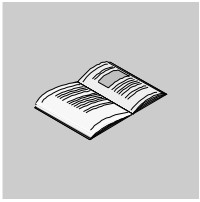
Concept
IEC Block Library
Part: COMM

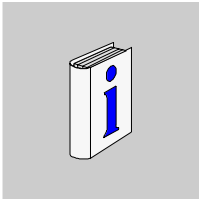
840 USE 504 00 eng Version 2.6

33002210.00

Table of Contents



About the Book



At a Glance

Document Scope

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

Validity Note

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

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

Related Documents

Title of Documentation	Reference Number
Concept Installation Instructions	840 USE 502 00
Concept User Manual	840 USE 503 00
Concept EFB User Manual	840 USE 505 00
Concept LL984 Block Library	840 USE 506 00
Modbus Plus network user manual	890 USE 100 00
Modbus Plus Bridge / Multiplexer User's Guide	GM-BM85-001
Quantum Ethernet TCI/IP module User's Guide	890 USE 107 00
XMIT-IEC User Manual	840 USE 499 00

User Comments

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

General information about the COMM module library

I

Overview

Introduction

This section contains general information about the COMM module library.

What's in this Part?

This part contains the following chapters:

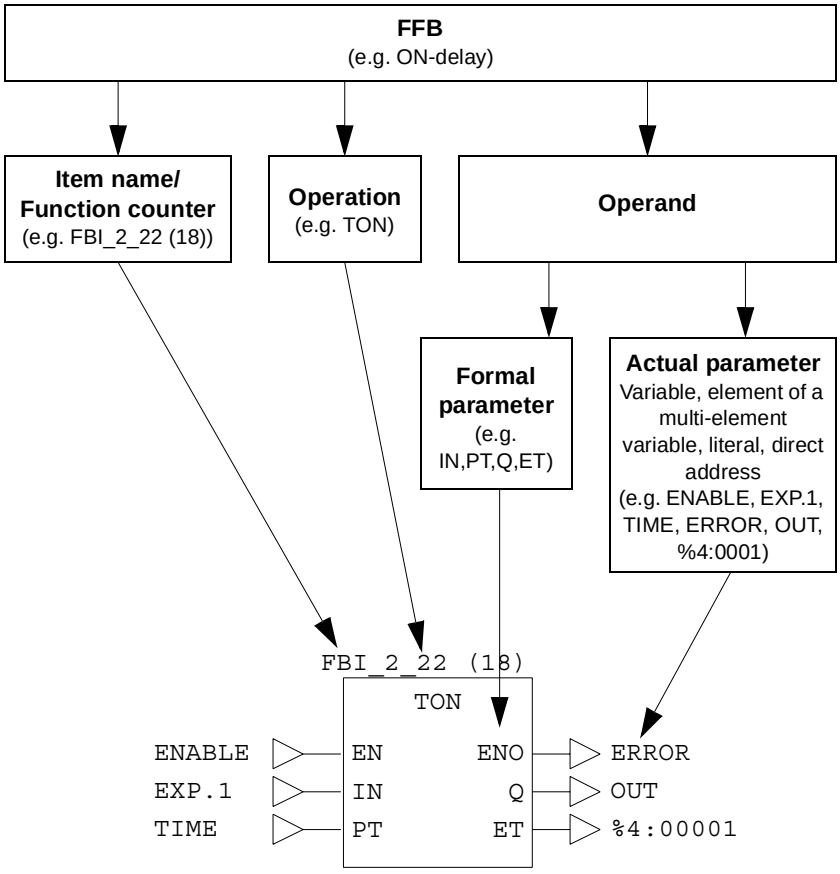
Parameterizing functions and function blocks

1

Parameterizing functions and function blocks

General

Each FFB consists of an operation, the operands needed for the operation and an instance name or function counter.



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

The formal parameter holds the place for an operand. During parameterization, an actual parameter is assigned to the formal parameter.

The actual parameter can be a variable, a multi-element variable, an element of a multi-element variable, a literal or a direct address.

