameter; this depends on the module.

Chapter 35
WRITE_CMD: Updating the command parameters

Description

Description of the function

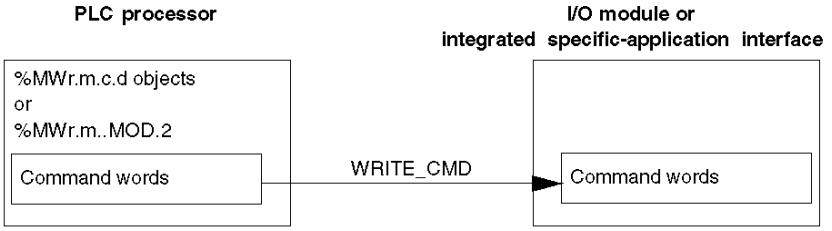
The WRITE_CMD function can be used to send a command to a module or integrated interface through the use of command words, by performing an explicit exchange.

NOTE: The command words are specific to each specific-application, and are described in the specific-application manuals.

The additional parameters EN and ENO may also be configured.

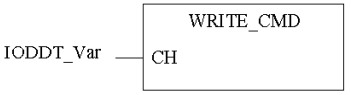
Operation

Operation flow diagram:



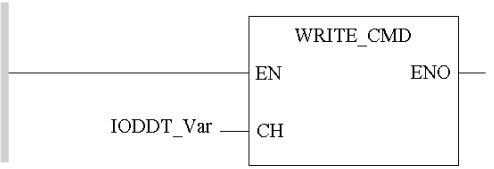
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
WRITE_CMD
```

ST representation

Representation:

```
WRITE_CMD(IODDT_Var);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	IODDT-type variable corresponding to the channel or module on which a command is to be performed. **Example**: `IODDT_Var` of type `T_COUNT_ACQ`

Chapter 36
WRITE_CMD_QX: Updating the Command Parameters on EIO Bus

Description

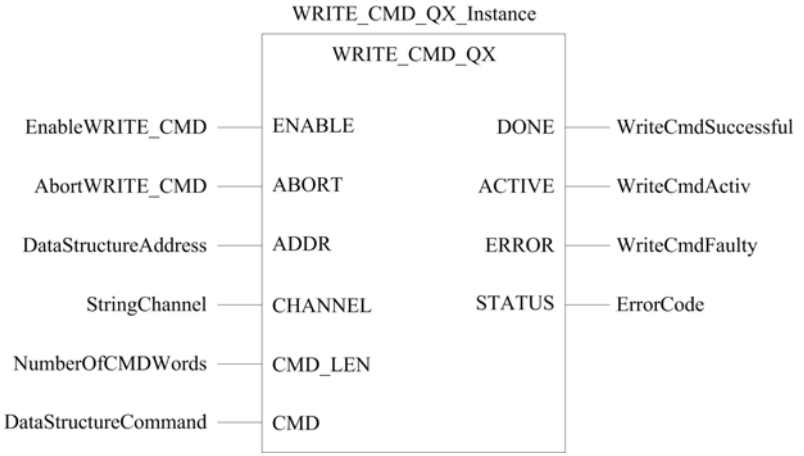
Description of the function

The WRITE_CMD_QX function block can be used to send a command to a Modicon M340 Ethernet I/O module through the use of command words, by performing an explicit exchange.

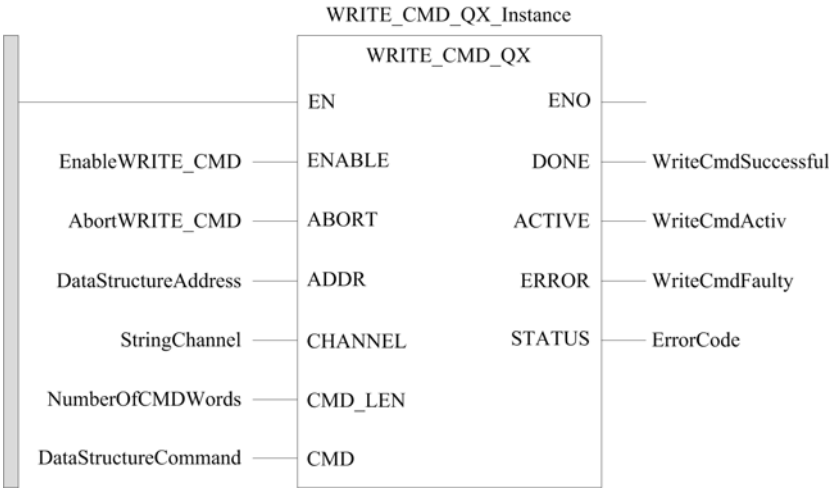
NOTE: The command words are specific to each specific application, and are described in the specific application manuals.

The additional parameters EN and ENO may also be configured.

FBD representation



LD representation



IL representation

Representation:

```
CAL WRITE_CMD_QX_Instance (ENABLE:=EnableWRITE_CMD,
ABORT:=AbortWRITE_CMD, ADDR:=DataStructureAddress,
CHANNEL:=StringChannel, CMD_LEN:=NumberOfCMDWords, CMD:=DataStructu-
reCommand, DONE=>WriteCmdSuccessful, ACTIVE=>WriteCmdActiv,
ERROR=>WriteCmdFaulty, STATUS=>ErrorCode)
```

ST representation

Representation:

```
WRITE_CMD_QX_Instance (ENABLE:=EnableWRITE_CMD, ABORT:=AbortWRITE_CMD,
ADDR:=DataStructureAddress, CHANNEL:=StringChannel, CMD_LEN:=NumberOfC-
MDWords, CMD:=DataStructureCommand, DONE=>WriteCmdSuccessful,
ACTIVE=>WriteCmdActiv, ERROR=>WriteCmdFaulty, STATUS=>ErrorCode);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
EnableWRITE_CMD	BOOL	Set to 1 to send a command.
AbortWRITE_CMD	BOOL	Set to 1 to abort the current operation.
DataStructureAddress	ANY_ARRAY_INT	Array containing the Modbus slave address, result of ADDMX *(see Unity Pro, Communication, Block Library)* function (1).
StringChannel	STRING	Identify the channel to work upon. Contains the rack, slot and channel numbers as follows: 'r.s.c'. If the channel is omitted, then the module server is addressed.
NumberOfCMDWords	INT	Number of command words to be sent. If set to 0, all command words will be sent.
DataStructureCommand	ANY	Channel command words. Command words to write. Instance of a DDT applicable to a specific IO channel command. The command word description is provided in each EIO device manual.
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Type	Comment
WriteCmdSuccessful	BOOL	Operation completed indication. Set to 1 when the execution of the operation is completed successfully.
WriteCmdActiv	BOOL	Operation in progress indication. Set to 1 when the execution of the operation is in progress.
WriteCmdFaulty	BOOL	Set to 1 if an error is detected by the function block.
ErrorCode	WORD	Code providing the detected error identification *(see page 323)*.

Chapter 37
WRITE_CMD_MX: Updating the Command Parameters on EIO Bus

Description

Description of the function

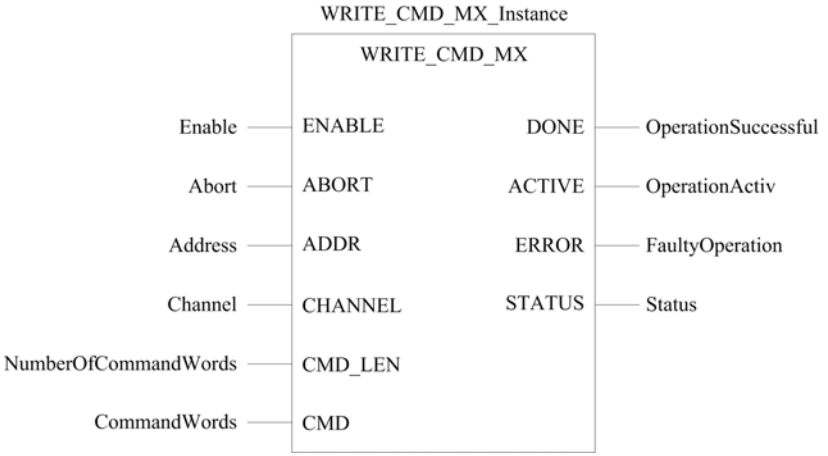
The `WRITE_CMD_MX` function block can be used to send a command to an (e)X80 local rack or Ethernet RIO module through the use of command words, by performing an explicit exchange.

NOTE: The command words are specific to each specific application, and are described in the specific application manuals.

The additional parameters `EN` and `ENO` may also be configured.

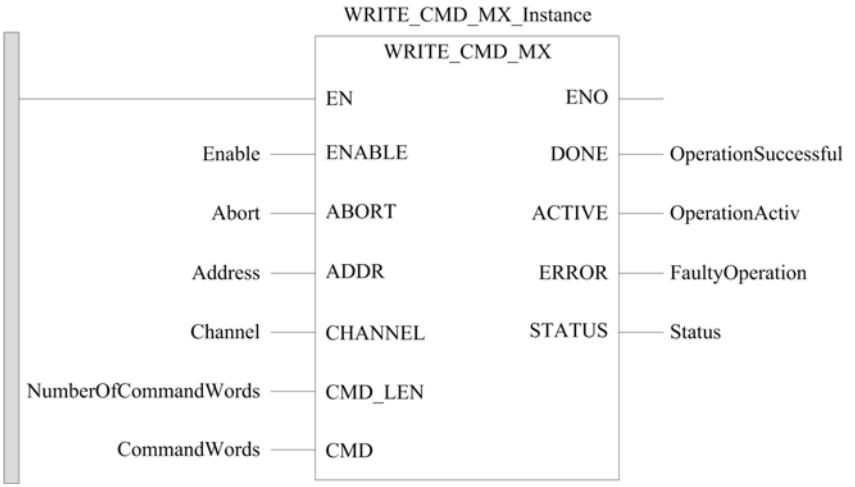
FBD representation

:



LD representation

:



IL representation

Representation:

```
CAL WRITE_CMD_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, CMD_LEN:=NumberOfCommandWords, CMD:=CommandWords,
DONE=>OperationSuccessful, ACTIVE=>OperationActiv, ERROR=>FaultyOp-
eration, STATUS=>Status)
```

ST representation

Representation:

```
WRITE_CMD_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, CMD_LEN:=NumberOfCommandWords, CMD:=CommandWords,
DONE=>OperationSuccessful, ACTIVE=>OperationActiv, ERROR=>FaultyOp-
eration, STATUS=>Status);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Enable	BOOL	Set to 1 to enable a operation.
Abort	BOOL	Set to 1 to abort the currently active operation.
Address	ANY_ARRAY_INT	Identifies the drop which the module belongs to. Array containing the Modbus slave address, result of ADDMX *(see Unity Pro, Communication, Block Library)* function (1).
Channel	STRING	Identifies the channel to work upon. Contains the rack, slot and channel numbers as follows: 'r.s.c'. If the channel is omitted, then the module server is addressed.
NumberOfCommandWords	INT	Number of command registers/words to be sent. If set to 0, all command registers will be sent.
CommandWords	ANY	Channel command words. Command words to write. A DDDT-type variable may be used. The command register/word description is provided in each EIO device manual.
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Type	Comment
OperationSuccessful	BOOL	Operation completed indication. Set to 1 when the execution of the operation is completed successfully.
OperationActiv	BOOL	Operation in progress indication. Set to 1 when the execution of the operation is in progress.
FaultyOperation	BOOL	Set to 1 if an error is detected by the function block.
Status	WORD	Code providing the detected error identification *(see page 323)*.

Chapter 38
WRITE_PARAM: Updating the adjustment parameters

Description

Description of the function

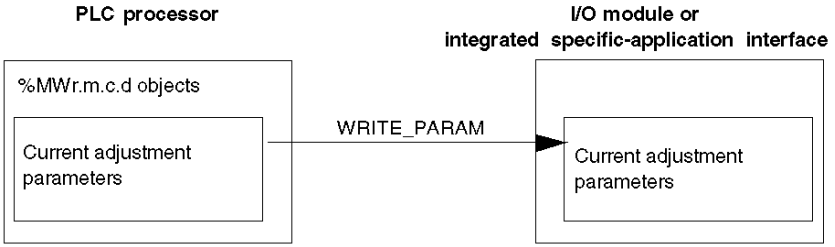
The WRITE_PARAM function is used to write the adjustment parameters of a module or integrated interface, by performing an explicit exchange with the processor memory.

NOTE: The advantage of this function is to be able to use a program to modify the adjustment values defined in the configuration.

The additional parameters EN and ENO may also be configured.

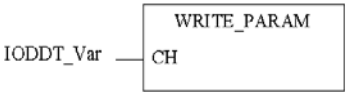
Operation

Operation flow diagram:



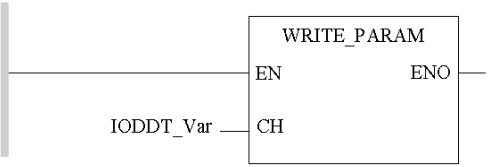
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
LD IODDT_Var
WRITE_PARAM
```

ST representation

Representation:

```
WRITE_PARAM(IODDT_Var);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`IODDT_Var`	`IODDT`	IODDT-type variable corresponding to the channel or module whose adjustment parameters the user wishes to write. **Example**: `IODDT_Var` of type `T_COUNT_ACQ`

Chapter 39
WRITE_PARAM_MX: Writing the Parameters on Local Rack

Description

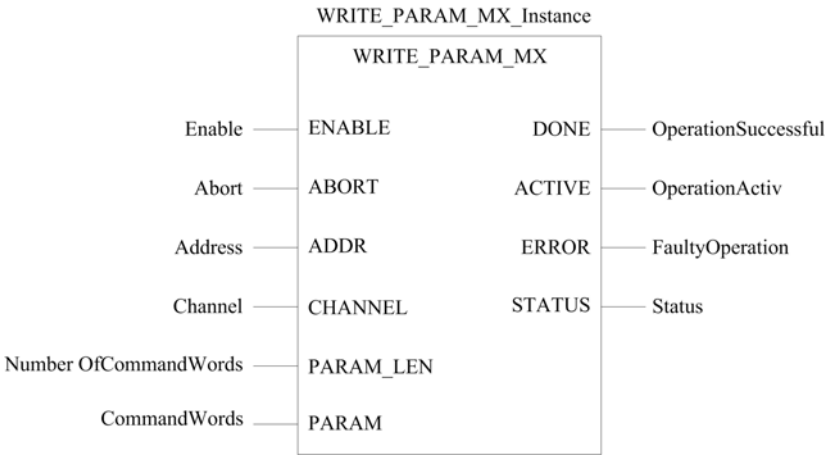
Description of the function

The `WRITE_PARAM_MX` function is used to write the parameters of a module declared in Device DDT in the local rack. It performs an explicit exchange *(see page 124)* with the Modicon M580 CPU memory.

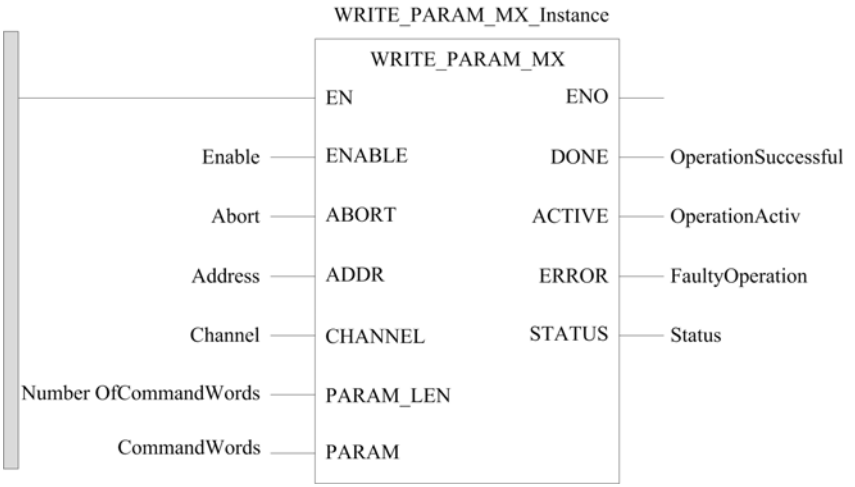
NOTE: The advantage of this function is to be able to use a program to modify the values defined in the configuration.

The additional parameters `EN` and `ENO` can also be configured.

FBD representation



LD representation



IL representation

Representation:

```
CAL WRITE_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, PARAM_LEN:=NumberOfCommandWords, PARAM:=CommandWords,
DONE=>OperationSuccessful, ACTIVE=>OperationActiv, ERROR=>FaultyOperation,
STATUS=>Status)
```

ST representation

Representation:

```
WRITE_PARAM_MX_Instance (ENABLE:=Enable, ABORT:=Abort, ADDR:=Address,
CHANNEL:=Channel, PARAM_LEN:=NumberOfCommandWords, PARAM:=CommandWords,
DONE=>OperationSuccessful, ACTIVE=>OperationActiv, ERROR=>FaultyOperation,
STATUS=>Status);
```

Parameter Description

The following table describes the input parameters:

Parameter	Data Type	Significance
Enable	BOOL	Set to 1 to trigger the operation
Abort	BOOL	Set to 1 to abort the currently active operation
Address	ANY ARRAY OF INT	Array that contains the address result of ADDMX function (1) and identifies the rack which the module belongs to.
Channel	STRING	Identifies the channel to work on; it contains rack, slot and channel numbers (r.s.c)
NumberOfCommandWords	INT	Number of parameter registers/words to write; 0: all registers will be sent.)
CommandWords	ANY	Parameter registers/words of the channel. Because they are module-specific, refer to the module manual for a detailed description. A Device DDT-type variable may be used
(1) To address a module in the local rack, enter 0.0.10 (CPU main server address).		

The following table describes the output parameters:

Parameter	Data Type	Significance
OperationSuccessful	BOOL	Operation terminated successfully
OperationActiv	BOOL	Operation active
FaultyOperation	BOOL	Operation terminated unsuccessfully
Status	WORD	Detected error identifier *(see page 323)*

Part V
Immediate I/O

Overview

This section describes the elementary functions and elementary function blocks from the family `Immediate I/O`.

NOTE: The immediate I/O function blocks can not be used in a Hot Standby configuration.

What Is in This Part?

This part contains the following chapters:

Chapter 40
IMIO_IN: Immediate I/O module input

Introduction

This chapter describes the `IMIO_IN` block.

What Is in This Chapter?

This chapter contains the following topics:

Description

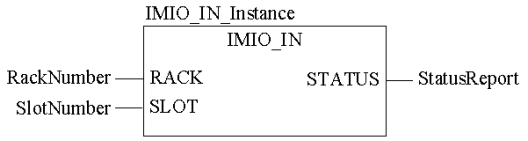
Function description

This function block reads I/O module signals immediately during processing. The input module must be in the local rack of the PLC.

EN and ENO can be configured as additional parameters.

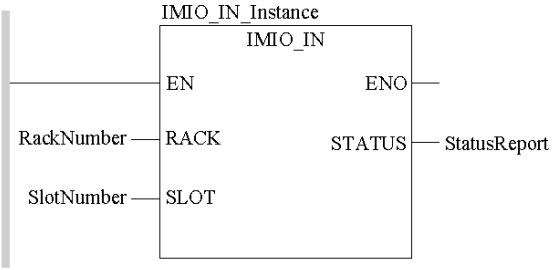
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL IMIO_IN_Instance (RACK:=RackNumber, SLOT:=SlotNumber,
    STATUS=>StatusReport)
```

Representation in ST

Representation:

```
IMIO_IN_Instance (RACK:=RackNumber, SLOT:=SlotNumber,
    STATUS=>StatusReport) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
RACK	INT	Rack number (Quantum: 1)
SLOT	INT	Slot number (Quantum: 1...16)

Description of the output parameters:

Parameter	Data type	Meaning
STATUS	WORD	Status report

Runtime error

The ENO parameter can be used for error display:

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

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 rack 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

Chapter 41
IMIO_OUT: Immediate I/O module output

Introduction

This chapter describes the IMIO_OUT block.

What Is in This Chapter?

This chapter contains the following topics:

Description

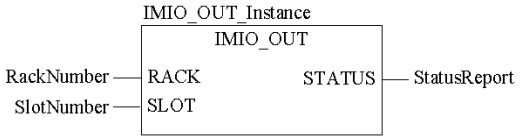
Function description

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 configured as additional parameters.

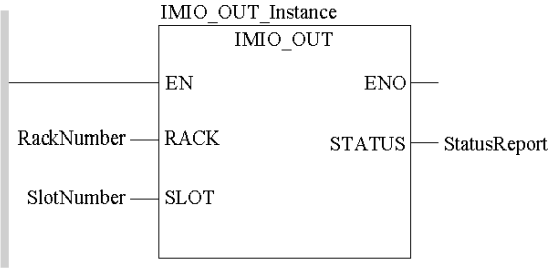
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL IMIO_OUT_Instance (RACK:=RackNumber, SLOT:=SlotNumber,
    STATUS=>StatusReport)
```

Representation in ST

Representation:

```
IMIO_OUT_Instance (RACK:=RackNumber, SLOT:=SlotNumber,
    STATUS=>StatusReport) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
RACK	INT	Rack number (Quantum: 1)
SLOT	INT	Slot number (Quantum: 1...16)

Description of the output parameters:

Parameter	Data type	Meaning
STATUS	WORD	Status report

Runtime error

The ENO parameter can be used for error display:

ENO	Meaning
1	Operation OK (STATUS equals "0")
0	Operation faulty (STATUS equals "0")

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 rack 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

Chapter 42
IU_ERIO: Quantum Ethernet I/O Immediate Access to an Ethernet Remote I/O Drop

Description

Function Description

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use the IU_ERIO function block in Quantum Hot Standby installations.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

An IU_ERIO function block updates Ethernet remote I/O drop input and output modules with an optimal response time. The Ethernet remote I/O drop inputs and outputs are updated during the MAST task.

Call this function block in a MAST task. It can be called more than once in a task.

NOTE: To maintain system performance, we recommend that you use no more than 10 executions of the IU_ERIO block during a single MAST task.

The input and output modules are physically located on an Ethernet remote I/O drop and declared in the Ethernet configuration.

EN and ENO can be configured as additional parameters.

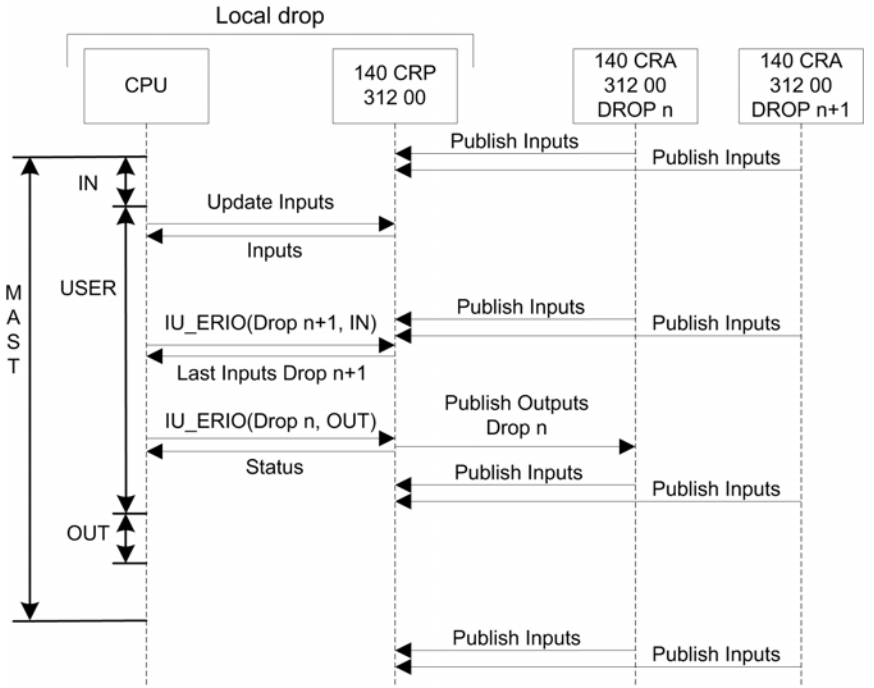
NOTE: Refer to the *Quantum EIO System Planning Guide* for calculating the ART when your application does not use an IU_ERIO function block.

IU_ERIO Mechanism

The Ethernet remote I/O drop input values are read in the 140 CRP 312 00 module with an optimal response time.

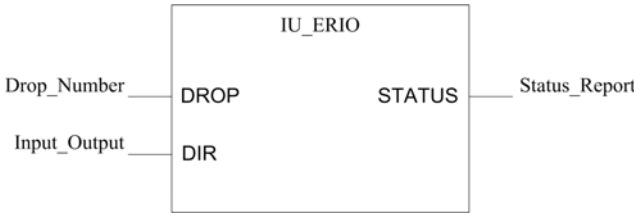
The input values read in the 140 CRP 312 00 module represent the latest values sent in an asynchronous way by the adapter module in each drop. The maximum time shift between values read in the 140 CRP 312 00 and actual input values depends on the adapter's publishing frequency (subscribe field **CRA-> RPI**) *(see Quantum EIO, Remote I/O Modules, Installation and Configuration Guide)*.

The following diagram represents the I/O exchanges between a CPU and the Ethernet remote I/O drops:



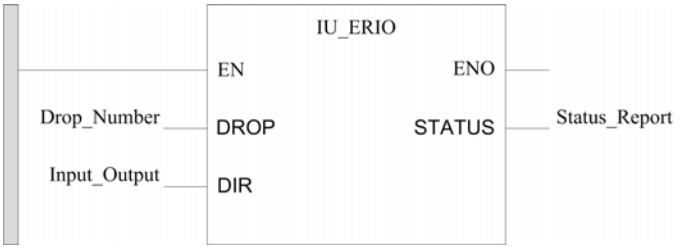
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

```
CAL IU_ERIO(DROP:=>Drop_Number, DIR:=>Input_Output, STATUS=>Status_Repo
rt);
```

Representation in ST

```
IU_ERIO(DROP:=>Drop_Number, DIR:=>Input_Output, STATUS=>Status_Report);
```

Parameter Description

Input parameters:

Parameter	Data type	Meaning
`Drop`	`INT`	Drop number (1...31) Drop number: ● 1: Drop 1 ● 2: Drop 2 ● ... ● 31: Drop 31
`Dir`	`BOOL`	Data direction: ● 0 = Outputs. The output values are sent immediately to the 140 CRP 312 00 module. ● 1 = Inputs. The input values are read immediately from the 140 CRP 312 00 module.

Output parameter:

Parameter	Data type	Meaning
Status	WORD	Status report from the 140 CRP 312 00 module: ● 0002 hex: Invalid drop number ● 0003 hex: Ethernet remote I/O drop is not configured ● 0004 hex: Ethernet remote I/O drop is not connected ● 0005 hex: Retry number is exceeded ● 0007 hex: An error is detected on the 140 CRP 312 00 module ● 0008 hex: Operation was not completed before time out ● 0009 hex: 140 CRP 312 00 module is not present on the local drop ● 000B hex: Operation OK

NOTE: If no connection is opened with the Ethernet remote I/O drop, a detected communication error (system words %SW172 to %SW173) is returned.

Part VI
Quantum I/O Configuration

Overview

This section describes the elementary functions and elementary function blocks from the family `Quantum I/O Configuration`.

What Is in This Part?

This part contains the following chapters:

Chapter 43
ACI030: Configuring the Quantum module ACI 030 00

Description

Function description

The function block is used to edit the configuration data of an ACI 030 00 Quantum module for subsequent use by the scaling EFBs.

This module has 8 unipolar input channels for mixed voltage and current processing.

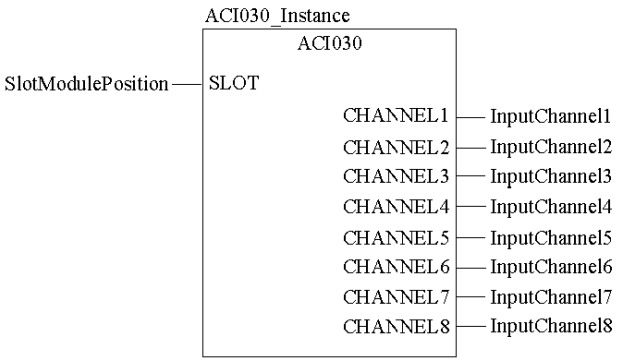
For the configuration of an ACI 030 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

The analog values can be further processed in Scaling sections using the I_NORM, I_RAW and I_SCALE function blocks.

EN and ENO can be configured as additional parameters.

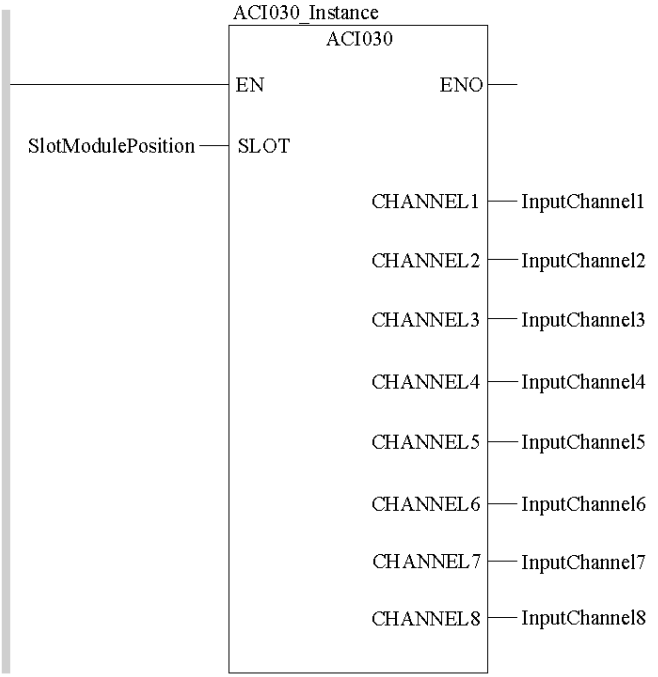
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ACI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8)
```

Representation in ST

Representation:

```
ACI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL IN	Channel 1
CHANNEL2	ANL IN	Channel 2
CHANNEL3	ANL IN	Channel 3
CHANNEL4	ANL IN	Channel 4
CHANNEL5	ANL IN	Channel 5
CHANNEL6	ANL IN	Channel 6
CHANNEL7	ANL IN	Channel 7
CHANNEL8	ANL IN	Channel 8

Runtime error

If no ACI 030 00 module has been configured for the specified SLOT input, an error message is returned.

The status information "Open circuit or under voltage on channel" can be collected via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 44
ACI040: Configuring the Quantum module ACI 040 00

Description

Function description

The function block is used to edit the configuration data of an ACI 040 00 Quantum module for subsequent use by the scaling EFBs.

The module has 16 channels which, according to requirement, can be used as differential or single inputs for the processing of current. When processing current the ranges are 020 mA, 025 mA and 420 mA.

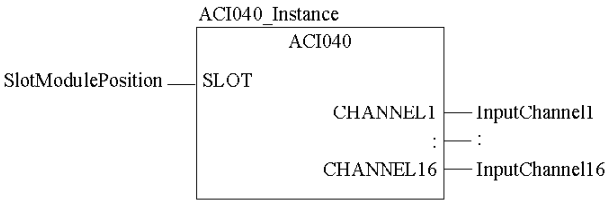
For the configuration of an ACI 040 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references specified in the I/O map are automatically assigned internally to individual channels. The channels can only be occupied by Unlocated variables.

The analog values can be further processed in Scaling Sections using the I_NORM, I_PHYS, I_RAW and I_SCALE functions.

EN and ENO can be configured as additional parameters.

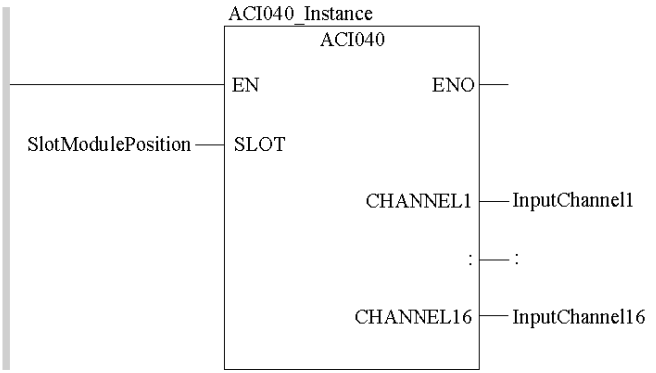
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ACI040_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8,
    CHANNEL9=>InputChannel9, CHANNEL10=>InputChannel10,
    CHANNEL11=>InputChannel11, CHANNEL12=>InputChannel12,
    CHANNEL13=>InputChannel13, CHANNEL14=>InputChannel14,
    CHANNEL15=>InputChannel15, CHANNEL16=>InputChannel16)
```

Representation in ST

Representation:

```
ACI040_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8,
    CHANNEL9=>InputChannel9, CHANNEL10=>InputChannel10,
    CHANNEL11=>InputChannel11, CHANNEL12=>InputChannel12,
    CHANNEL13=>InputChannel13, CHANNEL14=>InputChannel14,
    CHANNEL15=>InputChannel15, CHANNEL16=>InputChannel16) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_IN	Channel 1
:	:	:
CHANNEL16	ANL IN	Channel 16

Runtime error

If no ACI 040 00 module has been configured for the specified SLOT input, an error message is returned.

In the "4 to 20 mA" mode, the "open circuit on channel" status information is available. It can be requested via the %IW references of the module (%IWx+16) defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration,* *page 319*.

Chapter 45
ACO020: Configuring the Quantum module ACO 020 00

Description

Function description

The function block is used to edit the configuration data of an ACO 020 00 Quantum module for subsequent use by the scaling EFBs.

This module has 4 output channels for the processing of current in the range of 4 ... 20 mA.

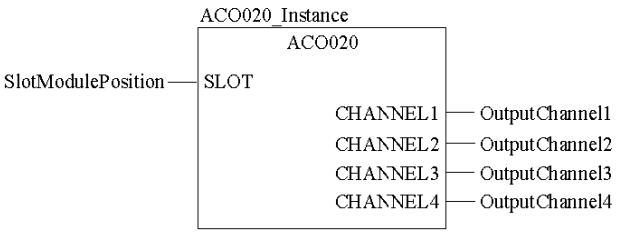
For the configuration of an ACO 020 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %MW references specified in the I/O map and the status information (if configured) are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

The analog values can be further processed in Scaling sections using the O_NORM, O_RAW and O_SCALE procedures.

EN and ENO can be configured as additional parameters.

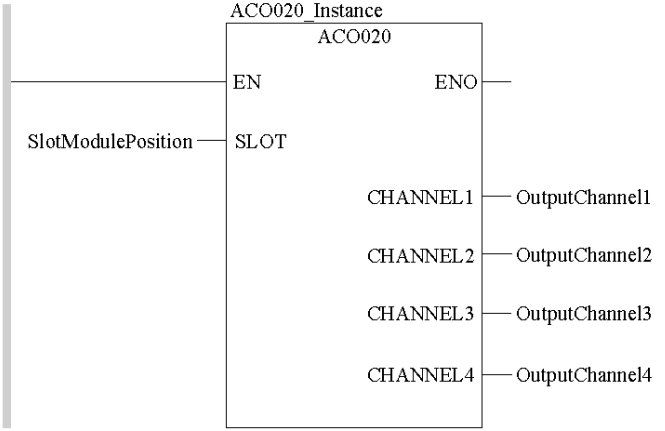
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ACO020_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4)
```

Representation in ST

Representation:

```
ACO020_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL IN	Channel 1
CHANNEL2	ANL_OUT	Channel 2
CHANNEL3	ANL_OUT	Channel 3
CHANNEL4	ANL_OUT	Channel 4

Runtime error

If no ACO 020 00 module has been configured for the specified SL0T input, an error message is returned.

The status information "Open circuit on channel" can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 46
ACO130: Configuring the Quantum module ACO 130 00

Description

Function description

The function block is used to edit the configuration data of an ACO 130 00 Quantum module for subsequent use by the scaling EFBs.

This module has 8 output channels for controlling and supervising the currents in the ranges 0 to 20 mA, 0 to 25 mA and 4 to 20 mA.

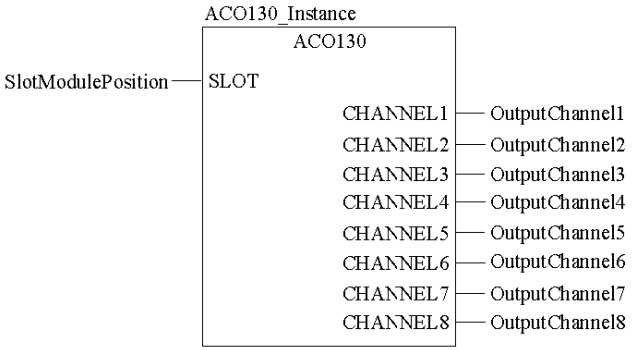
For the configuration of an ACO 130 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %MW references specified in the I/O map are automatically assigned internally to individual channels. The channels can only be occupied by Unlocated variables.

The analog values can be further processed in Scaling Sections using the O_NORM, O_PHYS, O_RAW and O_SCALE procedures.

EN and ENO can be configured as additional parameters.

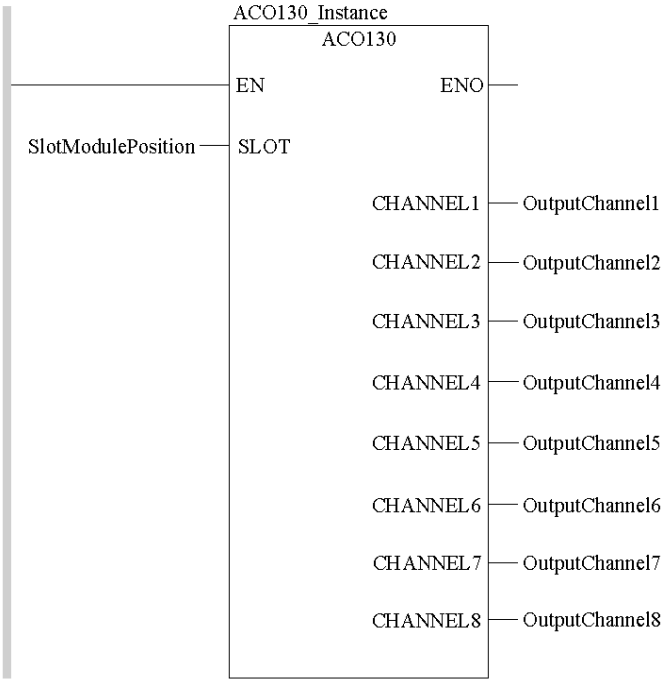
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ACO130_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4,
    CHANNEL5=>OutputChannel5, CHANNEL6=>OutputChannel6,
    CHANNEL7=>OutputChannel7, CHANNEL8=>OutputChannel8)
```

Representation in ST

Representation:

```
ACO130_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4,
    CHANNEL5=>OutputChannel5, CHANNEL6=>OutputChannel6,
    CHANNEL7=>OutputChannel7, CHANNEL8=>OutputChannel8) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_OUT ;	Channel 1
:	:	:
CHANNEL8	ANL_OUT	Channel 8

Runtime error

If no ACO 130 00 module has been configured for the specified SL0T input, an error message is returned.

In the "4 to 20 mA" mode, the "open circuit on channel" status information is available. It can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 47
AII330: Configuring the Quantum module AII 330 00

Description

Function description

The function block is used to edit the configuration data of an AII 330 00 Quantum module for subsequent use by the scaling EFBs.

The module has 8 intrinsically safe channels and can be used either as a resistor temperature sensor (RTD) or as a thermoelement/millivolt input module.

For the configuration of an AII 330 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references specified in the I/O map are automatically assigned internally to individual channels. The channels can only be occupied by Unlocated variables.

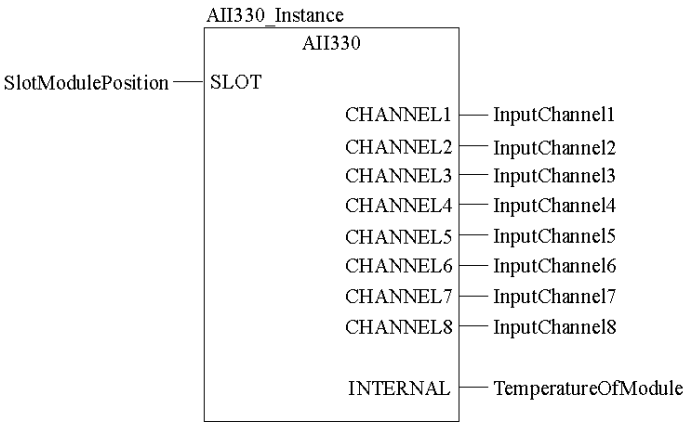
The analog values can be further processed in Scaling sections using the I_NORM, I_NORM_WARN, I_PHYS, I_PHYS_WARN, I_RAW, I_SCALE and I_SCALE_WARN function blocks.

NOTE: When setting parameters for physical or temperature values, I_SCALE and I_SCALE_WARN can not be used.

EN and ENO can be configured as additional parameters.

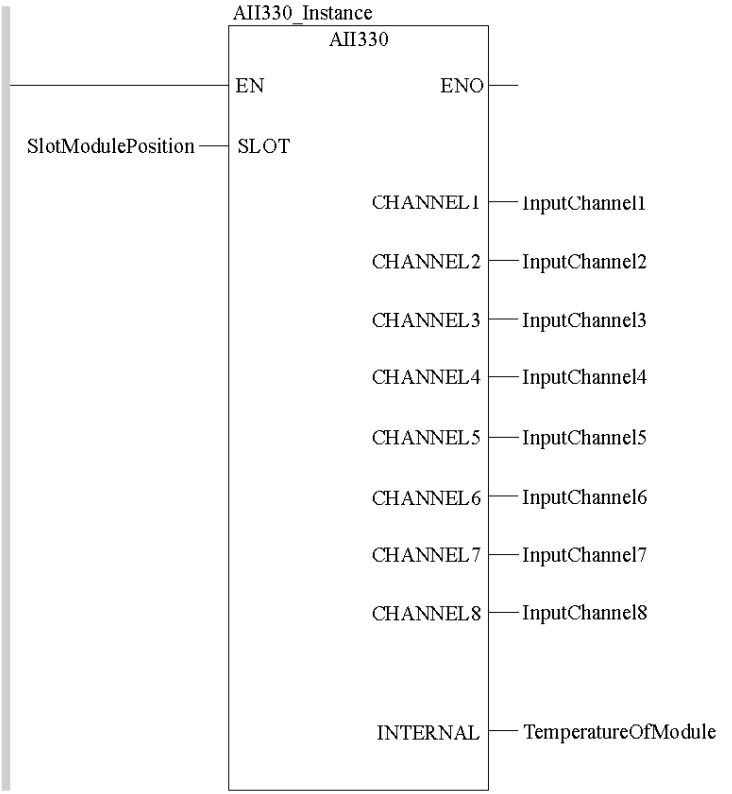
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AII330_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8,
    INTERNAL=>TemperatureOfModule)
```

Representation in ST

Representation:

```
AII330_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8,
    INTERNAL=>TemperatureOfModule) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_IN	Channel 1
CHANNEL2	ANL IN	Channel 2
CHANNEL3	ANL IN	Channel 3
CHANNEL4	ANL IN	Channel 4
CHANNEL5	ANL IN	Channel 5
CHANNEL6	ANL IN	Channel 6
CHANNEL7	ANL IN	Channel 7
CHANNEL8	ANL IN	Channel 8
INTERNAL	ANL IN	Temperature of module

Runtime error

If no AII 330 00 module has been configured for the specified SLOT input, an error message is returned.

The range warning for the channels can be evaluated using the function block I_NORM_WARN, I_SCALE_WARN or I_PHYS_WARN.

The status information "Open circuit or range violation on channel" can be requested via the %IW references (%IWx+8; module status register) defined in the I/O map or via the status register defined in the I/O map. (The information in the status register is a copy of the %IWx+8 module status register (High-Byte)).

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 48
AII33010: Configuring the Quantum module AII 330 10

Description

Function description

The function block is used to edit the configuration data of an AII 330 10 Quantum module for subsequent use by the scaling EFBs.

The module has 8 unipolar intrinsically safe channels. The following ranges can be selected: 0...20 mA , 0...25 mA and 4...20 mA.