Conditional/ unconditional calls

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

- Displayed EN
  conditional calls (the FFB is only processed if EN = 1)
- EN not displayed
  unconditional calls (FFB is always processed)

Note: If the EN input is not parameterized, it must be disabled. Any input pin that is not parameterized is automatically assigned a "0" value. Therefore, the FFB should never be processed.

Calling functions and function blocks in IL and ST

Information on calling functions and function blocks in IL (Instruction List) and ST (Structured Text) can be found in the relevant chapters of the user manual.

EFB descriptions

II

Overview

Introduction

These EFB descriptions are arranged in alphabetical order.

Note: The number of inputs of some EFBs can be increased to a maximum of 32 by changing the size of the FFB symbol vertically. Information on which EFBs have this capability is given in the descriptions of the individual EFBs.

What's in this Part?

This part contains the following chapters:

CREAD_REG: Continuous register reading

2

Overview

Introduction

This chapter describes the CREAD_REG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block reads the register area continuously. It reads data from an addressed node via Modbus Plus, TCP/IP-Ethernet or SY/MAX-Ethernet.

EN and ENO can be projected as additional parameters.

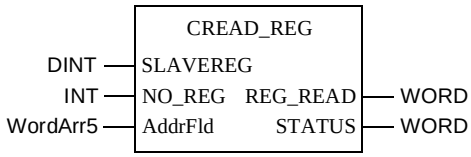
Note: When programming a CREAD_REG function, you must be familiar with the routing procedures used by your network. Modbus Plus routing path structures are described in detail in "Modbus Plus Network Planning and Installation Guide". If TCP/IP or SY/MAX Ethernet is

Note: For technical reasons, this function block does not allow the use of programming languages ST and IL .

Representation

Symbol

Block representation:



Parameter description

Description of parameters:

Parameter	Data type	Meaning
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be read from
NO_REG	INT	Number of registers to be read from slave
AddrFld	WordArr5	Data structure describing the Modbus Plus-address, TCI/IP address or SY/MAX-IP address.
REG_READ	WORD	First 4x area register for read values
STATUS	WORD	Error code, see *Runtime errors, p. 95*

Elementary description for WordArr5 in Modbus Plus

Elementary description for WordArr5 in Modbus Plus:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: Routing register 1 is used for address specification (routing path addresses one of five) of the destination node during network transfer. The last byte in the routing path that is not zero is the destination node. High value byte: Slot of the network adapter module (NOM), if any (only Quantum).
WordArr5[2]	WORD	Routing register 2
WordArr5[3]	WORD	Routing register 3
WordArr5[4]	WORD	Routing register 4
WordArr5[5]	WORD	Routing register 5

Elementary description for WordArr5 with TCP/IP EtherNet

Elementary description for WordArr5 with TCP/IP EtherNet

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slot of the NOE module
WordArr5[2]	WORD	Byte 4 (MSB) of the 32-bit destination IP address
WordArr5[3]	WORD	Byte 3 of the 32-bit destination IP address
WordArr5[4]	WORD	Byte 2 of the 32-bit destination IP address
WordArr5[5]	WORD	Byte 1 (LSB) of the 32-bit destination IP address

Elementary description for WordArr5 with SYMAX EtherNet

Elementary description for WordArr5 with SYMAX EtherNet

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slot of the NOE module
WordArr5[2]	WORD	Destination drop number (or set to FF hex)
WordArr5[3]	WORD	Terminator (set to FF hex)
WordArr5[4]	WORD	No significance
WordArr5[5]	WORD	No significance

Function mode

Function mode of the CREAD_REG block

Although a large number of CREAD_REG function blocks can be programmed, only four read operations may be active at the same time. In such a case it is insignificant whether they are the result of this function block or others (e.g. MBP_MSTR, MSTR, READ_REG). All function blocks use one data transaction path and require multiple cycles to complete a job.

Note: A TCP/IP communication between a Quantum PLC (NOE 711 00) and a Momentum PLC (all TCP/IP CPUs and all TCP/IP I/O modules) is only possible, when only **one** read or write job is carried out in every cycle. If several jobs are sent per PLC cycle, the communication stops without generating an error message in the status register of the function block.