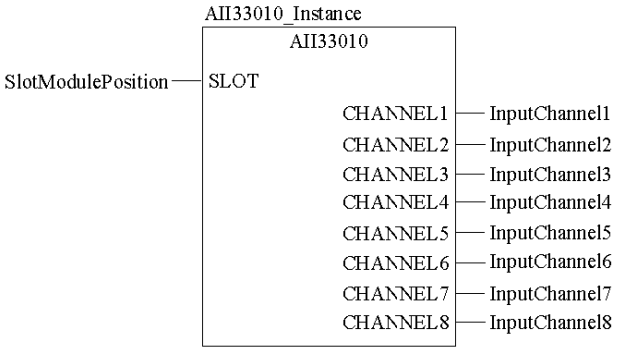
For the configuration of an AII 330 10 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references specified in the I/O map are automatically assigned internally to individual channels. The channels can only be occupied by Unlocated variables.

The analog values can be further processed in Scaling Sections using the I_NORM, I_PHYS, I_RAW and I_SCALE functions.

EN and ENO can be configured as additional parameters.

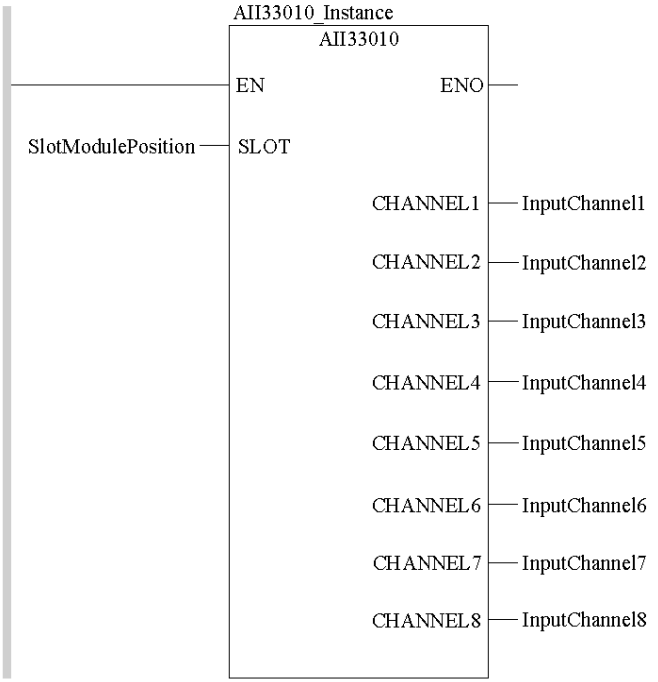
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AII33010_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8)
```

Representation in ST

Representation:

```
AII33010_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_IN	Channel 1
CHANNEL2	ANL IN	Channel 2
CHANNEL3	ANL IN	Channel 3
CHANNEL4	ANL IN	Channel 4
CHANNEL5	ANL IN	Channel 5
CHANNEL6	ANL IN	Channel 6
CHANNEL7	ANL IN	Channel 7
CHANNEL8	ANL IN	Channel 8

Runtime error

If no ACI 330 10 module has been configured for the specified SL0T input, an error message is returned.

In the "4 to 20 mA" mode, the "open circuit on channel" status information is available. It can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 49
AIO330: Configuring the Quantum module AIO 330 00

Description

Function description

The function block is used to edit the configuration data of an AIO 330 00 Quantum module for subsequent use by the scaling EFBs.

This module has 8 intrinsically safe symmetrical output channels for controlling and supervising the currents in the ranges 0 to 20 mA, 0 to 25 mA and 4 to 20 mA.

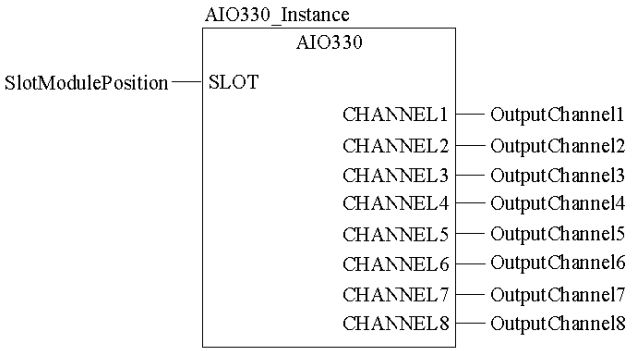
For the configuration of an AIO 330 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %MW references specified in the I/O map are automatically assigned internally to individual channels. The channels can only be occupied by Unlocated variables.

The analog values can be further processed in Scaling Sections using the O_NORM, O_PHYS, O_RAW and O_SCALE procedures.

EN and ENO can be configured as additional parameters.

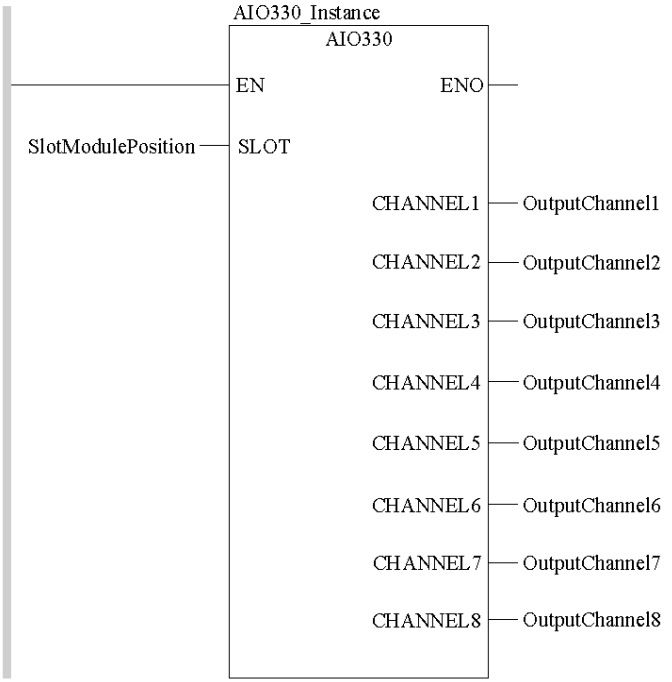
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AIO330_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4,
    CHANNEL5=>OutputChannel5, CHANNEL6=>OutputChannel6,
    CHANNEL7=>OutputChannel7, CHANNEL8=>OutputChannel8)
```

Representation in ST

Representation:

```
AIO330_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4,
    CHANNEL5=>OutputChannel5, CHANNEL6=>OutputChannel6,
    CHANNEL7=>OutputChannel7, CHANNEL8=>OutputChannel8) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_OUT ;	Channel 1
CHANNEL2	ANL_OUT	Channel 2
CHANNEL3	ANL_OUT	Channel 3
CHANNEL4	ANL_OUT	Channel 4
CHANNEL5	ANL_OUT	Channel 5
CHANNEL6	ANL_OUT	Channel 6
CHANNEL7	ANL_OUT	Channel 7
CHANNEL8	ANL_OUT	Channel 8

Runtime error

If no AIO 330 00 module has been configured for the specified SLOT input, an error message is returned.

The status message "Open circuit or range violation on channel" can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 50
AMM090: Configuring the Quantum module AMM 090 00

Description

Function description

The function block is used to edit the configuration data of an AMM 090 00 Quantum module for subsequent use by the scaling EFBs.

This module has 4 bipolar input channels for mixed voltage and current processing. The module also has 2 current output channels.

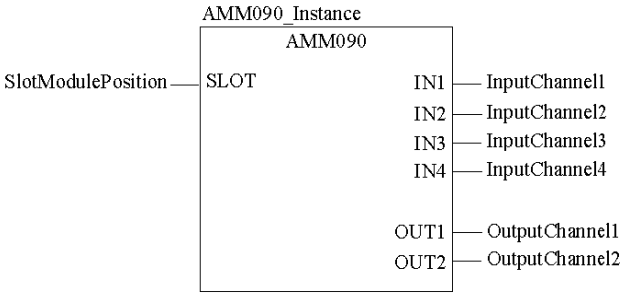
For the configuration of an AMM 090 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references and %MW references references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

The analog values can be further processed in Scalings Sections using the function blocks I_NORM, I_NORM_WARN, I_PHYS, I_PHYS_WARN, I_RAW, I_SCALE, I_SCALE_WARN for inputs and O_NORM, O_SCALE, O_PHYSand O_RAW for the outputs.

EN and ENO can be configured as additional parameters.

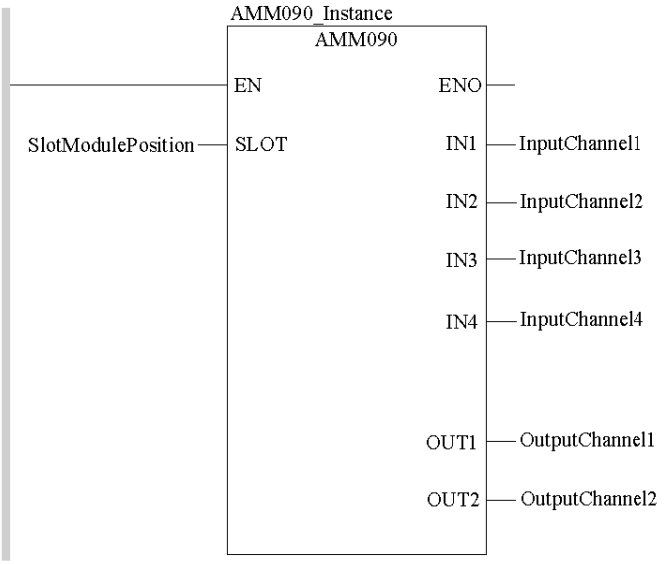
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AMM090_Instance (SLOT:=SlotModulePosition,
    IN1=>InputChannel1, IN2=>InputChannel2,
    IN3=>InputChannel3, IN4=>InputChannel4,
    OUT1=>OutputChannel1, OUT2=>OutputChannel2)
```

Representation in ST

Representation:

```
AMM090_Instance (SLOT:=SlotModulePosition,
    IN1=>InputChannel1, IN2=>InputChannel2,
    IN3=>InputChannel3, IN4=>InputChannel4,
    OUT1=>OutputChannel1, OUT2=>OutputChannel2) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
IN1	ANL_IN	Input channel 1
IN2	ANL IN	Input channel 2
IN3	ANL IN	Input channel 3
IN4	ANL IN	Input channel 4
OUT1	ANL_OUT	Output channel 1
OUT2	ANL_OUT	Output channel 2

Runtime error

If no AMM 090 00 module has been configured for the specified SLOT input, an error message is returned.

The range warning for the input channels can be evaluated using the I_NORM_WARN, I_PHYS_WARN or I_SCALE_WARN function blocks.

The status message "Open circuit or range violation on channel" can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 51
ARI030: Configuring the Quantum module ARI 030 10

Description

Function description

The function block is used to edit the configuration data of an ARI 030 10 Quantum module for subsequent use by the scaling EFBs.

This module has 8 resistor temperature sensor input channels (RTD) for the processing of four-wire RTD sensors.

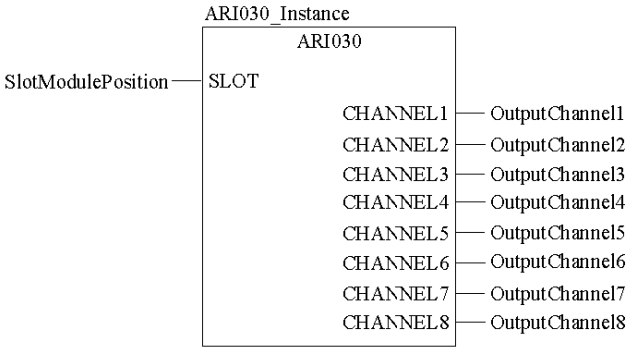
For the configuration of an ARI 030 10 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

The analog values can be further processed in Scaling sections using the I_NORM, I_NORM_WARN, I_PHYS, I_PHYS_WARN and I_RAW function blocks.

EN and ENO can be configured as additional parameters.

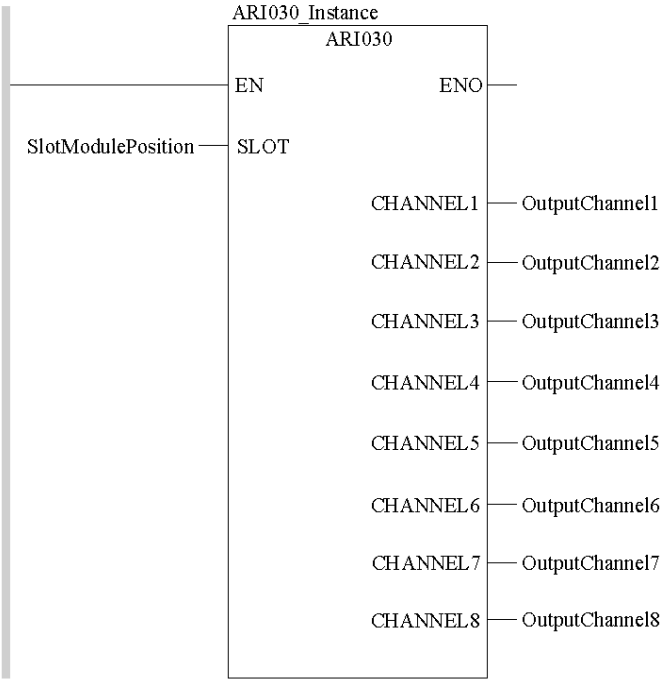
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ARI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8)
```

Representation in ST

Representation:

```
ARI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_IN	Channel 1
CHANNEL2	ANL IN	Channel 2
CHANNEL3	ANL IN	Channel 3
CHANNEL4	ANL IN	Channel 4
CHANNEL5	ANL IN	Channel 5
CHANNEL6	ANL IN	Channel 6
CHANNEL7	ANL IN	Channel 7
CHANNEL8	ANL IN	Channel 8

Runtime error

If no ACI 030 10 module has been configured for the specified SLOT input, an error message is returned.

The range warning for the channels can be evaluated through the I_NORM_WARN or I_PHYS_WARN function block.

The status message "Open circuit or range violation on channel" can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 52
ATI030: Configuring the Quantum module ATI 030 00

Description

Function description

The function block is used to edit the configuration data of an ATI 030 00 Quantum module for subsequent use by the scaling EFBs.

This module has 8 thermocouple input channels.

For the configuration of an ATI 030 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

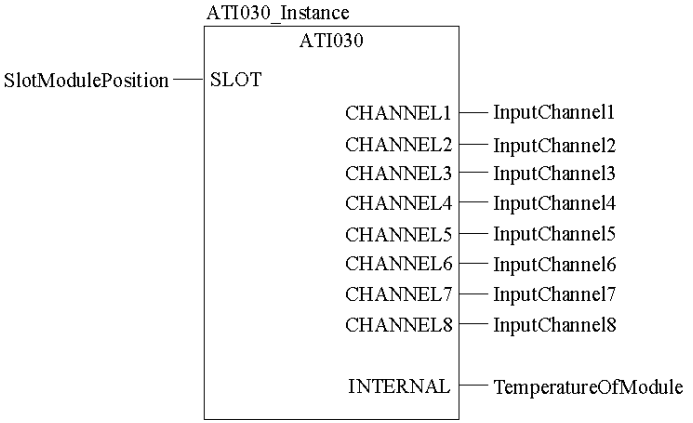
The analog values can be further processed in Scaling sections using the I_NORM, I_NORM_WARN, I_RAW, I_PHYS and I_PHYS_WARN function blocks.

NOTE: The I_NORM function block must be preferred to the I_PHYS function block.

EN and ENO can be configured as additional parameters.

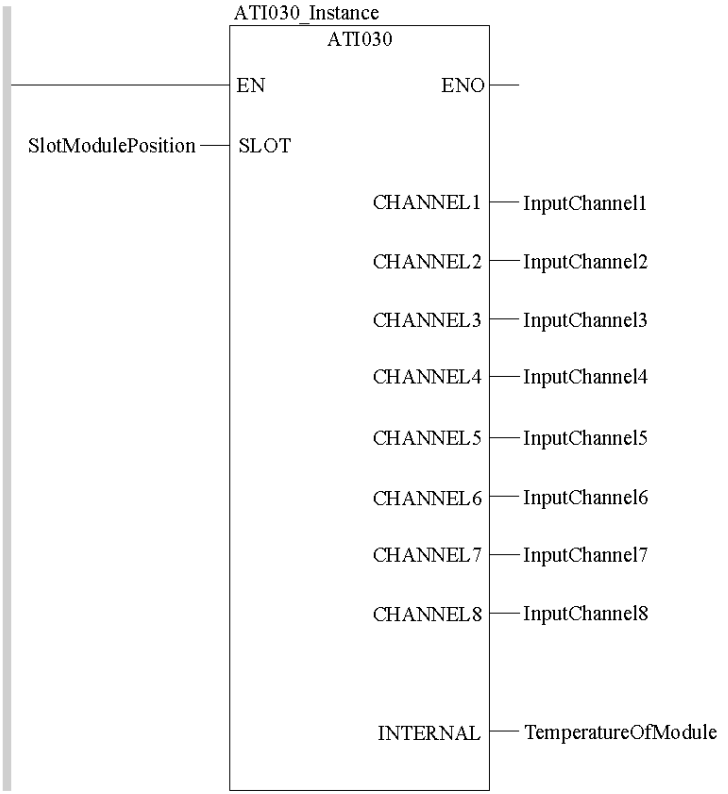
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ATI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8,
    INTERNAL=>TemperatureOfModule)
```

Representation in ST

Representation:

```
ATI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8,
    INTERNAL=>TemperatureOfModule) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_IN	Channel 1
CHANNEL2	ANL IN	Channel 2
CHANNEL3	ANL IN	Channel 3
CHANNEL4	ANL IN	Channel 4
CHANNEL5	ANL IN	Channel 5
CHANNEL6	ANL IN	Channel 6
CHANNEL7	ANL IN	Channel 7
CHANNEL8	ANL IN	Channel 8
INTERNAL	ANL IN	Temperature of module

Runtime error

If no ATI 030 00 module has been configured for the specified SLOT input, an error message is returned.

The range warning for the channels can be evaluated through the I_NORM_WARN or I_PHYS_WARN function block.

The status information "Range violation on channel" can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 53
AVI030: Configuring the Quantum module AVI 030 00

Description

Function description

The function block is used to edit the configuration data of an AVI 030 00 Quantum module for subsequent use by the scaling EFBs.

This module has 8 bipolar input channels for mixed voltage and current processing.

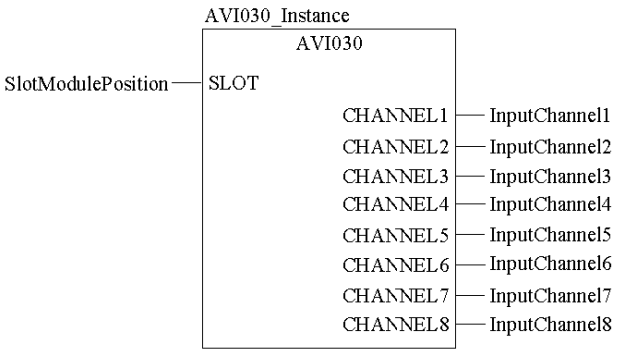
For the configuration of an AVI 030 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %IW references references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

The analog values can be further processed in Scaling sections using the I_NORM, I_NORM_WARN, I_PHYS, I_PHYS_WARN, I_RAW, I_SCALE and I_SCALE_WARN function blocks.

EN and ENO can be configured as additional parameters.

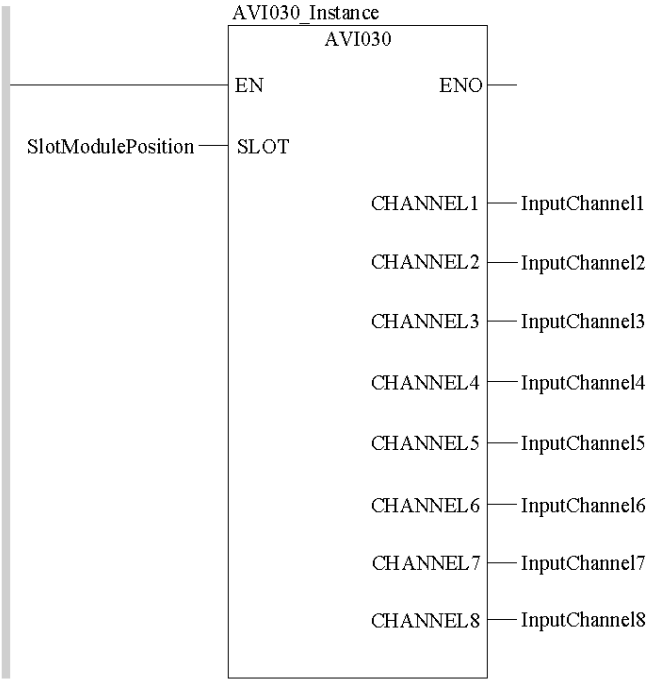
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AVI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8)
```

Representation in ST

Representation:

```
AVI030_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>InputChannel1, CHANNEL2=>InputChannel2,
    CHANNEL3=>InputChannel3, CHANNEL4=>InputChannel4,
    CHANNEL5=>InputChannel5, CHANNEL6=>InputChannel6,
    CHANNEL7=>InputChannel7, CHANNEL8=>InputChannel8) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_IN	Channel 1
CHANNEL2	ANL IN	Channel 2
CHANNEL3	ANL IN	Channel 3
CHANNEL4	ANL IN	Channel 4
CHANNEL5	ANL IN	Channel 5
CHANNEL6	ANL IN	Channel 6
CHANNEL7	ANL IN	Channel 7
CHANNEL8	ANL IN	Channel 8

Runtime error

If no AVI 030 00 module has been configured for the specified SLOT input, an error message is returned.

The range warning for the channels can be evaluated using the I_NORM_WARN, I_PHYS_WARN or I_SCALE_WARN function blocks.

The status message "Open circuit or range violation on channel" can be requested via the status register defined in the I/O map.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 54
AVO020: Configuring the Quantum module AVO 020 00

Description

Function description

The function block is used to edit the configuration data of an AVO 020 00 Quantum module for subsequent use by the scaling EFBs.

This module has 4 voltage output channels with mixed modes and levels.

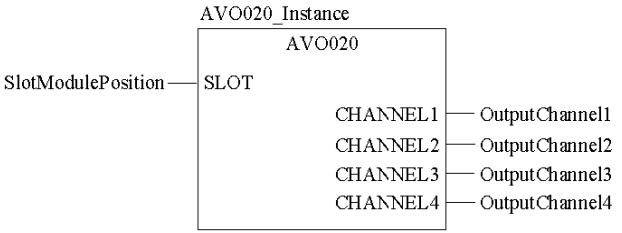
For the configuration of an AVO 020 00 the function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM or DROP function block. The %MW references specified in the I/O map are automatically assigned internally to the individual channels and can therefore only be occupied by Unlocated variables.

The analog values can be further processed in Scaling sections using the O_NORM, O_RAW and O_SCALE procedures.

EN and ENO can be configured as additional parameters.

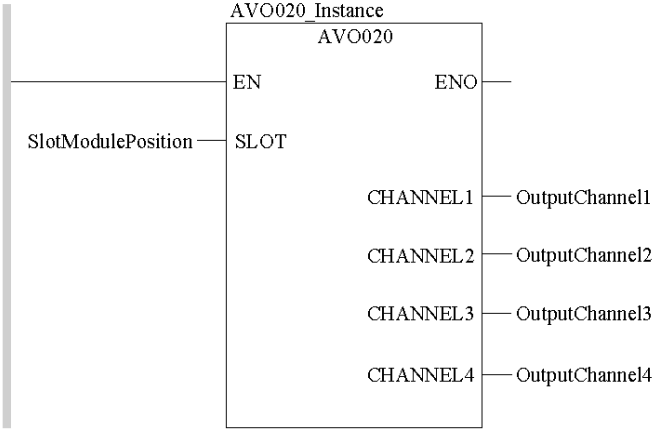
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AVO020_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4)
```

Representation in ST

Representation:

```
AVO020_Instance (SLOT:=SlotModulePosition,
    CHANNEL1=>OutputChannel1, CHANNEL2=>OutputChannel2,
    CHANNEL3=>OutputChannel3, CHANNEL4=>OutputChannel4) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Module slot

Description of output parameters:

Parameter	Data type	Meaning
CHANNEL1	ANL_OUT	Channel 1
CHANNEL2	ANL_OUT	Channel 2
CHANNEL3	ANL_OUT	Channel 3
CHANNEL4	ANL_OUT	Channel 4

Runtime error

If no ATI 020 00 module has been configured for the specified SLOT input, an error message is returned.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 55
DROP: Configuring a I/O Station Rack

Description

Function description

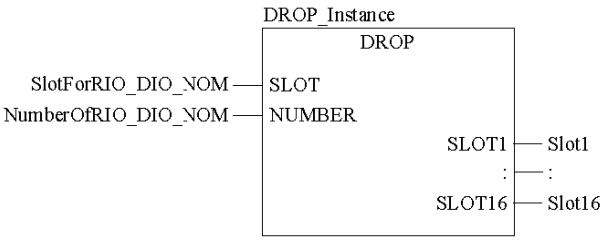
The function block is used to edit the configuration data of a remote or distributed I/O station for subsequent processing by module configuration EFBs.

To configure an I/O station rack, the DROP function block in the configuration section is connected to the corresponding SLOT output of the QUANTUM function block. The number of the I/O station defined in the I/O map has to be entered at 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 SLOT outputs.

EN and ENO can be configured as additional parameters.

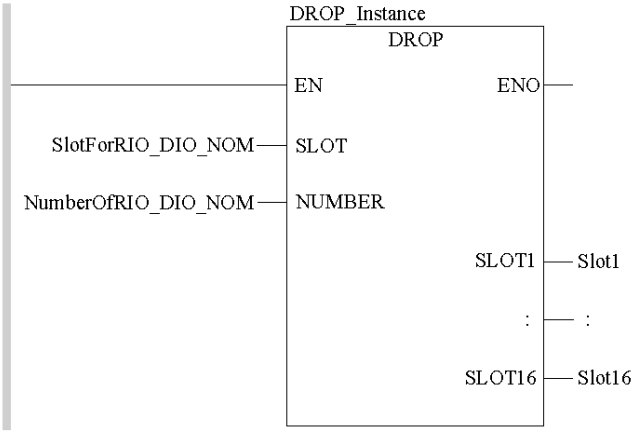
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL DROP_Instance (SLOT:=SlotForRIO_DIO_NOM,
    NUMBER:=NumberOfRIO_DIO_NOM, SLOT1=>Slot1,
    SLOT2=>Slot2, SLOT3=>Slot3, SLOT4=>Slot4, SLOT5=>Slot5,
    SLOT6=>Slot6, SLOT7=>Slot7, SLOT8=>Slot8, SLOT9=>Slot9,
    SLOT10=>Slot10, SLOT11=>Slot11, SLOT12=>Slot12,
    SLOT13=>Slot13, SLOT14=>Slot14, SLOT15=>Slot15,
    SLOT16=>Slot16)
```

Representation in ST

Representation:

```
DROP_Instance (SLOT:=SlotForRIO_DIO_NOM,
    NUMBER:=NumberOfRIO_DIO_NOM, SLOT1=>Slot1,
    SLOT2=>Slot2, SLOT3=>Slot3, SLOT4=>Slot4, SLOT5=>Slot5,
    SLOT6=>Slot6, SLOT7=>Slot7, SLOT8=>Slot8, SLOT9=>Slot9,
    SLOT10=>Slot10, SLOT11=>Slot11, SLOT12=>Slot12,
    SLOT13=>Slot13, SLOT14=>Slot14, SLOT15=>Slot15,
    SLOT16=>Slot16) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
SLOT	INT	Slot for RIO, DIO, NOM
NUMBER	DINT	Number of RIO, DIO, NOM

Description of output parameters:

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

Runtime error

If no "Head" has been configured for the I/O station rack, an error message is returned.

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 56
ERT_854_10: Data transfer EFB

Introduction

This chapter describes the ERT_854_10 block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function Description

The ERT_854_10 EFB provides the programmer with a software interface to the ERT 854 10 module which allows simple access of the functions such as counting, time stamp, status or time synchronization. The ERT_854_10 EFB coordinates the flow of Multiplex data from the ERT to the PLC using the input and output registers. It also ensures that the intermediate count values are put in an internal storage area until the data is complete, so a consistent set of all count values is made available to the statement list. A marker "New data" is always set for every data type if the input data type in the corresponding EFB output structure was copied.

The parameters EN and ENO can also be configured.

Inconsistency between EFB Output and %IW Data

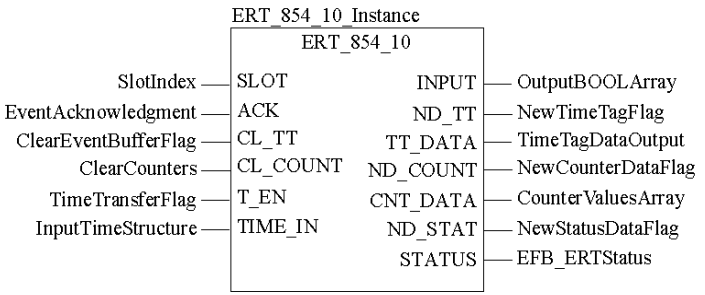
In general the %IW data correspond to the EFB output pin named INPUT.

It should be taken into account that this EFB output is inconsistent to the %IW data for a few scans after starting PLC, because of the implemented handshake mechanisms in communication between the ERT_854_10 EFB and the ERT hardware.

NOTE: In case the EFB reports any communication error the %IW data are not updated by the ERT hardware.
This means you must not use %IW data if the EFB reports a problem by returning ENO = false.

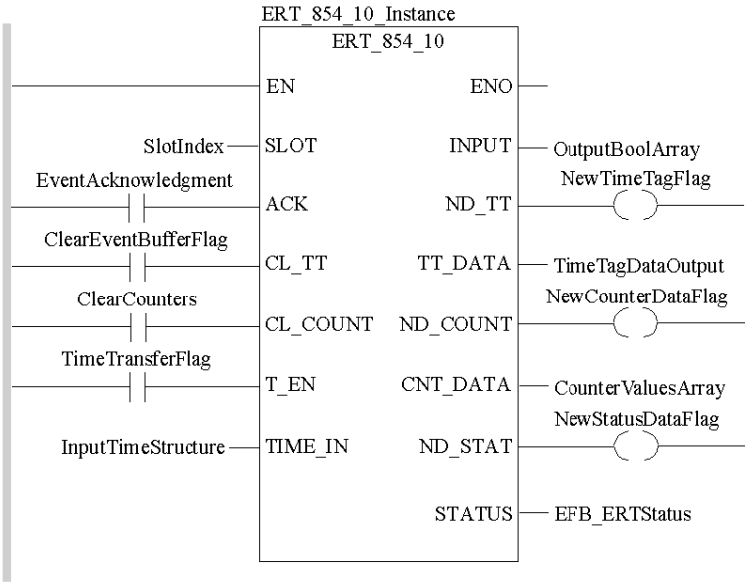
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ERT_854_10_Instance (SLOT:=SlotIndex,
    ACK:=EventAcknowledgment, CL_TT:=ClearEventBufferFlag,
    CL_COUNT:=ClearCounters, T_EN:=TimeTransferFlag,
    TIME_IN:=InputTimeStructure, INPUT=>OutputBoolArray,
    ND_TT=>NewTimeTagFlag, TT_DATA=>TimeTagDataOutput,
    ND_COUNT=>NewCounterDataFlag,
    CNT_DATA=>CounterValuesArray,
    ND_STAT=>NewStatusDataFlag, STATUS=>EFB_ERTStatus)
```

Representation in ST

Representation:

```
ERT_854_10_Instance (SLOT:=SlotIndex,
    ACK:=EventAcknowledgment, CL_TT:=ClearEventBufferFlag,
    CL_COUNT:=ClearCounters, T_EN:=TimeTransferFlag,
    TIME_IN:=InputTimeStructure, INPUT=>OutputBoolArray,
    ND_TT=>NewTimeTagFlag, TT_DATA=>TimeTagDataOutput,
    ND_COUNT=>NewCounterDataFlag,
    CNT_DATA=>CounterValuesArray,
    ND_STAT=>NewStatusDataFlag, STATUS=>EFB_ERTStatus) ;
```

Parameter Description

Description of the input parameters:

Parameter	Data type	Meaning
SLOT	INT	The Slot index is assigned to the EFB ERT_854_10 from either the QUANTUM EFB or DROP EFB and contains the configured input and output references (%IW and %MW)
ACK	BOOL	Event confirmation: Setting ACK signals that the user is ready to receive the next result and deletes the TT_DATA marker. If ACK remains set "continuous operation" appears.
CL_TT	BOOL	Delete the ERT event FIFO buffer by setting CL_TT. Saving of events is blocked until the CL_TT is reset to 0.
CL_COUNT	BOOL	Delete all ERT counters by setting CL_COUNT. Counting is interrupted until CL_COUNT is reset to 0.
T_EN	BOOL	Enables a time transfer, e.g. from the ESI using TIME_IN, if set
TIME_IN	DPM_Time	Structure of the ESI, e.g. input time through time synchronization of the ERT (carries the edge controlled time synchronization in the Sync element)

Description of the output parameters:

Parameter	Data type	Meaning
INPUT	BoolArr32	Output field for all 32 digital inputs in BOOL format (also provided in the form of word references as %IWx and %IWx+1)
ND_TT	BOOL	Marker, new data in TT_DATA structure: remains set until user confirmation with ACK
TT_DATA	ERT_10_TTag	Event message output structure with time mark. An event is held and ND_TT is set to 1 until there is a user enable with ACK = 1.
ND_COUNT	BOOL	Marker, new counter data in CNT_DATA Structure: The value 1 is set for only one cycle and is not recorded.
CNT_DATA	UDIntArr32	Output field for 32 counter values is overwritten after the EFB has received a complete set of consistent counter values (configured as: 8, 16, 24, or 32).
ND_STAT	BOOL	Marker; new status data in STATUS word: The value 1 is set for only one cycle and is not acknowledged.
STATUS	WORD	Output word for EFB/ERT status (for internal details see *Data Flow, page 256*)

Internal Time Synchronization

Structure of DPM_Time for ERT internal time synchronization, e.g. through the ESI:

Element	Element type	Meaning
Sync	BOOL	Clock synchronization with positive edge (hourly or on command)
Ms_Lsb	BYTE	Time in milliseconds (low value byte)
Ms_Msb	BYTE	Time in milliseconds (high value byte)
Min	BYTE	Time invalid / minutes
Hour	BYTE	Summer time / hours
Day	BYTE	Day of the week / Day in the month
Mon	BYTE	Month
Year	BYTE	Year

Event Structure

Event structure of the ERT_10_TTag with 5Byte time markers (further information can be found in *Data Flow, page 256*):

Element	Element type	Meaning
User	BYTE	Complete time / user number [module number]
INPUT	BYTE	Event set type / No. of the first input
In	BYTE	Event data: 1, 2 or 8 administered positions
Ms_Lsb	BYTE	Time in milliseconds (low value byte)
Ms_Msb	BYTE	Time in milliseconds (high value byte)
Min	BYTE	Time invalid / minutes
Hour	BYTE	Summer time / hours
Day	BYTE	Day of the week / Day of the month

Function mode

ERT data transfer

The number of I/O words available on the S908 remote drops is limited to 64 inputs and 64 outputs. For this reason the number of settable ERT modules per remote drop with the currently selected minimum requirements of 7 input words and 5 output words is limited to 9.

The size of the required ERT data transfer is considerably larger:

- 32 counters = 64 words,
- a event with a 5 byte time marker = 4 words,
- 32 digital values and the ERT status = 3 words.

These inconsistent size requirements necessitate the use of a special transfer EFB called `ERT_854_10` to execute the required operations on the PLC and to adjust the ERT representation of the data in Multiplex form. This type of EFB is required for every ERT module.

To simplify matters, configure only the EFB parameters which will actually be used. This saves on configuration, particularly when the counter inputs and event inputs get mixed with one another. Memory is not saved because Unity fills the outputs with invisible.

Underlying structure of the register block

Underlying structure of the `ERT_854_10` input register block with seven %IW input words for transfer from the ERT to the PLC:

Contents	Function
Digital inputs 1 ... 16	Digitally processed input data which is cyclically updated (the module's input address corresponds to that of the digital standard input modules, i.e. inputs 1 16 correspond to bits 15 0)
Digital inputs 17 ... 32	
Transfer status	IN transfer status (TS_IN)
MUX 1	Multiplex data block for block transfer, such as: ● 1 event with 5 byte time marker or ● 2 counter values of maximal configuration 32 or ● 1 status word
MUX 2	
MUX 3	
MUX 4	

Simplified structure of the `ERT_854_10` output register block with five %MW output words for the transfer from the PLC to the ERT

ERT_854_10 output register block:

Contents	Function
Transfer status	OUT transfer status (TS_OUT)
MUX 1	Time data block for the ERT for the clock synchronization
MUX 2	
MUX 3	
MUX 4	

NOTE: User interfaces are normally the inputs and outputs of the ERT_854_10 EFB, not the %IW and %MW input/output words.

EFB configuration

EFB connection

The EFB connection to the input and output references (%IW and %QW) is accomplished through a graphic connection to the ERT slot number, in the same way as with analog modules. The currently available QUANTUM and DROP EFBs from the I/O Management library are used as follows: QUANTUM for local and DROP for remote racks. These EFBs transfer an integer index to every specified slot, which points to an internal data structure with the configured values. The module parameters and the ID are stored there, in addition to the addresses and lengths of the assigned input and output references (%IW and %MW).

A significant improvement in the runtime can be achieved by deactivating the QUANTUM or the DROP EFB after the first execution.

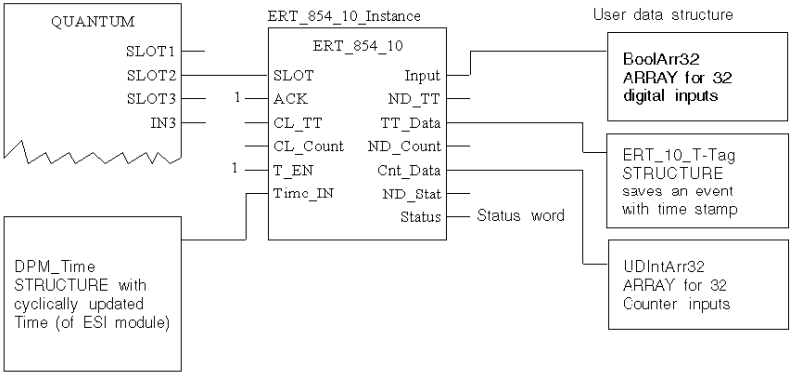
Function of CL_TT and CL_COUNT

Setting the input marker CL_TT causes the FIFO buffer event of the ERT to be cleared. Setting the markers for one cycle is sufficient.

Setting the input marker CL_COUNT causes the ERT counter to be cleared by the ERT. Setting the markers for one cycle is sufficient.

Block diagram

Principle structure



Data Flow

Digital Inputs

No marker for new data is provided for this input type. The digital inputs in the first two input register words are updated directly by the ERT in every PLC cycle. The EFB makes the processed values available as Bool if the BoolArr32 output field has been configured accordingly.

Counter Inputs

Cyclic updating of the counted values lasts significantly longer than for other data types. Counted values are saved as a data set in CNT_DATA after a complete series (configured as: 8, 16, or 32) of time consistent counted values in multiplex form has been transferred by the ERT. The marker for new data ND_COUNT is set for one cycle.

Event Inputs

Readiness to receive new events must be actively confirmed by the user, therefore the administration of markers becomes somewhat more complex (a handshake mechanism is required) Event data remain in the data structure ERT_10_TTag and the marker for new data ND_TT stays set until the ACK input is set and a new event thus requested. The EFB responds to this by resetting ND_TT for at least one cycle. After the new event has been sent to the ERT_10_TTag structure (marker structure), ND_TT is reset by the EFB. To prevent the new event data from being overwritten attention must be paid to fundamentally resetting the ACK input after the EFB has reset the ND_TT marker. This state can then remain stable to allow the user program enough time for event processing. Each subsequent event tracked with the ERT is temporarily stored within the event FIFO buffer.

New events are sent directly from the internal buffer of the EFB in intervals of at least 2 cycles for as long as the ACK input is set (for the special continuous operating mode); the effect is, however, that the ND_TT only stays set for one cycle. In this special mode the user program's task is still to terminate event processing before ND_TT signals the transfer of other new events to the ERT_10_TTag structure as handshake protection by ACK is not available in this case.

ERT_10_TTag

ERT_10_TTag event structure with 5 byte time marks

Byte	Bits	Function
1	D0...D6 = Module no. 0...127 D7 = CT	Rough time: CT = 1 indicates that this time mark contains the whole time declaration including month and year in bytes 2 + 3. The Module no. can be set in any way in the parameter screen.
2	D0D5 = input no. D6 = P1 D7 = P2	No. of the first input of the event group: 1...32 Type of the event message (P2, P1). 1..0.3 see *Note 1:, page 257* [Month value with CT = 1]

Byte	Bits	Function
3	D0D7 = data from the event group (D7D0 with right alignment)	1, 2 or 8 managed positions [year value, if CT = 1]
4	Time in milliseconds (low value byte)	0 ...59999 milliseconds (max. 61100) see *Note 2:, page 257* and *Note 3:, page 257*
5	Time in milliseconds (high value byte)	
6	D0...D5 = minutes D6 = R D7 = TI	Minutes: 0...59 Time invalid: TI = 1 means invalid time / reserved = 0 see *Note 3:, page 257*
7	D0...D4 = hours D5 = R D6 = R D7 = DS	Hours: 0...23 Summer time: DS = 1 indicates that summer time is set With shift SZ -> WZ has hour 2A and id SZ, and hour 2B has id WZ
8	D0...D4 = DOM D5...D7 = DOW	Day of the Month: 1 ... 31 Day of Week: Mon ... Sun = 1 ... 7 The day of week corresponds to CET thus it deviates from the standard used in the US (Sun = 1).

Note 1:

Interpretation for byte 2

D7 D6	Type of event message	D5...D0	No. of the first input of the event group
0 1	1 pin message	1 ... 32	Input pin number
1 0	2 pin message	1, 3, 5, ...31	First input of the group
1 1	8 pin message	1, 9, 17, 25	First input of the group

Note 2:

The value for the milliseconds is a maximum of 61100 ms with switch seconds (61000 plus a tolerance of 100 milliseconds)

Note 3:

For time markers containing an invalid time (TI = 1), the time in milliseconds is set to FFFF HEX. Minutes, hours and DOW/DOM values are invalid (i.e. undefined).

Rough time declaration

If the "rough time declaration" has been activated during the ERT configuration, the transfer of the complete time (with month/year) is executed in the following conditions: when the month changes, after the module restarts, during every start or stop of the PLC user program, when the event FIFO buffer is deleted, when the clock is started or set. If this rough time declaration is sent without the data input values, "triggering" basically takes place through a correct time stamped event. If this does not happen the values remain "stuck" in the ERT until an event occurs. Within the time mark of a "rough time declaration", the CT bit is always set so that byte 2 contains the information about the month, byte 3 the information about the year and bytes 4 to 8 display the same time mark values of the triggered event whose event message appears immediately after the rough time declaration.

Status Inputs

The marker for new status data `ND_STAT` is set for one cycle. The status inputs can be overwritten after 2 inquiry cycles.

The status word contains EFB and ERT error bits

Division of the Error Bits

Internal structure of the EFB/ERT status word:

EFB error bits					**ERT error bits**										
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

ERT Error Bits

D8 ... D0 ERT error bits

Bit	Brief designation	Meaning
D0	FW	Firmware errors, self test errors within internal memories (severe module errors)
D1	FP	Parameterization errors (severe internal errors)
D2	TE	External time reference error (time-basis signal disrupted or not present)
D3	TU	Time became invalid
D4	TA	Time is not synchronized (Free run mode, permanent run without time error message, see also:*Without power reserve, page 262*
D5	PF	FIFO buffer overrun (loss of the most recent event data)
D6	PH	FIFO buffer half full
D7	DC	Stabilize active (some event data lost)
D8	CE	ERT communication errors (procedure errors or time out)

When configuring the parameter screen some of these errors can be assigned to grouped error messages with the "F" light as well as the module's error byte within the status table. All other errors are then defined as warnings.

D11 ... D9 reserved

EFB Error Bits

D15 ... D12 EFB error bits:

Bin.	Hex	Meaning
1001	9 HEX	wrong answer recognized, command (EFB internal error)
1000	8 HEX	EFB communication time out
0101	5 HEX	Wrong slot
0110	6 HEX	Health status bit is not set (ERT appears as not available)
Other values		Internal error

Online error display

The following ERT/EFB error messages are displayed in the **Tools** → **Diagnostic Display** UNITY window with an error number and explanation.

EFB error messages:

Message	Error	Meaning
-30210	User error 11	communication time out occurred
-30211	User error 12	wrong answer recognized, synchronization (EFB internal error)
-30212	User error 13	wrong packet number detected (EFB internal error)
-30213	User error 14	wrong field number detected (EFB internal error)
-30214	User error 15	unexpected time tag (EFB internal error)
-30215	User error 16	wrong slot data (configuration check required)
-30216	User error 17	health status bit is not set (ERT appears as not available)
-30217	User error 18	EFB internal command buffer out of bounds
-30218	User error 19	wrong answer recognized, command (EFB internal error)
-30219	User error 20	ERT error

ERT error messages:

Message	Error	Meaning
-30200	User error 1	ERT internal error
...	...	...
-30203	User error 4	ERT internal error
-30204	User error 5	ERT communication timeout
-30205	User error 6	ERT internal error
...	...	...
-30207	User error 8	ERT internal error

Other Functions

Input marker

Setting the input marker `CL_TT` deletes the Event FIFO buffer of the ERT. Setting the marker for one cycle is sufficient.

If the input marker `CL_Count` is set, the ERT counter is deleted by the EFB. Setting the marker for one cycle is sufficient.

Use of the DPM_Time structure for the synchronization of the internal ERT clock

Time synchronization

If the time cannot be synchronized through a standard time receiver, the time information can alternatively be transferred from the 140 ESI 062 01 communication module. The ESI makes the updated time available directly to the EFB in a DPM_Time structure via the TIME_IN parameter. The data structure can also be filled by the user program and the respective bits can be managed. In this way, for example, the time can be set by the CPU.

With power reserve

As soon as the "clock" parameter of the ERT is configured as an "internal clock" with a power reserve not equal to zero (i.e. not free running) the EFB must use the time supplied by the ESI for the synchronization of the internal ERT clock. Until the first synchronization has taken place, the ERT sends back the set Bit "invalid time" in the STATUS output word (Bit 3 TU).

The conditions for the first synchronization of the internal ERT clock via the DPM_Time structure are:

The EFB Parameter T_EN must change from 0 to 1 to enable the time setting.

The time in TIME_IN made available by ESI must look as follows:
- valid (i.e. the bit for the message "time invalid" in Min value must not be set),
- and the values in Ms must change continually.

Should the time data later become invalid or no longer set, then the TU does not switch to 1 until the configured power reserve has expired.

The synchronization/setting of the internal ERT clock takes place via the DPM_Time structure, if:
- EFB-Parameter T_EN is set to 1 to enable the time setting.
- The time data in TIME_IN made available by ESI are valid (i.e. the "Time invalid" Bit in the Min value must not be set).
- The status of the DPM_Time element Sync changes from 0 to 1. This change is run every full hour by the 140 ESI 062 01 but can also be performed as the result of a suitable telecontrol command.

The precision of the time synchronized by the ESI at the ERT can be influenced by delays, by the PLC cycle time, as well as by the cumulative component, which reflects the differences in the ERT software clock (< 360 milliseconds/hour).

Without power reserve

If the "clock" parameter of the ERT was configured as an "internal clock" in free running mode (with a power reserve of zero), the internal clock starts with a default setting at hour 0 on 1/1/1990. In this case the time can also be provided by using the DPM_Time data structure of the 140 ESI 062 01 module, as described above. As there is no power reserve available for use, the time will never be invalid and the Bit "Time not synchronized" within the STATUS output word (Bit 4 TA), given back by the EFB, is always set.

Using the ERT >EFB Time Data Flow

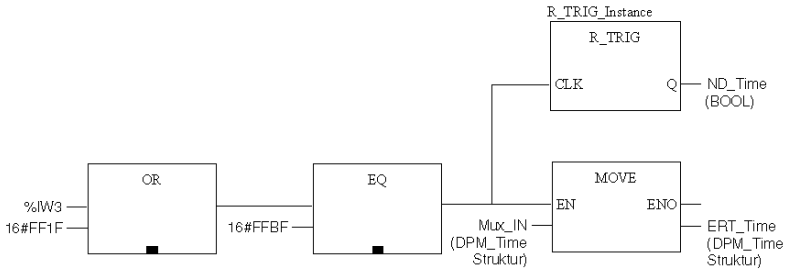
Application examples

This section presents an internal function which is made available through the ERT for diagnostics and development. It covers the cyclic transfer of the ERT internal time to the corresponding EFB in greater intervals. This time application can be used to display or set the PLC clock etc, regardless of whether it comes from the free-running internal clock or was synchronized through an external reference clock signal. The time appears as a DPM_Time structure beginning with word 4 of the IN register block of the ERT. The following diagram shows the program elements involved in selection.

Commissioning information

A ERT_854_10 was assigned the IN references %IW1 ... . %IW3 during I/O addressing. The IN transfer status (TS_IN) in the third word of the register block is sent to an OR block. A DPM_Time structure is defined within the variable editor as Variable Mux_IN in the fourth word of the IN register block, and therefore has the address %IW4 ... %IW7. This variable is sent to the MOVE block as an entry. The MOVE block output is a DPM_Time structure defined by the variable editor as variable ERT_Time.

Typical recording mechanism for ERT time data



NOTE: The ERT_854_10 EFB must be active and error free.

Explanation:

The MOVE block transfers the time data cyclically stored in the MUX zone of the IN register block to the DPM_Time structure ERT_Time belonging to the user as soon as the OR and the EQ block signals a time data transfer. R_TRIG makes a signal in ND_Time available for further processing of the time data available for one cycle. The BOOL Sync element value of the ERT_Time should begin to "tick" during each new transfer from the ERT. There is a new transfer after a maximum of each 200 PLC cycles.

Chapter 57
ERT_854_20: Data transfer EFB

Introduction

This chapter describes the ERT_854_20 block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function Description

The ERT_854_20 EFB provides a software interface to the ERT 854 20 module which gives you simple access of the functions such as counting, time stamp, status or time synchronization. The ERT_854_20 EFB coordinates the flow of Multiplex data from the ERT to the PLC using the input and output registers. It also puts the intermediate count values in an internal storage area until the data is complete, so a consistent set of all count values is made available to the statement list. A marker "New data" is always set for every data type if the input data type in the corresponding EFB output structure was copied.

As additional parameters, EN and ENO can be configured.

Inconsistency between EFB Output and %IW Data

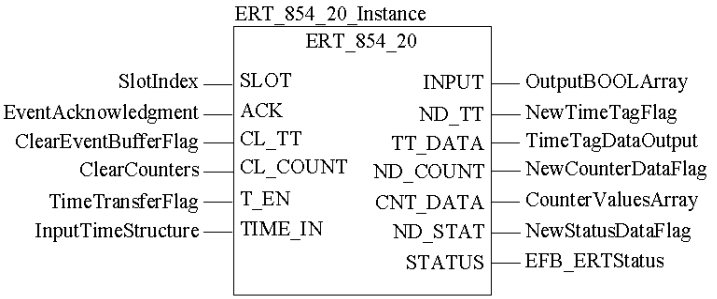
In general the %IW data correspond to the EFB output pin named INPUT.

It should be taken into account that this EFB output is inconsistent to the %IW data for a few scans after starting PLC, because of the implemented handshake mechanisms in communication between the ERT_854_20 EFB and the ERT hardware.

NOTE: In case the EFB reports a detected communication error the %IW data are not updated by the ERT hardware.
Do not use %IW data if the EFB returns ENO = false.

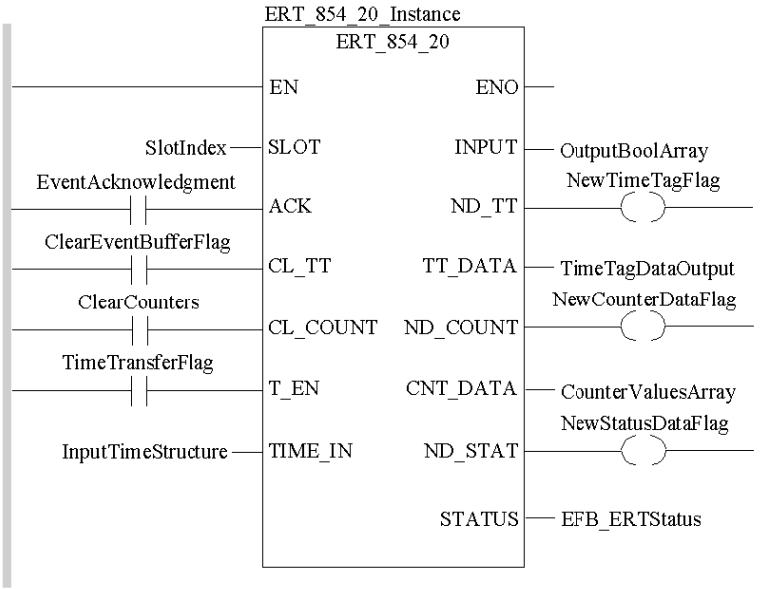
Appearance in FBD

Representation:



Appearance in LD

Representation:



Appearance in IL

Appearance:

```
CAL ERT_854_20_Instance (SLOT:=SlotIndex,
    ACK:=EventAcknowledgment, CL_TT:=ClearEventBufferFlag,
    CL_COUNT:=ClearCounters, T_EN:=TimeTransferFlag,
    TIME_IN:=InputTimeStructure, INPUT=>OutputBoolArray,
    ND_TT=>NewTimeTagFlag, TT_DATA=>TimeTagDataOutput,
    ND_COUNT=>NewCounterDataFlag,
    CNT_DATA=>CounterValuesArray,
    ND_STAT=>NewStatusDataFlag, STATUS=>EFB_ERTStatus)
```

Appearance in ST

Appearance:

```
ERT_854_20_Instance (SLOT:=SlotIndex,
    ACK:=EventAcknowledgment, CL_TT:=ClearEventBufferFlag,
    CL_COUNT:=ClearCounters, T_EN:=TimeTransferFlag,
    TIME_IN:=InputTimeStructure, INPUT=>OutputBoolArray,
    ND_TT=>NewTimeTagFlag, TT_DATA=>TimeTagDataOutput,
    ND_COUNT=>NewCounterDataFlag,
    CNT_DATA=>CounterValuesArray,
    ND_STAT=>NewStatusDataFlag, STATUS=>EFB_ERTStatus) ;
```

Parameter Description

Description of the input parameters:

Parameter	Data type	Meaning
SLOT	INT	The Slot index is assigned to the EFB ERT_854_20 from either the QUANTUM or DROP EFB and contains the configured input and output references (%IW und %MW).
ACK	BOOL	Event confirmation: Setting ACK signals that the user is ready to receive the next event and deletes the TT_DATA marker. If ACK remains set, "Continuous operation" is executed.
CL_TT	BOOL	Delete the ERT event FIFO buffer by setting CL_TT. Storage of events is blocked until CL_TT is reset to 0.
CL_COUNT	BOOL	Clears all ERT counters by setting CL_COUNT. Counting is interrupted until CL_COUNT is reset to 0.
T_EN	BOOL	Enables a time transfer, that is from the ESI using TIME_IN if set.
TIME_IN	DPM_Time	Structure of the ESI, that is input time through time synchronization of the ERT (carries the edge controlled time synchronization in the Sync element).

Description of output parameters:

Parameter	Data type	Meaning
INPUT	BOOLArr32	Output array for all 32 digital inputs in BOOL format. (Also provided in the form of word references as %IWx and %IWx+1).
ND_TT	BOOL	Marker, new data in TT_DATA structure: remains set until acknowledged by the user with ACK.
TT_DATA	ERT_10_TTag	Event message output structure with time stamp. An event is held and ND_TT is set to 1 until acknowledged by the user with ACK = 1.
ND_COUNT	BOOL	Marker, new counter data in CNT_DATA structure: The value 1 is set for only 1 cycle and is not acknowledged.
CNT_DATA	UDIntArr32	Output array for 32 counter values (is overwritten after the EFB has received a complete set of consistent counter values (configured as: 8, 16, 24, or 32).
ND_STAT	BOOL	Marker; new status data in STATUS word: The value 1 is set for only 1 cycle and is not acknowledged.
STATUS	WORD	Output word for EFB/ERT status. For more details, refer to *Data Flow (see page 274)*.

Internal Time Synchronization

Structure of DPM_Time for ERT internal time synchronization, that is through the ESI:

Element	Element type	Meaning
Sync	BOOL	Clock synchronization with positive edge (hourly or on command)
Ms_Lsb	BYTE	Time in milliseconds (least significant byte)
Ms_Msb	BYTE	Time in milliseconds (most significant byte)
Min	BYTE	Invalid time / minutes
Hour	BYTE	Summer time / hours
Day	BYTE	Week day / day of month
Mon	BYTE	Month
Year	BYTE	Year

Event Structure

Event structure of the ERT_10_TTag with 5 Byte time markers (more information can be found in *Data Flow (see page 274)*):

Element	Element type	Meaning
User	BYTE	Complete time/user number [module number]
INPUT	BYTE	Event set type/Number of the first input
In	BYTE	Event data: 1, 2 or 8 scheduled positions
Ms_Lsb	BYTE	Time in milliseconds (least significant byte)
Ms_Msb	BYTE	Time in milliseconds (most significant byte)
Min	BYTE	Invalid time/minutes
Hour	BYTE	Summer time/hours
Day	BYTE	Weekday/Day of the month

Function mode

ERT data transfer

The number of I/O words available on the S908 remote drops is limited to 64 inputs and 64 outputs. For this reason, the number of settable ERT modules per remote drop with the currently selected minimum requirements of 8 input words and 5 output words is limited to 8. There is no quantity limitation of ERT modules in EIO drops.

The size of the required ERT data transfer is considerably larger:

- 32 counters = 64 words,
- an event with a 5 byte time marker = 4 words,
- 32 digital values and the ERT status = 3 words.

These inconsistent size requirements necessitate the use of a special transfer EFB called `ERT_854_20` to execute the required operations on the PLC and to adjust the ERT representation of the data in Multiplex form. This type of EFB is required for every ERT module.

To simplify matters, configure only the EFB parameters which will actually be used. This saves on configuration, particularly when the counter inputs and event inputs get mixed with one another. Memory is not saved because Unity Pro fills the outputs with invisible.

Underlying structure of the register block

Underlying structure of the `ERT_854_20` input register block with 8 %IW input words for transfer from the ERT to the PLC:

Contents	Function
Digital inputs 1...16	Digitally processed input data which is cyclically updated (the module's input address corresponds to that of the digital standard input modules, that is inputs 1...16 correspond to bits 15...0)
Digital inputs 17...32	
Transfer status	IN transfer status (TS_IN)
MUX 1	Multiplex data block for block transfer, such as: ● 1 event with 5 byte time marker or ● 2 counter values of maximal configuration 32 or ● 1 status word
MUX 2	
MUX 3	
MUX 4	
RESERVED	Reserved for internal use

Simplified structure of the `ERT_854_20` output register block with 5 %MW output words for the transfer from the PLC to the ERT.

ERT_854_20 output register block:

Contents	Function
Transfer status	OUT transfer status (TS_OUT)
MUX 1	Time data block for the ERT for the clock synchronization
MUX 2	
MUX 3	
MUX 4	

NOTE: User interfaces are normally the inputs and outputs of the ERT_854_20 EFB, not the %IW and %MW input/output words.

EFB configuration

EFB connection

The EFB connection to the input and output references (%IW and %QW) is accomplished through a graphic connection to the ERT slot number, in the same way as with analog modules. The currently available QUANTUM and DROP EFBs from the I/O Management library are used as follows:

- QUANTUM for local
- DROP for remote racks

These EFBs transfer an integer index to every specified slot, which points to an internal data structure with the configured values. The module parameters and the ID are stored there, in addition to the addresses and lengths of the assigned input and output references (%IW and %MW).

A significant improvement in the runtime can be achieved by deactivating the QUANTUM or the DROP EFB after the first execution.

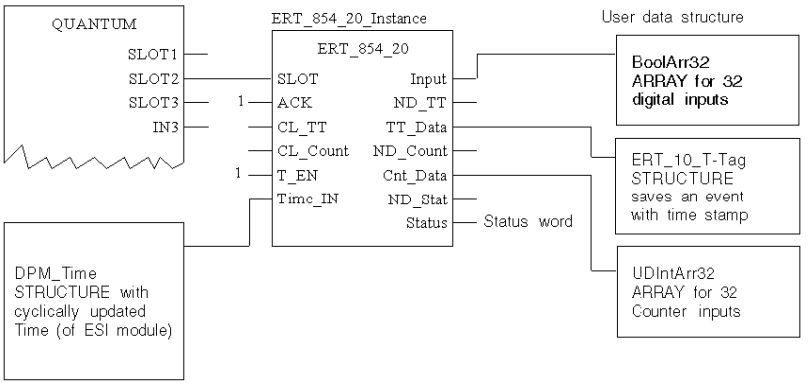
Function of CL_TT and CL_COUNT

Setting the input marker CL_TT causes the FIFO buffer event of the ERT to be cleared. Setting the markers for one cycle is sufficient.

Setting the input marker CL_COUNT causes the ERT counter to be cleared by the ERT. Setting the markers for one cycle is sufficient.

Block diagram

Principle structure:



Data Flow

Digital Inputs

No marker for new data is provided for this input type. The digital inputs in the first two input register words are updated directly by the ERT in every PLC cycle. The EFB makes the processed values available as Bool if the BoolArr32 output field has been configured accordingly.

Counter Inputs

Cyclic updating of the counted values significantly last longer than for other data types. Counted values are saved as a data set in CNT_DATA after a complete series (configured as: 8, 16, or 32) of time consistent counted values in multiplex form has been transferred by the ERT. The marker for new data ND_COUNT is set for one cycle.

Event Inputs

You need to confirm your readiness to receive new events. Therefore the administration of markers becomes somewhat more complex (a handshake mechanism is required). Event data remains in the data structure ERT_10_TTag and the marker for new data ND_TT stays set until the ACK input is set and a new event thus requested. The EFB responds to this by resetting ND_TT for at least one cycle. After the new event has been sent to the ERT_10_TTag structure (marker structure), ND_TT is reset by the EFB. Reset the ACK input after the EFB has reset the ND_TT marker so that new event data does not get overwritten. This state can then remain stable to allow the user program enough time for event processing. Each subsequent event tracked with the ERT is temporarily stored within the event FIFO buffer.

New events are sent directly from the internal buffer of the EFB in intervals of at least 2 cycles for as long as the ACK input is set (for the special continuous operating mode); the effect is, however, that the ND_TT only stays set for one cycle. In this special mode the user program's task is still to terminate event processing before ND_TT signals the transfer of other new events to the ERT_10_TTag structure as handshake protection by ACK is not available in this case.

ERT_10_TTag

ERT_10_TTag event structure with 5 byte time marks:

Byte	Bits	Function
1	D0...D6 = Module number 0...127 D7 = CT	Rough time: CT = 1 indicates that this time mark contains the whole time declaration including month and year in bytes 2 + 3. The Module no. can be set in any way in the parameter screen.
2	D0D5 = input number D6 = P1 D7 = P2	Number of the first input of the event group: 1...32 Type of the event message (P2, P1). 1..0.3 see *Note 1:, page 275* [Month value with CT = 1]

Byte	Bits	Function
3	D0D7 = data from the event group (D7D0 with right alignment)	1, 2 or 8 managed positions [year value, if CT = 1]
4	Time in milliseconds (low value byte)	0...59999 milliseconds (maximum 61100) see *Note 2:, page 275* and *Note 3:, page 275*
5	Time in milliseconds (high value byte)	
6	D0...D5 = minutes D6 = R D7 = TI	Minutes: 0...59 Time invalid: TI = 1 means invalid time/reserved = 0 see *Note 3:, page 275*
7	D0...D4 = hours D5 = R D6 = R D7 = DS	Hours: 0...23 Summer time: DS = 1 indicates that summer time is set With shift SZ -> WZ has hour 2A and id SZ, and hour 2B has id WZ
8	D0...D4 = DOM D5...D7 = DOW	Day of the Month: 1...31 Day of Week: Mon...Sun = 1...7 The day of week corresponds to CET thus it deviates from the standard used in the US (Sun = 1).

Note 1:

Interpretation for byte 2:

D7 D6	Type of event message	D5...D0	Number of the first input of the event group
0 1	1 pin message	1...32	Input pin number
1 0	2 pin message	1, 3, 5...31	First input of the group
1 1	8 pin message	1, 9, 17, 25	First input of the group

Note 2:

The value for the milliseconds is a maximum of 61100 ms with switch seconds (61000 plus a tolerance of 100 milliseconds).

Note 3:

For time markers containing an invalid time (TI = 1), the time in milliseconds is set to FFFF HEX. Minutes, hours and DOW/DOM values are invalid (that is undefined).

Rough time declaration

If the "rough time declaration" has been activated during the ERT configuration, the transfer of the complete time (with month/year) is executed in the following conditions:

- when the month changes,
- after the module restarts,
- during every start or stop of the PLC user program,
- when the event FIFO buffer is deleted,
- when the clock is started or set.

If this rough time declaration is sent without the data input values, "triggering" basically takes place through a correct time stamped event. If this does not happen the values remain "stuck" in the ERT until an event occurs. Within the time mark of a "rough time declaration", the CT bit is set so that byte 2 contains the information about the month, byte 3 the information about the year and bytes 4...8 display the same time mark values of the triggered event whose event message appears immediately after the rough time declaration.

Status Inputs

The marker for new status data ND_STAT is set for one cycle. The status inputs can be overwritten after 2 inquiry cycles.

The status word contains EFB and ERT error bits.

Division of the Error Bits

Internal structure of the EFB/ERT status word:

EFB error bits				**ERT error bits**											
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

ERT Error Bits

D8...D0 ERT error bits:

Bit	Brief designation	Meaning
D0	FW	Firmware mismatches self test errors within internal memories
D1	FP	Parameterization errors
D2	TE	External time reference error (time-basis signal disrupted or not present)
D3	TU	Time became invalid
D4	TA	Time is not synchronized (Free run mode, permanent run without time error message). Refer to *Without power reserve (see page 280)*.
D5	PF	FIFO buffer overrun (loss of the most recent event data)

Bit	Brief designation	Meaning
D6	PH	FIFO buffer half full
D7	DC	Stabilize active (some event data lost)
D8	CE	ERT communication errors (procedure errors or time out)

When configuring the parameter screen some of these errors can be assigned to grouped error messages with the "F" light as well as the module's error byte within the status table. All other errors are then defined as warnings.

D11...D9 reserved.

EFB Error Bits

D15...D12 EFB error bits:

Bin.	Hex	Meaning
1001	9 HEX	Wrong answer recognized, command (EFB internal error)
1000	8 HEX	EFB communication time out
0101	5 HEX	Wrong slot
0110	6 HEX	Health status bit is not set (ERT appears as not available)
1010	A HEX	CRC checksum error
Other values	–	Internal error

Online error display

The following ERT/EFB error messages are displayed in the **Tools** → **Diagnostic View** UNITY window with a number and explanation.

EFB error messages:

Message	Error	Meaning
-30210	User error 11	Communication time out occurred
-30211	User error 12	Wrong answer recognized, synchronization (EFB internal error)
-30212	User error 13	Wrong packet number detected (EFB internal error)
-30213	User error 14	Wrong field number detected (EFB internal error)
-30214	User error 15	Unexpected time tag (EFB internal error)
-30215	User error 16	Wrong slot data (configuration check required)

Message	Error	Meaning
-30216	User error 17	Health status bit is not set (ERT appears as not available)
-30217	User error 18	EFB internal command buffer out of bounds
-30218	User error 19	Wrong answer recognized, command (EFB internal error)
-30219	User error 20	ERT error
-30220	User error 21	CRC checksum error

ERT error messages:

Message	Error	Meaning
-30200	User error 1	ERT internal error
...	...	...
-30203	User error 4	ERT internal error
-30204	User error 5	ERT communication timeout
-30205	User error 6	ERT internal error
...	...	...
-30207	User error 8	ERT internal error

Other Functions

Input marker

Setting the input marker `CL_TT` deletes the Event FIFO buffer of the ERT. Setting the marker for one cycle is sufficient.

If the input marker `CL_Count` is set, the ERT counter is deleted by the EFB. Setting the marker for one cycle is sufficient.

Use of the DPM_Time structure for the synchronization of the internal ERT clock

Time synchronization

If the time cannot be synchronized through a standard time receiver, the time information can alternatively be transferred from the 140 ESI 062 01 communication module. The ESI makes the updated time available directly to the EFB in a DPM_Time structure via the TIME_IN parameter. The data structure can also be filled by the user program and the respective bits can be managed. In this way, the time can be set by the CPU.

With power reserve

As soon as the "clock" parameter of the ERT is configured as an "internal clock" with a power reserve not equal to 0 (that is not free running). The EFB uses the time supplied by the ESI for the synchronization of the internal ERT clock. Until the first synchronization has taken place, the ERT sends back the set Bit "invalid time" in the STATUS output word (Bit 3 TU).

The conditions for the first synchronization of the internal ERT clock via the DPM_Time structure are:

The EFB Parameter T_EN changes from 0 to 1 to enable the time setting.

The time in TIME_IN made available by ESI appears as follows:
- valid (for example, the bit for the message "time invalid" in Min value is not set),
- and the values in Ms change continually.

Should the time data later become invalid or no longer set, then the TU does not switch to 1 until the configured power reserve has expired.

The synchronization/setting of the internal ERT clock takes place via the DPM_Time structure, if:
- EFB-Parameter T_EN is set to 1 to enable the time setting.
- The time data in TIME_IN made available by ESI are valid (for example, the "Time invalid" Bit in the Min value is not set).
- The status of the DPM_Time element Sync changes from 0 to 1. This change is run every full hour by the 140 ESI 062 01 but can also be performed as the result of a suitable telecontrol command.

The precision of the time synchronized by the ESI at the ERT can be influenced by delays, by the PLC cycle time, as well as by the cumulative component, which reflects the differences in the ERT software clock (< 360 milliseconds/hour).

Without power reserve

If the "clock" parameter of the ERT was configured as an "internal clock" in free running mode (with a power reserve of zero), the internal clock starts with a default setting at hour 0 on 1/1/1990. In this case the time can also be provided by using the DPM_Time data structure of the 140 ESI 062 01 module, as described above. As there is no power reserve available for use, the time will not be invalid and the Bit "Time not synchronized" within the STATUS output word (Bit 4 TA), given back by the EFB, is set.

Using the ERT >EFB Time Data Flow

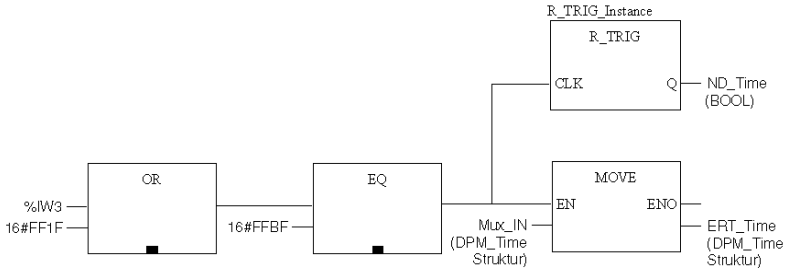
Application examples

This section presents an internal function which is made available through the ERT for diagnostics and development. It covers the cyclic transfer of the ERT internal time to the corresponding EFB in greater intervals. This time application can be used to display or set the PLC clock etc, regardless of whether it comes from the free-running internal clock or was synchronized through an external reference clock signal. The time appears as a `DPM_Time` structure beginning with word 4 of the IN register block of the ERT. The following diagram shows the program elements involved in selection.

Commissioning information

An `ERT_854_20` was assigned the IN references %IW1...%IW3 during I/O addressing. The IN transfer status (TS_IN) in the third word of the register block is sent to an OR block. A `DPM_Time` structure is defined within the variable editor as Variable Mux_IN in the fourth word of the IN register block, and therefore has the address %IW4...%IW8. This variable is sent to the `MOVE` block as an entry. The `MOVE` block output is a `DPM_Time` structure defined by the variable editor as variable `ERT_Time`.

Typical recording mechanism for ERT time data:



NOTE: The `ERT_854_20` EFB must be active and error free.

Explanation:

The `MOVE` block transfers the time data cyclically stored in the MUX zone of the IN register block to the `DPM_Time` structure `ERT_Time` belonging to the user as soon as the `OR` and the `EQ` block signals a time data transfer. `R_TRIG` makes a signal in `ND_Time` available for further processing of the time data available for one cycle. The `BOOL Sync` element value of the `ERT_Time` should begin to "tick" during each new transfer from the ERT. There is a new transfer after a maximum of each 200 PLC cycles.

Chapter 58
QUANTUM: Configuring a main rack

Description

Function description

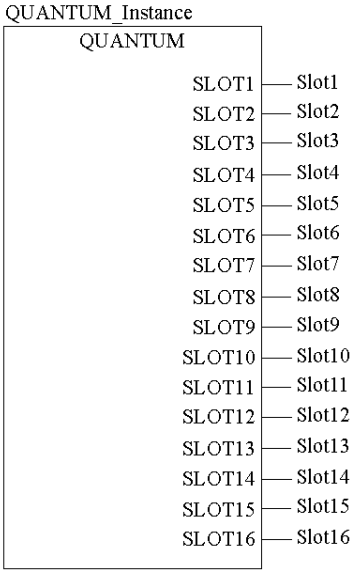
The function block is used to edit the configuration data of a QUANTUM main rack for subsequent use by the scaling EFBs.

To configure a Quantum main rack, 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.

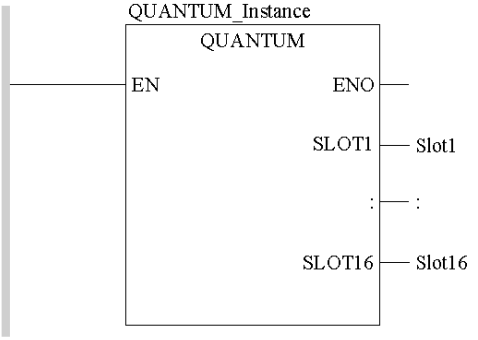
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL QUANTUM_Instance (SLOT1=>Slot1, SLOT2=>Slot2,
    SLOT3=>Slot3, SLOT4=>Slot4, SLOT5=>Slot5, SLOT6=>Slot6,
    SLOT7=>Slot7, SLOT8=>Slot8, SLOT9=>Slot9,
    SLOT10=>Slot10, SLOT11=>Slot11, SLOT12=>Slot12,
    SLOT13=>Slot13, SLOT14=>Slot14, SLOT15=>Slot15,
    SLOT16=>Slot16)
```