The entire routing information is contained in data structure WordArr5 of input AddrFld. The type of function block connected to this input and thus the contents of the data structure depend on the network used.

Please use:

- Modbus Plus for function block MODBUSP_ADDR
- TCP/IP Ethernet: the function block TCP_IP_ADDR
- SY/MAX Ethernet: the function block SYMAX_IP_ADDR

Note: For experts:

The WordArr5 data structure can also be used with constants.

Note: This function block puts a heavy load on the network. The network load must therefore be carefully monitored. If the network load is too high, the program logic should be reorganized in order to work with the READ_REG function block, a variation of this function block that does not operate in a continuous mode, but under command control.

Parameter description

SLAVEREG	Start of the area in the addressed slave from which the source data is read. The source area always resides within the 4x register area. SLAVEREG expects the source reference as offset within the 4x area. The leading "4" must be omitted (e.g. 59 (contents of the variables or value of the literal) = 40059). The parameter can be specified as direct address, located variable, unlocated variable or literal.
NO_REG	Number of registers to be read from the addressed slave (1 ... 100). The parameter can be entered as a Direct address, Located variable, Unlocated variable or Literal . The parameter can be entered as a Direct address, Located variable or Unlocated variable .
REG_READ	This word parameter addresses the first register in a series of NO_REG registers, listed one after the other, which are used as a destination data area. The parameter must be entered as a Direct address or located Variable .
STATUS	Error code, see *Runtime errors, p. 95*

CREADREG: Continuous register reading

3

Overview

Introduction

This chapter describes the CREADREG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block reads a register area continuously. It reads data from addressed nodes via Modbus Plus.

EN and ENO can be configured as additional parameters.

Note: It is necessary to be familiar with the routing procedures of your network when programming a CREADREG function. Modbus Plus routing path structures are described in detail in "Modbus Plus Network Planning and Installation Guide".

Note: This function block only supports the local Modbus Plus interface (no NOM).

If using a NOM please work with the block CREAD_REG.

Note: This function block does not support TCP/IP- or SY/MAX-Ethernet.

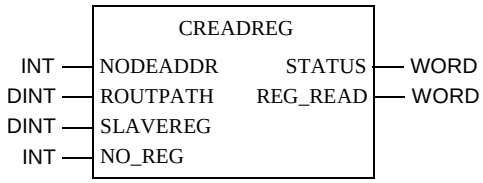
If TCP/IP- or SY/MAX-Ethernet is needed, please use the block CREAD_REG.

Note: For technical reasons, this function block does not allow the use of ST and IL programming languages.

Representation

Symbol

Block representation



Parameter description

Description of block parameters:

Parameter	Data type	Meaning
NODEADDR	INT	Device address within the target segment
ROUTEPATH	DINT	Routing path to target segment
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be read from
NO_REG	INT	Number of registers to be read by the slave
STATUS	WORD	Error code, see *Runtime errors, p. 95*
REG_READ	WORD	First 4x area register of the area, for values read

Function mode

Function mode of CREADREG blocks

Although a large number of CREADREG function blocks can be programmed, only four read operations may be active at the same time. It makes no difference whether these operations are performed using this function block or others (e.g. MBP_MSTR, MSTR, READREG). All function blocks use one data transaction path and require multiple cycles to complete a job.

The complete routing information must be separated into two parts:

- into the NOEADDR of the destination node (regardless of whether it is located in the local segment or in another segment) and
- the routing path, in case there is a link via network bridges.

The resulting destination address consists of these two information components.

The routing path is a DINT data type, which is interpreted as a sequence of two-digit information units. It is not necessary to use "00" extensions (e.g. both routing paths 4711 and 47110000 are valid, for NODEADDR 34 the result is destination reference 47.11.34.00.00).

Note: This function block puts a heavy load on the network; therefore the network load must be carefully monitored. If the network load is too high, the program logic should be reorganized, in order to work with the READREG function block, a variation of this function block that does not operate in a continuous mode, but under command control.