Representation in ST

Representation:

```
QUANTUM_Instance (SLOT1=>Slot1, SLOT2=>Slot2,
    SLOT3=>Slot3, SLOT4=>Slot4, SLOT5=>Slot5, SLOT6=>Slot6,
    SLOT7=>Slot7, SLOT8=>Slot8, SLOT9=>Slot9,
    SLOT10=>Slot10, SLOT11=>Slot11, SLOT12=>Slot12,
    SLOT13=>Slot13, SLOT14=>Slot14, SLOT15=>Slot15,
    SLOT16=>Slot16) ;
```

Parameter description

Description of output parameters:

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

Runtime error

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

NOTE: For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 59
XBE: Configuring a module rack expansion

Description

Function description

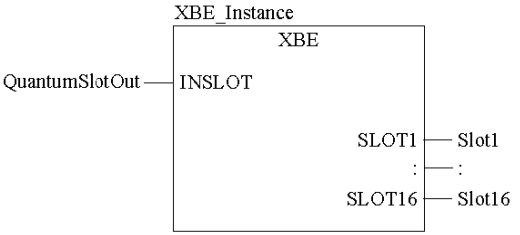
The function block is used to edit the configuration data of a Quantum (main) module rack expansion for subsequent use by the scaling EFBs.

To configure a Quantum (main) module rack expansion, the XBE function block is inserted into the configuration section *(see page 31)*. It is connected to its INSLOT input at the corresponding SLOTx output of the QUANTUM function block. The function blocks for the configuration of analog modules or the DROP function block for the I/O station are connected at its SLOTx outputs.

EN and ENO can be configured as additional parameters.

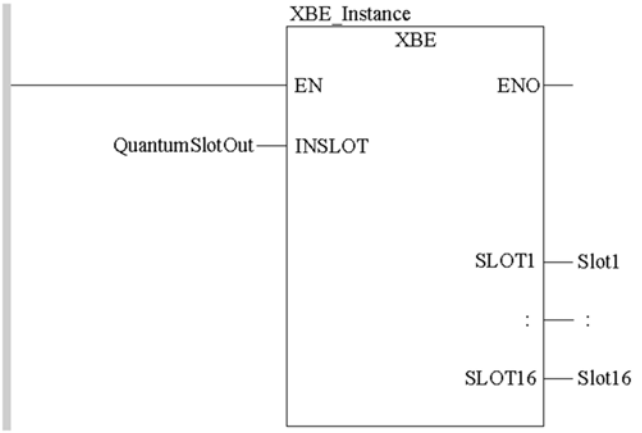
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL XBE_Instance (INSLOT:=QuantumSlotOut,
    SLOT1=>Slot1, SLOT2=>Slot2, SLOT3=>Slot3, SLOT4=>Slot4,
    SLOT5=>Slot5, SLOT6=>Slot6, SLOT7=>Slot7, SLOT8=>Slot8,
    SLOT9=>Slot9, SLOT10=>Slot10, SLOT11=>Slot11,
    SLOT12=>Slot12, SLOT13=>Slot13, SLOT14=>Slot14,
    SLOT15=>Slot15, SLOT16=>Slot16)
```

Representation in ST

Representation:

```
XBE_Instance (INSLOT:=QuantumSlotOut,
    SLOT1=>Slot1, SLOT2=>Slot2, SLOT3=>Slot3, SLOT4=>Slot4,
    SLOT5=>Slot5, SLOT6=>Slot6, SLOT7=>Slot7, SLOT8=>Slot8,
    SLOT9=>Slot9, SLOT10=>Slot10, SLOT11=>Slot11,
    SLOT12=>Slot12, SLOT13=>Slot13, SLOT14=>Slot14,
    SLOT15=>Slot15, SLOT16=>Slot16) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
INSLOT	INT	140 XBE 100 00 slot in the central rack

Description of output parameters:

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

Runtime error

For a list of all block error codes and values, see*Quantum I/O Configuration, page 319*.

Chapter 60
XDROP: Configuring a module rack expansion

Description

Functional description

The function block is used to edit the configuration data for a remote or distributed module rack expansion (via Module XBE) for subsequent processing by module configuration EFBs.

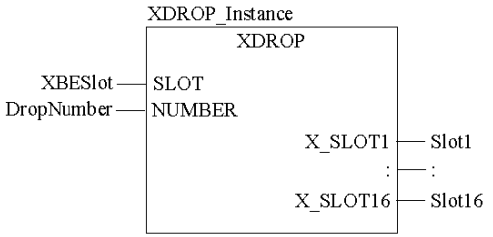
To configure an module rack expansion, the SLOT input in the configuration section *(see page 31)* is connected to the function blockSLOT input of the DROP *(see page 243)* function block. The number of XDROP function block input has to be entered at the NUMBER input of the DROP function block. The function blocks for configuration of the analog modules of the I/O stations are connected to the X_SLOT outputs.

NOTE: The function block XDROP is only used to edit the configuration data for a remote or **distributed**module rack expansion, see also*Procedure for expanding the remote module rack using the XBE module, page 38*. The configuration of the main module rack expansion is used for the XBE *(see page 287)* function block.

EN and ENO can be configured as additional parameters.

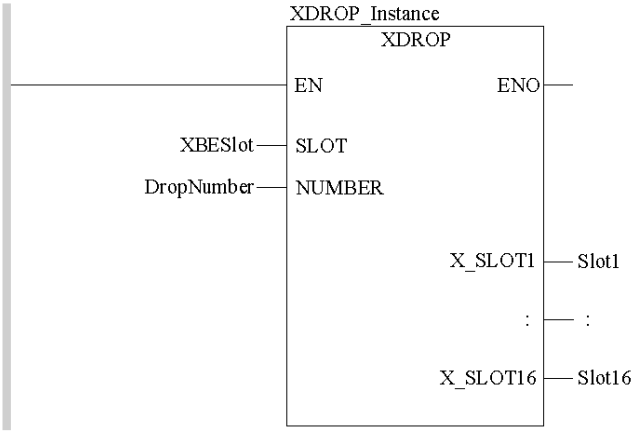
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL XDROP_Instance (SLOT:=XBESlot, NUMBER:=DropNumber,
    X_SLOT1=>Slot1, X_SLOT2=>Slot2, X_SLOT3=>Slot3,
    X_SLOT4=>Slot4, X_SLOT5=>Slot5, X_SLOT6=>Slot6,
    X_SLOT7=>Slot7, X_SLOT8=>Slot8, X_SLOT9=>Slot9,
    X_SLOT10=>Slot10, X_SLOT11=>Slot11, X_SLOT12=>Slot12,
    X_SLOT13=>Slot13, X_SLOT14=>Slot14, X_SLOT15=>Slot15,
    X_SLOT16=>Slot16)
```

Representation in ST

Representation:

```
XDROP_Instance (SLOT:=XBESlot, NUMBER:=DropNumber,
    X_SLOT1=>Slot1, X_SLOT2=>Slot2, X_SLOT3=>Slot3,
    X_SLOT4=>Slot4, X_SLOT5=>Slot5, X_SLOT6=>Slot6,
    X_SLOT7=>Slot7, X_SLOT8=>Slot8, X_SLOT9=>Slot9,
    X_SLOT10=>Slot10, X_SLOT11=>Slot11, X_SLOT12=>Slot12,
    X_SLOT13=>Slot13, X_SLOT14=>Slot14, X_SLOT15=>Slot15,
    X_SLOT16=>Slot16) ;
```

Parameter description

Input parameter description:

Parameter	Data type	Meaning
SLOT	INT	XBE slot in the central rack
NUMBER	DINT	Number of remote drop

Description of output parameters:

Parameter	Data type	Meaning
X_SLOT1	INT	Expansion Connector 1
:	:	:
X_SLOT16	INT	Expansion Connector 16

Runtime error

If no "Head" has been configured for the I/O station rack, an error message is returned in (**Tools → Diagnostics display**).

NOTE: For a list of all block error codes and values, see *Quantum I/O Configuration, page 319*.

Part VII
Simulation

Overview

This section describes the elementary functions and elementary function blocks from the family `Simulation`.

What Is in This Part?

This part contains the following chapters:

Chapter 61
WRITE_INPUT_DINT: Writing Inputs of Type DINT

Description

Function Description

WRITE_INPUT_DINT is used to simulate (write) a value at a %ID input. The input is written directly when WRITE_INPUT_DINT is invoked.

The function block can be used with a real PLC and the PLC simulator.

Restrictions

- The function block provides access to one variable at a time (elementary data types and not arrays of elementary data types).
- Only variable addresses are allowed, you cannot use declared variables even if they are mapped on a %ID direct address.
- The simulation block WRITE_INPUT_DINT does not work if the I/O module is configured in a RIO rack but works properly if the I/O module is configured in a local or DIO rack.

The additional parameters EN and ENO can be configured.

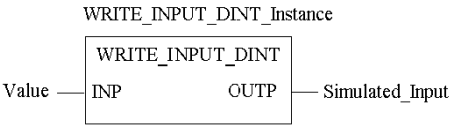
NOTE: Using the WRITE_INPUT_DINT EFB with RIO networks, the behavior in simulation mode could be different from PLC behavior.
For example, set 1 as input and %I2 as output.

- In simulation mode %I2 is set to 1.
- Using a PLC %I2 could be set to 0.

The %I2 is set by the WRITE_INPUT_DINT EFB but overwritten by the value sent by the CRP in the OUT phase of the PLC scan.

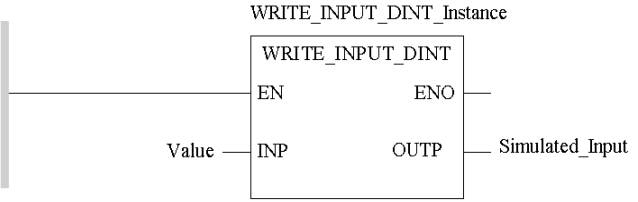
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
WRITE_INPUT_DINT (INP:=Value (*DINT*)
ST Simulated_Input (*DINT*));
```

ST representation

Representation:

```
(*DINT*) Simulated_Input:=WRITE_INPUT_DINT (INP:=Value (*DINT*));
```

Parameter Description

The following table describes the input parameters:

Parameter	Type	Comment
INP	DINT	value to be assigned to the variable

The following table describes the output parameters:

Parameter	Type	Comment
OUTP	DINT	variable to be modified

Chapter 62
WRITE_INPUT_UDINT: Writing Inputs of Type UDINT

Description

Function Description

WRITE_INPUT_UDINT is used to simulate (write) a value at a %ID input. The input is written directly when WRITE_INPUT_UDINT is invoked.

The function block can be used with a real PLC and the PLC simulator.

Restrictions:

- The function block provides access to one variable at a time (elementary data types and not arrays of elementary data types).
- Only variable addresses are allowed, you cannot use declared variables even if they are mapped on a %ID direct address.
- The simulation block WRITE_INPUT_UDINT does not work if the I/O module is configured in a RIO rack but works properly if the I/O module is configured in a local or DIO rack.

The additional parameters EN and ENO can be configured.

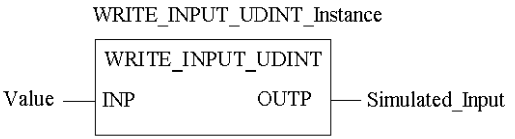
NOTE: Using the WRITE_INPUT_UDINT EFB with RIO networks, the behavior in simulation mode could be different from PLC behavior.
For example, set 1 as input and %I2 as output:

- in simulation mode %I2 is set to 1
- using a PLC %I2 could be set to 0

The %I2 is set by the WRITE_INPUT_UDINT EFB but overwritten by the value sent by the CRP in the OUT phase of the PLC scan.

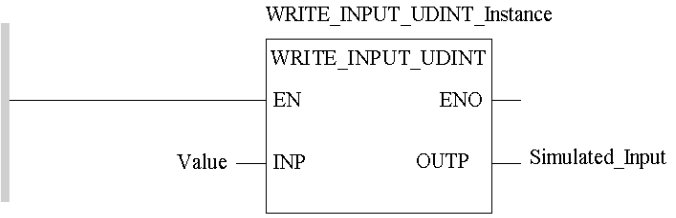
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
WRITE_INPUT_UDINT (INP:=Value (*UDINT*))
ST Simulated_Input (*UDINT*)
```

ST representation

Representation:

```
(*UDINT*) Simulated_Input:=WRITE_INPUT_UDINT (INP:=Value, (*UDINT*));
```

Parameter Description

The following table describes the input parameters:

Parameter	Type	Comment
INP	UDINT	value to be assigned to the variable

The following table describes the output parameters:

Parameter	Type	Comment
OUTP	UDINT	variable to be modified

Chapter 63
WRITE_INPUT_UINT: Writing Inputs of Type UINT

Description

Function Description

WRITE_INPUT_UINT is used to simulate (write) a value at a %ID input. The input is written directly when WRITE_INPUT_UINT is invoked.

The function block can be used with a real PLC and the PLC simulator.

Restrictions:

- The function block provides access to one variable at a time (elementary data types and not arrays of elementary data types).
- Only variable addresses are allowed, you cannot use declared variables even if they are mapped on a %ID direct address.
- The simulation block WRITE_INPUT_UINT does not work if the I/O module is configured in a RIO rack but works properly if the I/O module is configured in a local or DIO rack.

The additional parameters EN and ENO can be configured.

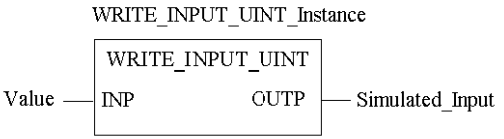
NOTE: Using the WRITE_INPUT_UINT EFB with RIO networks, the behavior in simulation mode could be different from PLC behavior.
For example, set 1 as input and %I2 as output:

- in simulation mode %I2 is set to 1
- using a PLC %I2 could be set to 0

The %I2 is set by the WRITE_INPUT_UINT EFB but overwritten by the value sent by the CRP in the OUT phase of the PLC scan.

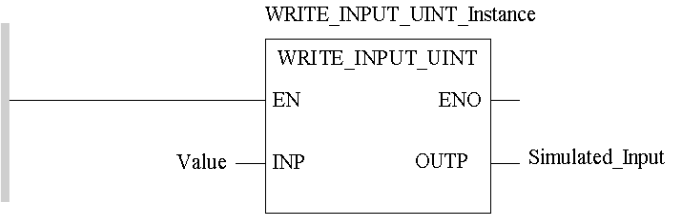
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
WRITE_INPUT_UINT (INP:=Value (*UINT*))
ST Simulated_Input (*UINT*)
```

ST representation

Representation:

```
(*UINT*) Simulated_Input:=WRITE_INPUT_UINT (INP:=Value (*UINT*));
```

Parameter Description

The following table describes the input parameters:

Parameter	Type	Comment
INP	UINT	Value to be assigned to the variable

The following table describes the output parameters:

Parameter	Type	Comment
OUTP	UINT	Variable to be modified

Chapter 64
WRITE_INPUT_EBOOL: Writing Inputs of Type EBOOL

Description

Function Description

WRITE_INPUT_EBOOL is used to simulate (write) a value at a %I input. The input is written directly when WRITE_INPUT_EBOOL is invoked.