Parameter description

NODEADDR

Identifies the node address within the target segment.

The parameter can be specified as direct address, located variable, unlocated variable or literal.

ROUTPATH

Identifies the routing path to the target segment. The two-digit information units run from 01 … 64 (see *Function mode, p. 24*). If the slave resides in the local network segment, ROUTPATH must be set to "0" or must be left unconnected.

The parameter can be specified as direct address, located variable, unlocated variable or literal.

SLAVEREG

Start of the area in the addressed slave from which the source data are read. The source area always resides within the 4x register area. SLAVEREG expects the source address as offset within the 4x area. The initial "4" must be omitted (e.g. 59 (contents of the variable or value of the literal) = 40059).

The parameter can be specified as direct address, located variable, unlocated variable or literal.

NO_REG

Number of registers to be read from slave processor (1 ... 100).

The parameter can be specified as direct address, located variable, unlocated variable or literal.

STATUS

Error code, see *Runtime errors, p. 95*

The parameter can be specified as direct address, located variable or unlocated variable.

REG_READ

This word parameter addresses the first register in a series of NO_REG successive registers used as destination data area.

The parameter must be entered as a direct address or located variable.

CWRITE_REG: Continuous register writing

4

Overview

Introduction

This chapter describes the CWRITE_REG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The purpose of this Function block is to write the register area continuously. It transfers data from the PLC via Modbus Plus, TCP/IP Ethernet or SY/MAX Ethernet to an addressed slave.

EN and ENO can be configured as additional parameters.

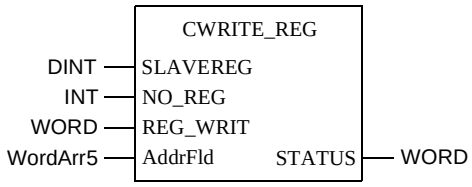
Note: You must be familiar with the routing procedures of the network when programming a CWRITE_REG function. (Modbus Plus routing path structures) are described in detail in "Modbus Plus Network Planning and Installation Guide". If TCP/IP or SY/MAX EtherNet is imp

Note: For technical reasons, this function block does not allow the use of ST and IL programming languages.

Representation

Symbol

Block representation:



Parameter description

Description of parameters:

Parameter	Data type	Meaning
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be written to
NO_REG	INT	Number of registers to be written to slave
REG_WRIT	WORD	First 4x register of the source data area
AddrFld	WordArr5	Data structure for transferring the Modbus Plus-address, TCI/IP address or SY/MAX-IP address.
STATUS	WORD	MSTR error code, see *Runtime errors, p. 95*

Elementary description for WordArr5 in Modbus Plus

Elementary description for WordArr5 in Modbus Plus:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: Routing register 1 is used for address specification (routing path addresses one of five) of the destination node during network transfer. The last byte in the routing path that is not zero is the destination node. High value byte: Slot of the network adapter module (NOM), if any.
WordArr5[2]	WORD	Routing register 2
WordArr5[3]	WORD	Routing register 3
WordArr5[4]	WORD	Routing register 4
WordArr5[5]	WORD	Routing register 5

Elementary description for WordArr5 with TCP/IP EtherNet

Elementary description for WordArr5 with TCP/IP EtherNet:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slots of the NOE module
WordArr5[2]	WORD	Byte 4 (MSB) of the 32-bit destination IP address
WordArr5[3]	WORD	Byte 3 of the 32-bit destination IP address
WordArr5[4]	WORD	Byte 2 of the 32-bit destination IP address
WordArr5[5]	WORD	Byte 1 (LSB) of the 32-bit destination IP address

Elementary description for WordArr5 with SYMAX EtherNet

Elementary description for WordArr5 with SYMAX EtherNet:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slot of the NOE module
WordArr5[2]	WORD	Destination drop number (or set to FF hex)
WordArr5[3]	WORD	Terminator (set to FF hex)
WordArr5[4]	WORD	No significance
WordArr5[5]	WORD	No significance

Function mode

CWRITE_REG block Function mode

Although a large number of CWRITE_REG function blocks can be programmed, only four write operations may be active at the same time. It makes no difference whether these operations are performed using this function block or others (e.g. MBP_MSTR, MSTR, WRITE_REG). All function blocks use one data transaction path and require multiple cycles to complete a job.