The function block can be used with a real PLC and the PLC simulator.

Restrictions

- The function block provides access to one variable at a time (elementary data types and not arrays of elementary data types).
- Only variable addresses are allowed, you cannot use declared variables even if they are mapped on a %I direct address.
- The simulation block WRITE_INPUT_EBOOL does not work if the I/O module is configured in a RIO rack but works properly if the I/O module is configured in a local or DIO rack.

The additional parameters EN and ENO can be configured.

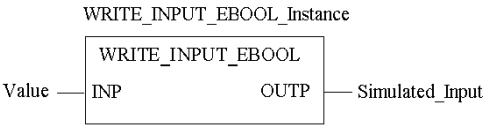
NOTE: Using the WRITE_INPUT_EBOOL EFB with RIO networks, the behavior in simulation mode could be different from PLC behavior.
For example, set 1 as input and %I2 as output.

- In simulation mode %I2 is set to 1.
- Using a PLC %I2 could be set to 0.

The %I2 is set by the WRITE_INPUT_EBOOL EFB but overwritten by the value sent by the CRP in the OUT phase of the PLC scan.

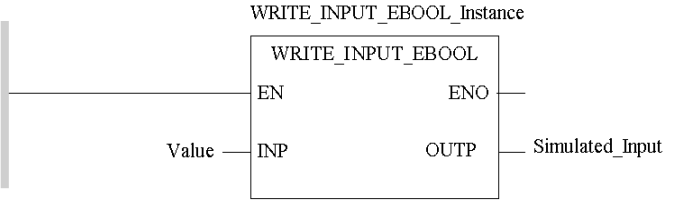
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
WRITE_INPUT_EBOOL (INP:=Value (*BOOL*))
ST Simulated_Input (*EBOOL*)
```

ST representation

Representation:

```
(*EBOOL*) Simulated_Input:=WRITE_INPUT_EBOOL (INP:=Value (*BOOL*));
```

Parameter Description

The following table describes the input parameters:

Parameter	Type	Comment
INP	BOOL	Value to be assigned to the variable

The following table describes the output parameters:

Parameter	Type	Comment
OUTP	EBOOL	variable to be modified

Chapter 65
WRITE_INPUT_INT: Writing Inputs of Type INT

Description

Function Description

WRITE_INPUT_INT is used to simulate (write) a value at a %IW input. The input is written directly when WRITE_INPUT_INT is invoked.

The function block can be used with a real PLC and the PLC simulator.

Restrictions

- The function block provides access to one variable at a time (elementary data types and not arrays of elementary data types).
- Only variable addresses are allowed, you cannot use declared variables even if they are mapped on a %IW direct address.
- The simulation block WRITE_INPUT_INT does not work if the I/O module is configured in a RIO rack but works properly if the I/O module is configured in a local or DIO rack.

The additional parameters EN and ENO can be configured.

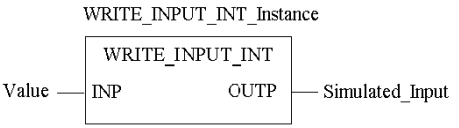
NOTE: Using the WRITE_INPUT_INT EFB with RIO networks, the behavior in simulation mode could be different from PLC behavior.
For example, set 1 as input and %I2 as output.

- In simulation mode %I2 is set to 1.
- Using a PLC %I2 could be set to 0.

The %I2 is set by the WRITE_INPUT_INT EFB but overwritten by the value sent by the CRP in the OUT phase of the PLC scan.

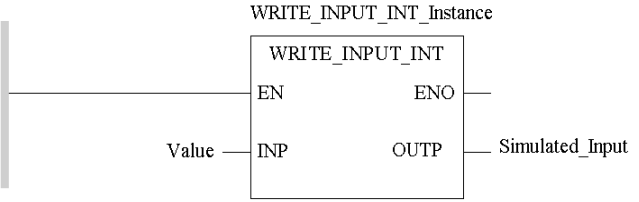
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
WRITE_INPUT_INT (INP:=Value (*INT*))
ST Simulated_Input (*INT*)
```

ST representation

Representation:

```
(*INT*) Simulated_Input:=WRITE_INPUT_INT (INP:=Value (*INT*));
```

Parameter Description

The following table describes the input parameters:

Parameter	Type	Comment
INP	INT	value to be assigned to the variable

The following table describes the output parameters:

Parameter	Type	Comment
OUTP	INT	variable to be modified

Chapter 66
WRITE_INPUT_REAL: Writing Inputs of Type REAL

Description

Function Description

WRITE_INPUT_REAL is used to simulate (write) a value at a %IF input. The input is written directly when WRITE_INPUT_REAL is invoked.

The function block can be used with a real PLC and the PLC simulator.

Restrictions

- The function block provides access to one variable at a time (elementary data types and not arrays of elementary data types).
- Only variable addresses are allowed, you cannot use declared variables even if they are mapped on a %IF direct address.
- The simulation block WRITE_INPUT_REAL does not work if the I/O module is configured in a RIO rack but works properly if the I/O module is configured in a local or DIO rack.

The additional parameters EN and ENO can be configured.

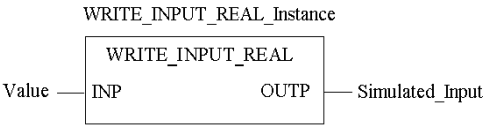
NOTE: Using the WRITE_INPUT_REAL EFB with RIO networks, the behavior in simulation mode could be different from PLC behavior.
For example, set 1 as input and %I2 as output.

- In simulation mode %I2 is set to 1.
- Using a PLC %I2 could be set to 0.

The %I2 is set by the WRITE_INPUT_REAL EFB but overwritten by the value sent by the CRP in the OUT phase of the PLC scan.

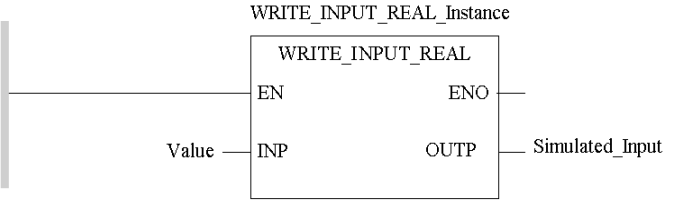
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
WRITE_INPUT_REAL (INP:=Value (*Real*))
ST Simulated_Input (*Real*)
```

ST representation

Representation:

```
(*Real*) Simulated_Input:=WRITE_INPUT_REAL (INP:=Value (*Real*));
```

Parameter Description

The following table describes the input parameters:

Parameter	Type	Comment
INP	REAL	value to be assigned to the variable

The following table describes the output parameters:

Parameter	Type	Comment
OUTP	REAL	variable to be modified

Chapter 67
WRITE_INPUT_AREBOOL_16: Writing Array Inputs of Type EBOOL

Description

Function Description

WRITE_INPUT_AREBOOL_16 is used to simulate (write) a value at a %I input. The input is written directly when WRITE_INPUT_AREBOOL_16 is invoked.

The maximum managed array size is 16

The function block can be used with a real PLC and the PLC simulator.

Restrictions

- Only variable addresses are allowed, you cannot use declared variables even if they are mapped on a %I direct address.

The parameters EN and ENO can be configured.

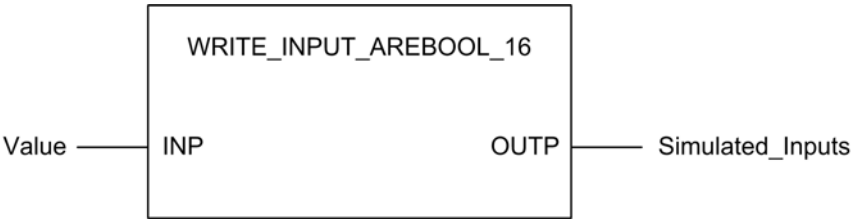
NOTE: When using the WRITE_INPUT_AREBOOL_16 EF with RIO networks, the behavior in simulation mode could be different from PLC behavior.
For example, set 1 as input and %I2:5 as output:

- In simulation mode, %I2...%I6 is set to 1.
- Using a PLC, %I2 may be set to 0.

The %I2...%I6 are set by the WRITE_INPUT_AREBOOL_16 EF but are overwritten by the value sent by the CRP in the OUT phase of the PLC scan.

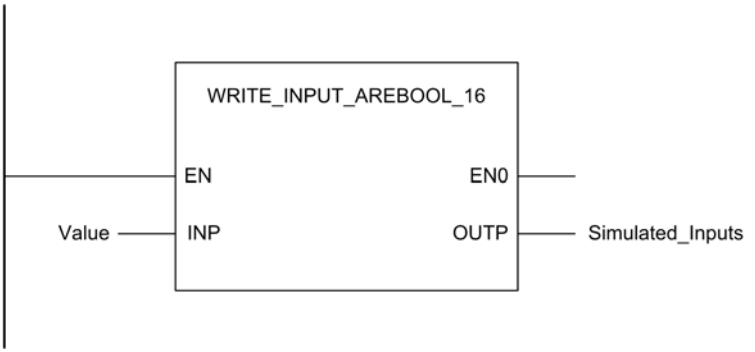
FBD representation

Representation:



LD representation

Representation:



IL representation

Representation:

```
CAL WRITE_INPUT_AREBOOL_16 (INP:=Value, OUTP=>Simulated_Inputs)
```

ST representation

Representation:

```
WRITE_INPUT_AREBOOL_16 (INP:=Value, OUTP=>Simulated_Inputs);
```

Parameter Description

The following table describes the input parameters:

Parameter	Type	Comment
INP	INT	Value to be assigned to the array elements. Each bit is assigned to one array element according to the bit order.

The following table describes the output parameters:

Parameter	Type	Comment
OUTP	ANY_ARRAY_EBOOL	Array of variables to be modified

Appendices

Appendix A
EFB Error Codes and Values

Introduction

The following tables show the error codes and error values created for the EFBs of the IO Management Library.

What Is in This Chapter?

This chapter contains the following topics:

Tables of Error Codes for the IO Management Library

Introduction

The following tables show the error codes and error values created for the EFBs of the IO Management Library.

Analog I/O Configuration

Table of error codes and errors values created for EFBs of the `Analog I/O Configuration` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
I_FILTER	E_EFB_NOT_CONFIGRED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
I_SET	E_EFB_USER_EROR_1	F	-30200	16#8A08	The input IN_REG is not connected with the number of an input word (%IW).
I_SET	E_EFB_USER_EROR_2	F	-30201	16#8A07	The input IN_REG is connected with an invalid number of an input word (%IW).
I_SET	E_EFB_USER_EROR_3	F	-30202	16#8A06	MN_RAW MX_RAW
I_SET	E_EFB_USER_ERROR_4	F	-30203	16#8A05	Unknown value for MN_PHYS
I_SET	E_EFB_USER_ERROR_5	F	-30204	16#8A04	Unknown value for MX_PHYS
I_SET	E_EFB_USER_EROR_11	F	-30210	16#89FE	ST_REG not entered
I_SET	E_EFB_USER_ERROR_12	F	-30211	16#89FD	ST_REG too large
I_SET	E_EFB_USER_ERROR_13	F	-30212	16#89FC	ST_CH not entered
O_FILTER	E_EFB_NOT_CONFIGRED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
O_SET	E_EFB_USER_ERROR_1	F	-30200	16#8A08	The input OUT_REG is not connected with the number of an output word (%MW).
O_SET	E_EFB_USER_ERROR_2	F	-30201	16#8A07	The input OUT_REG is connected with an invalid number of an output word (%MW).
O_SET	E_EFB_USER_ERROR_3	F	-30202	16#8A06	MN_RAW MX_RAW
O_SET	E_EFB_USER_ERROR_4	F	-30203	16#8A05	Unknown value for MN_PHYS
O_SET	E_EFB_USER_ERROR_5	F	-30204	16#8A04	Unknown value for MX_PHYS
O_SET	E_EFB_USER_ERROR_11	F	-30210	16#89FE	ST_REG not entered

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
O_SET	E_EFB_USER_ERROR_12	F	-30211	16#89FD	ST_REG too large
O_SET	E_EFB_USER_ERROR_13	F	-30212	16#89FC	ST_CH not entered

Analog I/O Scaling

Table of error codes and errors values created for EFBs of the `Analog I/O Scaling` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
I_NORM	E_EF_NEG_OVER_ RANGE	F	-30187	16#8A15	Negative overflow
I_NORM	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
I_NORM_WARN	E_EFB_NO_ WARNING_ STAUS_AVAILABLE	F	-30189	16#8A13	Module delivers no warning status
I_NORM_WARN	E_EFB_POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
I_NORM_WARN	E_EFB_NEG_OVER_ RANGE	F	-30187	16#8A15	Negative overflow
I_NORM_WARN	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
I_PHYS	E_EFB_NO_ WARNING_ STATUS_AVAILABLE	F	-30189	16#8A13	Module delivers no warning status
I_PHYS	E_INPUT_VALUE_ OUT_OF_ RANGE	F	-30183	16#8A19	Input value is out of range
I_PHYS	E_EFB_NO_ MEASURING_ RANGE	F	-30185	16#8A17	Internal error
I_PHYS	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
I_PHYS	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
I_PHYS	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
I_PHYS_WARN	E_EFB_NO_ WARNING_ STATUS_AVAILABLE	F	-30189	16#8A13	Module delivers no warning status
I_PHYS_WARN	E_EFB_FILTER_ SQRT_ NOT_AVAIL	F	-30195	16#8A0D	Filter SQRT is not available
I_PHYS_WARN	E_INPUT_VALUE_ OUT_OF_ RANGE	F	-30183	16#8A19	Input value is out of range
I_PHYS_WARN	E_EFB_NO_ MEASURING_RANGE	F	-30185	16#8A17	Internal error
I_PHYS_WARN	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
I_PHYS_WARN	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
I_PHYS_WARN	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
I_RAW	E_EFB_ OUT_OF_ RANGE	F	-30192	16#8A10	Internal error: EFB has detected a violation e.g. write exceeds %MW (4x) boundaries
I_RAW	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
I_RAWSIM	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
I_SCALE	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
I_SCALE	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
I_SCALE	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
I_SCALE_WARN	E_EFB_NO_WARNING_ STATUS_AVAILABLE	F	-30189	16#8A13	Module delivers no warning status
I_SCALE_WARN	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
I_SCALE_WARN	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
I_SCALE_WARN	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
O_NORM	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
O_NORM	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
O_NORM	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
O_NORM_ WARN	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
O_NORM_ WARN	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
O_NORM_ WARN	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
O_PHYS	E_EFB_NO_ MEASURING_RANGE	F	-30185	16#8A17	Internal error
O_PHYS	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
O_PHYS	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
O_PHYS	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
O_PHYS_WARN	E_EFB_NO_ MEASURING_RANGE	F	-30185	16#8A17	Internal error
O_PHYS_WARN	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
O_PHYS_WARN	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
O_PHYS_WARN	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
O_RAW	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
O_RAW	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
O_SCALE	E_INPUT_VALUE_ OUT-_OF_RANGE	F	-30183	16#8A19	Input value is out of range
O_SCALE	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
O_SCALE	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
O_SCALE	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
O_SCALE_ WARN	E_INPUT_VALUE_ OUT_ OF_RANGE	F	-30183	16#8A19	Input value is out of range
O_SCALE_ WARN	E_EFB_ POS_ OVER_ RANGE	F	-30186	16#8A16	Positive overflow
O_SCALE_ WARN	E_EFB_ NEG_ OVER_ RANGE	F	-30187	16#8A15	Negative overflow
O_SCALE_ WARN	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration

Immediate I/O

Table of error codes and errors values created for EFBs of the `Immediate I/O` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
IMIO_IN	-	F	0000	0000	Operation OK
IMIO_IN	-	F	8193	2001	invalid operation type (e.g. the I/O module addressed is not an input module)
IMIO_IN	-	F	8194	2002	Invalid rack or slot number (I/O map in the configurator contains no module entry for this slot)
IMIO_IN	-	F	8195	2003	invalid slot number
IMIO_IN	-	F	-4095	F001	Module not OK
IMIO_OUT	-	F	0000	0000	Operation OK
IMIO_OUT	-	F	8193	2001	invalid operation type (e.g. the I/O module addressed is not an input module)
IMIO_OUT	-	F	8194	2002	Invalid rack or slot number (I/O map in the configurator contains no module entry for this slot)
IMIO_OUT	-	F	8195	2003	invalid slot number
IMIO_OUT	-	F	-4095	F001	Module not OK

Quantum I/O Configuration

Table of error codes and errors values created for EFBs of the `Quantum I/O Configuration` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
ACI030	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
ACI040	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
ACI040	E_EFB_CURRENT_- MODE_ NOT_ALLOWED	F	-30197	16#8A0B	EFB error: Current mode is not allowed
ACO020	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
ACO130	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
ACO130	E_EFB_CURRENT_- MODE_NOT_ALLOWED	F	-30197	16#8A0B	EFB error: Current mode is not allowed
AII330	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
AII330	E_EFB_ILLEGAL_ CONFIG_DATA	F	-30198	16#8A0A	EFB error: Illegal configuration data
AII33010	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
AII33010	E_EFB_CURRENT_- MODE_NOT_ALLOWED	F	-30197	16#8A0B	EFB error: Current mode is not allowed
AIO330	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
AIO330	E_EFB_CURRENT_- MODE_NOT_ALLOWED	F	-30197	16#8A0B	EFB error: Current mode is not allowed
AMM090	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
ARI030	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
ARI030	E_EFB_ILLEGAL_ CONFIG_DATA	F	-30198	16#8A0A	EFB error: Illegal configuration data
ATI030	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
AVI030	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
AVO020	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
DROP	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
ERT_854_10	ES_WRONG_SLOT	F	20480	16#5000	-
ERT_854_10	E_WRONG_SLOT	F	-30215	16#89F9	Defined as E_EFB_USER_ERROR_16
ERT_854_10	ES_HEALTHBIT	F	24576	16#6000	-
ERT_854_10	E_HEALTHBIT	F	-30216	16#89F8	Defined as E_EFB_USER_ERROR_17
ERT_854_10	ES_TIMEOUT	F	32768	16#8000	-
ERT_854_10	E_TIMEOUT	F	-30210	16#89FE	Defined as E_EFB_USER_ERROR_11
ERT_854_10	E_ERT_BASIC - values	F	-30199	16#8A09	Defined as E_EFB_USER_ERROR_1 + 1
ERT_854_10	E_WRONG_ANSW	F	-30211	16#89FD	Defined as E_EFB_USER_ERROR_12
ERT_854_10	ES_CBUF_OFLOW	F	28672	16#7000	-
ERT_854_10	E_CBUF_OFLOW	F	-30217	16#89F7	Defined as E_EFB_USER_ERROR_18
ERT_854_10	ES_WRONG_PAKET	F	8192	16#2000	-
ERT_854_10	E_WRONG_PAKET	F	-30212	16#89FC	Defined as E_EFB_USER_ERROR_13
ERT_854_10	ES_WRONG_FELD	F	12288	16#3000	-
ERT_854_10	E_WRONG_FELD	F	-30213	16#89FB	Defined as E_EFB_USER_ERROR_14
QUANTUM	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
QUANTUM	E_EFB_UNKNOWN_- DROP	F	-30190	16#8A12	Unknown drop / No Quantum traffic cop
XBE	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration
XBE	E_EFB_UNKNOWN_- DROP	F	-30190	16#8A12	Unknown drop / No Quantum traffic cop
XDROP	E_EFB_NOT_ CONFIGURED	F	-30188	16#8A14	EFB configuration does not match hardware configuration

NOTE: For details about ERT_854_10, please refer to the ERT_854_10 description *(see page 256)* in the IO Management Library.

Common Floating Point Errors

Introduction

The following table shows the common error codes and error values created for floating point errors. These error information are displayed in the Diagnostic Viewer and the error code values are written in %SW125 *(see Unity Pro, System Bits and Words, Reference Manual)*.

Common Floating Point Errors

Table of common floating point errors

Error codes	Error value in Dec	Error value in Hex	Error description
FP_ERROR	-30150	16#8A3A	Base value (not apearing as an error value)
E_FP_STATUS_FAILED_IE	-30151	16#8A39	Illegal floating point operation
E_FP_STATUS_FAILED_DE	-30152	16#8A38	Operand is denormalized - not a valid REAL number
E_FP_STATUS_FAILED_ZE	-30154	16#8A36	Illegal divide by zero
E_FP_STATUS_FAILED_ZE_IE	-30155	16#8A35	Illegal floating point operation / Divide by zero
E_FP_STATUS_FAILED_OE	-30158	16#8A32	Floating point overflow
E_FP_STATUS_FAILED_OE_IE	-30159	16#8A31	Illegal floating point operation / Overflow
E_FP_STATUS_FAILED_OE_ZE	-30162	16#8A2E	Floating point overflow / Divide by zero
E_FP_STATUS_FAILED_OE_ZE_IE	-30163	16#8A2D	Illegal floating point operation / Overflow / Divide by zero
E_FP_NOT_COMPARABLE	-30166	16#8A2A	Internal error

Error Codes of EFBs with STATUS parameter

Form of the Function Error Code

STATUS parameter error codes appear as **Mmss**, where:

- **M** is the high code
- **m** is the low code
- **ss** is a subcode

Common Error Codes

Hexadecimal error codes description:

Hex. Error Code	Description
1001	Abort by user.
1002	Warm Start initiated abort.
2001	An operation type that is not supported has been specified in the control block.
2002	One or more control block parameters were modified while the MSTR element was active (this only applies to operations which require several cycles for completion). Control block parameters may only be modified in inactive MSTR components.
2003	Invalid value in the length field of the control block.
2004	Invalid value in the offset field of the control block.
2005	Invalid value in the length and offset fields of the control block.
2006	Unauthorized data field on slave.
2007	Unauthorized network field on slave.
2008	Unauthorized network routing path on slave.
2009	Routing path equivalent to their own address.
200A	Attempt to get more global data words than available.
200B	Peer cop conflict on WR/RD global.
200C	Bad pattern for change address request.
200D	Bad address for change address request.
200E	The control block or data buffer is not assigned, or (incorrect size, parts of the control block or data buffer are located outside of the %MW (4x) range.
200F	Space for response in data buffer is too small.
2010	Control buffer length invalid.
2011	Invalid parameter.
2012	Syntax error in “rail.slot.chan” string.
2013	Missing module or module failure.
2015	No channel data (channel out of bound).

Hex. Error Code	Description
2016	Abort on timeout.
2017	Invalid task context.
2018	Ethernet security system service error.
30ss	Exceptional response by the Modbus slave with specific **ss** exception code *(see page 325)*.
31ss	Exceptional Unity protocol error response by the Modbus slave with specific **ss** error code *(see page 326)*.
32ss	Exceptional Unity protocol IO request error acknowledgement by the Modbus slave with specific **ss** error code *(see page 327)*.
33ss	UNI-TE specific report.
34ss	Generic communication report *(see page 328)* (correspond to Communication Report field of Premium/M340 EF Management Parameters).
35ss	Generic Operation Report in case of correct exchange *(see page 328)* (correspond to Operation Report field of Premium/M340 EF Management Parameters when Communication Report = 16#00).
36ss	Generic Operation Report in case of refused message *(see page 328)* (correspond to Operation Report field of Premium/M340 EF Management Parameters when Communication Report = 16#FF).
4001	Inconsistent response by the Modbus slave.
4002	Inconsistent Modbus Umas response.
4003	Inconsistent UNI-TE response (depending on the module)
4004	Read request for the status words not accepted by the module channel.
4005	Command parameters not accepted by the module channel.
4006	Adjustment parameters not accepted by the module.
5mss	TCP/IP Ethernet specific error codes *(see page 330)*
6mss	Modbus Plus specific routing path error *(see page 334)* The subfield m shows where the error occurred (a 0 value means local node, 2 means 2nd device in route, and so on).
7mss	SY/MAX specific error codes *(see page 335)*
F001	Wrong destination node was specified for the MSTR operation. Referenced S985 option not present or in reset mode.
F002	Component not fully initialized

Modbus Specific Exception Function Codes (30ss)

This table lists the ss hexadecimal valuse in 30ss error codes:

Hex. Error Code	Description
3001	Slave does not support requested operation.
3002	Non-existing slave registers were requested.
3003	An unauthorized data value was requested.
3004	Slave device failure
3005	Slave has accepted a lengthy program command.
3006	Function cannot currently be carried out: lengthy command running.
3007	Slave has rejected lengthy program command.
300A	Gateway unable to allocate an internal communication path
300B	No response obtained from target device

The ss value corresponds to the Modbus Exception code returned by the Modbus slave device in case of error (second byte of the Modbus exception PDU):

- exception-function_code = request function code + 0x80: 1 byte
- exception_code: 1 byte (returned as ss in 16#30ss error code

Details of STATUS error code from 31ss to 36ss

Unity Protocol Error Codes (31ss)

This table lists the ss hexadecimal value in 31ss error code:

Hex. Error Code	Description
3100	Generic Unity Protocol error.
3180	Generic communication error.
3181	PLC is reserved by someone else.
3182	You must reserve the PLC.
3183	Unknown request or subcode.
3184	Unknown object (for example: %Z not implemented).
3185	The response could not be built.
3186	Request has invalid parameters (for example: the request is badly built, or has few/many parameters or has a bad Csa command, and so on.)
3187	Bad sequence (for example: EndDownload executed before BeginDownload).
3188	Too big response for the available buffer.
3189	Module not configured (may be bad address).
318A	Action is not permitted on this object.
318B	Busy state: the previous operation is still working, or all internal resources are busy for the IO request, or too much parallel upload, and so on.
3190	Generic error: something wrong in application.
3191	Access violation: write read-only block, or read-only variable, or download while memory protected, and so on.
3192	Object not accessible because it is in use.
3193	Out of bounds: exceeds the range of %MW, or too many breakpoints, or call stack too large, and so on.
3194	Invalid length.
3195	Reference to a non-existent resource or task, or var address is not in the DFB data area, and so on.
3196	The object or resource is already defined. For example: trying to start an already started, or a breakpoint-ID already used, and so on.
3197	Data inconsistent or data state not allowed. For example: wrong data or bad value when writing an object.
3198	The object exists but is not initialized.
3199	Channel out of bounds in an IO request.
319A	Request not yet implemented.

Hex. Error Code	Description
31A0	Incompatible application, wrong target or platform.
31A1	Signature check failed.
31A2	Wrong PCMCIA memory configuration.
31B0	PLC is not in the right mode: download with PLC in RUN mode, or debug while PLC in NOCONF, try to step a task , not break-pointed, upload operation aborted by download or OnlineChange, and so on.
31B1	Mode can not be changed: an I/O forces premium to stop.
31B2	An internal timeout occurred.
31B3	The watchdog time is elapsed.
31FF	Undefined general error.

Unity Protocol IO Request Error Acknowledgement (32ss)

This tables lists the ss hexadecimal valuse in 32ss error code:

Hex. Error Code	Description
3202	Exchange error.
3207	Another explicit exchange is in progress.
3209	Operation is impossible.
320A	Data is rejected by the IOB.
320B	Writing is not authorized.
320C	Maximum number of exchanges.
3284	Unknown object.
3286	Invalid read buffer.
328A	Unknown or invalid action.
328B	All the buffers are in use.
3293	Object out of range.
3297	Object value forbidden (write operations only).
3299	Channel out of range.

Generic Communication Report (34ss)

This table lists the ss hexadecimal value in 34ss error code:

Hex. Error Code	Description
3401	Exchange stop on timeout.
3402	Exchange stop on user request (CANCEL).
3403	Incorrect address format.
3404	Incorrect destination address
3405	Incorrect management parameter format.
3406	Incorrect specific parameters.
3407	Error detected in sending to the destination.
3409	Reserved.
340A	Insufficient receive buffer size.
340B	No processor system resources.
340C	Incorrect exchange number.
340D	No telegram received.
340E	Incorrect length.
340F	Telegram service not configured.
3410	Network module missing.
3411	Request missing.
3412	Application server already active.
3413	UNI-TE V2 transaction number incorrect.

The ss value corresponds to the Communication Report code *(see Unity Pro, Communication, Block Library)* returned by existing communications EFs on Premium/Atrium/Mxxx platforms.

Generic Operation Report (35ss and 36ss)

This report byte is specific to each function and specifies the result of the operation on the remote application.

Generic operation report in case of correct exchange is coded under the form 16#35ss. If the message is refused by the remote device, the report will have the form 16#36ss.

This table lists the ss hexadecimal value in 35ss error code:

Hex. Error Code	Description
3501	Request not processed.
3502	Incorrect response.

The ss value in 35ss codes corresponds to operation report field of Premium/M340 EF Management Parameters when Communication Report *(see Unity Pro, Communication, Block Library)*= 16#00.

This table lists the ss hexadecimal value in 36ss error code:

Hex. Error Code	Description
3601	No resources towards the processor.
3602	No line resources.
3603	No device or device without resources.
3604	Line error.
3605	Length error.
3606	Faulty communication channel.
3607	Addressing error.
3608	Application error.
360B	No system resources.
360C	Communication function not active.
360D	Destination missing.
360F	Intra-station routing problem or channel not configured.
3611	Address format not managed.
3612	No destination resources.
3614	Non-operational connection (example: Ethernet TCP/IP).
3615	No resource on the local channel.
3616	Access not authorized (example: Ethernet TCP/IP).
3617	Inconsistent network configuration (example: Ethernet TCP/IP).
3618	Connection temporarily unavailable.
3621	Application server stopped.
3630	Transmission error.

The ss value in 36ss codes corresponds to operation report field of Premium/M340 EF Management Parameters when Communication Report *(see Unity Pro, Communication, Block Library)*= 16#FF.

EFB TCP/IP Ethernet Error Codes

TCP/IP Ethernet Network Specific Error Codes (5mss)

Hexadecimal error codes for TCP/IP Ethernet (Ethernet errors are managed by the Ethernet modules or the Ethernet coprocessor except for error code hex 5050):

Hex. Error Code	Meaning
5001	Inconsistent response by the network
5004	Interrupted system call
5005	I/O error
5006	No such address
5009	Socket descriptor is invalid
500C	Not enough memory
500D	Authorization denied
5011	Entry exists
5016	Argument is invalid
5017	Internal table has run out of space
5020	Connection is lost
5023	This operation was blocked and the socket is non-blocking
5024	The socket is non-blocking and the connection can not be closed
5025	The socket is non-blocking and a previous connection attempt has not been finalized
5026	Socket operation performed on a non-socket
5027	The destination address is not valid
5028	Message too long
5029	Wrong type of protocol for the socket
502A	Protocol not available
502B	Protocol not supported
502C	Socket type not supported
502D	Operation not supported on a socket
502E	Protocol family not supported
502F	Address family not supported
5030	Address already in use
5031	Address not available
5032	Network is out of order
5033	Network is unreachable

Hex. Error Code	Meaning
5034	Network dropped connection on reset
5035	Connection aborted by the peer
5036	Connection reset by the peer
5037	An internal buffer is required but can not be assigned
5038	Socket already connected
5039	Socket not connected
503A	Cannot send after socket shutdown
503B	Too many references, cannot splice
503C	Connection timed out (see note below)
503D	Connection refused
5040	Host is out of order
5041	The destination host could not be reached from this node
5042	Directory not empty
5046	NI_INIT returned '-1'
5047	MTU is not valid
5048	Hardware length is not valid
5049	Route specified cannot be found
504A	Collision in Select call: these conditions have already been selected by another task
504B	Task ID is invalid
5050	No network resource
5051	Length error
5052	Addressing error
5053	Application error
5054	Client can not process the request
5055	No network resource
5056	Non-operational TCP connection
5057	Incoherent configuration
51ss	SMTP service error codes *(see page 332)*
53ss	Modbus client service error codes *(see page 332)*

NOTE:

- Error code hex 5055 can occur before a hex 503C error.
- No remote device takes precedence over a timeout.

SMTP Service Error Codes (51ss)

Hexadecimal SMTP service error codes:

Hex. Error Code	Meaning
5100	Internal error
5101	SMTP component not operational
5102	Mail header not configured
5103	Invalid mail header value
5104	Cannot connect to the SMTP server
5105	Error in transmitting content of email body to the SMTP server
5106	Closing SMTP connection with the server returned an error
5107	SMTP **HELO** request failed
5108	SMTP **MAIL** request failed. SMTP server may require authentication
5109	SMTP **RCPT** request failed
510A	No recipient has been accepted by the SMTP server
510B	SMTP **DATA** request failed
510C	Send email request contains an invalid length
510D	Authentication failed
510E	A reset component request has been received while the connection was open

Modbus Client Service Error Codes (53ss)

Hexadecimal Modbus client service error codes:

Hex. Error Code	Meaning
5300	Not used (reserved for future use)
5301	The component has run out of resources
5302	The IP address provided is not appropriate. For example: 0.0.0.0, or broadcast address, or multicast address, etc.
5303	The transaction has timed-out. The remote server accepted a request and has not answered within 2 minutes.
5304	All connections are currently used
5305	Access denied
5306	Network can not be reached
5307	Host is down
5308	Network dropped connection on reset

Hex. Error Code	Meaning
5309	Network is down
530A	Connection refused
530B	Connection timed out
530C	Wrong MBAP header

Quantum EFB Modbus Plus Error Codes

Modbus Plus Specific Error Codes (6mss)

NOTE: The **m** field in **6mss** error code is an `Index` in the routing information that shows where an error has been detected. **m** = 0 means that the error has been detected in the local node, **m** = 2 means it has been detected in the second device in the route, etc.

Hex. Error Code	Description
6m01	No response reception
6m02	Access to program denied
6m03	Node out of service and unable to communicate
6m04	Unusual response received
6m05	Router-node data path busy
6m06	Slave out of order
6m07	Wrong destination address
6m08	Unauthorized node type in routing path
6m10	Slave has rejected the command
6m20	Slave has lost an activated transaction
6m40	Unexpected master output path received
6m80	Unexpected response received
F001	Wrong destination node was specified for the `MSTR` operation

Quantum EFB SY/MAX Specific Error Codes

SY/MAX-Specific Error Codes

When using SY/MAX Ethernet, three additional types of errors may appear in the `CONTROL[1]` register of the control block.

The error codes have the following meaning:

- 71xx Error: Errors found by the SY/MAX remote device
- 72xx Error: Errors found by the server
- 73xx Error: Errors found by the Quantum translator

SY/MAX-Specific Hexadecimal Error Codes

SY/MAX-specific hexadecimal error codes:

Hex. Error Code	Description
7101	Invalid opcode found by the SY/MAX remote device
7103	Invalid address found by the SY/MAX remote device
7109	Attempt to write to a write protected register found by the SY/MAX remote device
F710	Receiver overflow found by the SY/MAX remote device
7110	Invalid length found by the SY/MAX remote device
7111	Remote device not active, no connection (occurs when retry attempts and time-out have been used up), found by the SY/MAX remote device
7113	Invalid parameter in a read operation found by the SY/MAX remote device
711D	Invalid route found by the SY/MAX remote device
7149	Invalid parameter in a write operation found by the SY/MAX remote device
714B	Invalid drop number found by the SY/MAX remote device
7101	Invalid opcode found by the SY/MAX server
7203	Invalid address found by the SY/MAX server
7209	Attempt to write to a write protected register found by the SY/MAX server
F720	Receiver overflow found by the SY/MAX server
7210	Invalid length found by the SY/MAX server
7211	Remote device not active, no connection (occurs when retry attempts and time-out have been used up), found by the SY/MAX server
7213	Invalid parameter in a read operation found by the SY/MAX server
721D	Invalid route found by the SY/MAX server
7249	Invalid parameter in a write operation found by the SY/MAX server
724B	Invalid drop number found by the SY/MAX server

Hex. Error Code	Description
7301	Invalid opcode in an MSTR block request from the Quantum translator
7303	Read/Write QSE module status (200 route address out of range)
7309	Attempt to write to a write protected register when a status write is carried out (200 route)
731D	Invalid route found by the Quantum translator. Valid routes: ● dest_drop, 0xFF ● 200, dest_drop, 0xFF ● 100+drop, dest_drop, 0xFF ● All other routing values produce an error
734B	One of the following errors occurred: ● No CTE (configuration extension table) has been configured ● No CTE table entry has been made for the QSE model slot number ● No valid drop has been specified ● The QSE module has not been reset after the creation of the CTE. Note: After writing and configuring the CTE and downloading to the QSE module, the QSE module must be reset for the modifications to become effective. ● When using an MSTR instruction no valid slot or drop has been specified

Glossary

!

%IW

According to the CEI standard, %IW indicates a language object of type analog IN.

%MW

According to the CEI standard, %MW indicates a language object of type memory word.

A

ANL_IN

ANL_IN is an internal data type that is used between Quantum specific FFBs. For instance, an ACI040 FFB provides the channels structures of the module as ANL_IN types, which are used as input of an I_FILTER FFB. This data type must not be used outside the Quantum specific FFBs.

Elements of the DDT:

Name	Type	Comment
valuePtr	UDINT	3x or 4x raw value register
rawControl	BYTE	Control Byte(internal use only)
rawSpecific	BYTE	Specific Byte (internal use only)
offset	INT	Offset Value
range	WORD	Input range (resolution)
channel	BYTE	Input Channel number
statusMode	BYTE	Status Mode (internal use only)
statusPtr	UDINT	Identifies high byte or low byte of status register
warnCode	BYTE	Warning Code (internal use only)

ANL_OUT

ANL_OUT is an internal data type that is used between Quantum specific FFBs. For instance, an ACO130 FFB provides the channel structures of the module as ANL_OUT types, which are used as input of an O_PHYS FFB. This data type must not be used outside the Quantum specific FFBs.

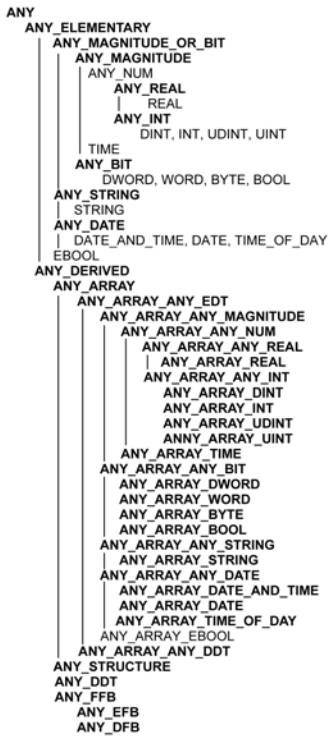
Elements of the DDT:

Name	Type	Comment
valuePtr	UDINT	3x or 4x raw value register
rawControl	BYTE	Control Byte(internal use only)
rawSpecific	BYTE	Specific Byte (internal use only)
offset	INT	Offset Value
range	WORD	Input range (resolution)
channel	BYTE	Input Channel number
statusMode	BYTE	Status Mode (internal use only)
statusPtr	UDINT	Identifies high byte or low byte of status register
warnCode	BYTE	Warning Code (internal use only)

ANY

There is a hierarchy among the various data types. In the DFBs, it is sometimes possible to declare variables that can contain several types of values. In that case we use ANY_xxx types.

The figure below describes this hierarchical structure:



ARRAY

An ARRAY is a table containing elements of a single type.

The syntax is as follows: `ARRAY [<limits>] OF <Type>`

Example:

`ARRAY [1..2] OF BOOL` is a one-dimensional table with two elements of type `BOOL`.

`ARRAY [1..10, 1..20] OF INT` is a two-dimensional table with 10x20 elements of type `INT`.

B

BOOL

`BOOL` is the abbreviation for the Boolean type. This is the basic data type in computing. A `BOOL` variable can have either of the following two values: 0 (FALSE) or 1 (TRUE).

A bit extracted from a word is of type `BOOL`, for example: `%MW10.4`.

BYTE

When 8 bits are grouped together, they are called a BYTE. You can enter a BYTE either in binary mode or in base 8.

The BYTE type is encoded in an 8 bit format which, in hexadecimal format, ranges from 16#00 to 16#FF.

D

Device DDT (DDDT)

A Device DDT is a DDT predefined by the manufacturer and not modifiable by user. It contains the I/O language elements of an I/O module.

DINT

DINT is the abbreviation of Double INTeger (encoded in 32 bits).

The upper/lower limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

-2147483648, 2147483647, 16#FFFFFFFF.

E

EBOOL

EBOOL is the abbreviation of Extended BOOLean. An EBOOL type has a value (0 (FALSE) or 1 (TRUE), but also rising or falling edges and forcing functions.

An EBOOL variable occupies one byte in memory.

The byte contains the following information:

- one bit for the value;
- one bit for the history (whenever the object changes state, the value is copied to the history bit);
- one bit for forcing (equal to 0 if the object is not forced, or 1 if the bit is forced).

The default value of each bit is 0 (FALSE).

EN

EN stands for **EN**able; it is an optional block input. When the EN input is enabled, an ENO output is set automatically.

If EN = 0, the block is not enabled; its internal program is not executed, and ENO is set to 0.

If EN = 1, the block's internal program is run and ENO is set to 1. If an error occurs, ENO is set to 0.

If the EN input is not connected, it is set automatically to 1.

ENO

ENO stands for **E**rror **NO**tification; this is the output associated with the optional input EN.

If ENO is set to 0 (because EN = 0 or in case of an execution error):

- the status of the function block outputs remains the same as it was during the previous scanning cycle that executed correctly;
- the output(s) of the function, as well as the procedures, are set to "0".

I

INT

INT is the abbreviation of single INTeger (encoded in 16 bits).

The upper/lower limits are as follows: -(2 to the power of 15) to (2 to the power of 15) - 1.

Example:

-32768, 32767, 2#1111110001001001, 16#9FA4.

IODDT

IODDT is the abbreviation of Input/Output Derived Data Type.

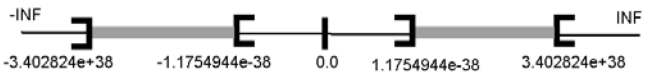
The term IODDT indicates a structured data type representing a module or a channel of a PLC module. Each expert module has its own IODDTs.

R

REAL

The REAL type is encoded in a 32 bit format.

The possible value ranges are shown in the figure below:



When a result is:

- between -1,175494e-38 and 1,175494e-38, it is considered to be a DEN;
- less than -3.402824e+38, the symbol -INF (for - infinity) is displayed;
- greater than +3.402824e+38, the symbol INF (for + infinity) is displayed;
- undefined (square root of a negative number), the symbolNAN is displayed.

NOTE: The IEC 559 standard defines two classes of NAN: the silent NAN (QNAN) and the signaling NAN (SNAN). A QNAN is a NAN with a most significant fraction bit while an SNAN is a NAN without a most significant fraction bit (bit number 22). QNANs can be propagated via most arithmetic operations without throwing an exception. As for SNANs, they generally indicate n invalid operation when they are used as operands in arithmetic operations (see %SW17 and %S18).

NOTE: When a DEN (non-standardized number) is used as an operand, the result is not significant.

S

STRING

A STRING variable is a series of ASCII characters. The maximum length of a string is 65,534 characters.

T

TOPO_ADDR_TYPE

This predefined type is used as an output for the READ_TOPO_ADDR function. This is an ARRAY[0..4] OF Int. You can find it in the library, in the same family as the EFs that use it.

U

UDINT

UDINT is the abbreviation of Unsigned Double INTeger (encoded in 32 bits). The upper/lower limits are as follows: 0 to (2 to the power of 32) - 1.

Example:

0, 4294967295, 2#11111111111111111111111111111111, 8#37777777777, 16#FFFFFFFF.

UINT

UINT is the abbreviation of the Unsigned INTeger format (encoded in 16 bits). The upper/lower limits are as follows: 0 to (2 to the power of 16) - 1.

Example:

0, 65535, 2#1111111111111111, 8#177777, 16#FFFF.

W

WORD

The type WORD is encoded in a 16 bit format and is used to perform processing on series of bits.

This table shows the upper/lower limits of each of the bases that can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Examples of representation

Data	Representation in one of the bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index

I

O

Q

R

S

T

W

X