If several CWRITE_REG function blocks are used within an application, they must at least differ in the values of their NO_REG or REG_WRITE parameters.

Note: A TCP/IP communication between a Quantum PLC (NOE 711 00) and a Momentum PLC (all TCP/IP CPUs and all TCP/IP I/O modules) is only possible, when only **one**read or write job is carried out in every cycle. If several jobs are sent per PLC cycle, the communication stops without generating an error message in the status register of the function block.

The entire routing information is contained in data structure WordArr5 of input AddrFld. The type of function block connected to this input and thus the contents of the data structure depend on the network used.

Please use:

- Modbus Plus for function block MODBUSP_ADDR
- TCP/IP Ethernet: the function block TCP_IP_ADDR
- SY/MAX Ethernet: the function block SYMAX_IP_ADDR

Note: For experts:

The WordArr5 data structure can also be used with constants.

Note: This function block puts a heavy load on the network. The network load must therefore be carefully monitored. If the network load is too high, the program logic should be reorganized to work with the WRITE_REG function block, which is a variant of this function block that does not operate in continuous mode but is command driven.

Parameter description

SLAVEREG	Start of the area in the addressed slave to which the source data are written. The destination area always resides within the 4x register area. SLAVEREG expects the destination address as offset within the 4x area. The initial "4" must be omitted (e.g. 59 (contents of the variables or value of the literal) = 40059). The parameter can be specified as direct address, located variable, unlocated variable or Literal.
NO_REG	Number of registers to be written to slave processor (1 ... 100). The parameter can be specified as direct address , located variable, unlocated variable or Literal.
STATUS	Error code, see *Runtime errors, p. 95* The parameter can be specified as direct address, located variable or unlocated variable.
REG_WRIT	This word parameter addresses the first register in a series of NO_REG Successive registers used as source data area. The parameter must be entered as a direct address or located variable.

CWRITREG: Continuous register writing

5

Overview

Introduction

This chapter describes the CWRITEREG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The purpose of this Function block to write the register area continuously. It transfers data from the PLC via Modbus Plus to a specified slave destination processor.

EN and ENO can be configured as additional parameters.

Note: It is necessary to be familiar with the routing procedures of your network when programming a CWRITEREG function. Modbus Plus routing path structures will be described in detail in "Modbus Plus Network Planning and Installation Guide".

Note: This function block only supports the local Modbus Plus interface (no NOM).

If using a NOM please work with the block CWRITE_REG.

Note: This function block does not support TCP/IP- or SY/MAX-Ethernet.

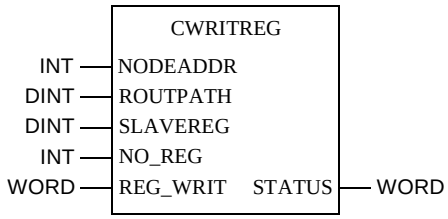
If TCP/IP- or SY/MAX-Ethernet is needed, please use the block CWRITE_REG.

Note: For technical reasons, this function block does not allow the use of ST and IL programming languages.

Representation

Symbol

Block representation



Parameter description

Description of parameters:

Parameter	Data type	Meaning
NODEADDR	INT	Device address within the target segment
ROUTEPATH	DINT	Routing path to target segment
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be written to
NO_REG	INT	Number of registers to be written by the slave
REG_WRIT	WORD	First 4x register of the source data area
STATUS	WORD	Error code, see *Runtime errors, p. 95*

Function mode

Function mode of CWRITEREG blocks

Although an unlimited number of CWRITEREG function blocks can be programmed, only four write operations may be active at the same time. It makes no difference whether these operations are performed using this function block or others (e.g., MBP_MSTR, MSTR, WRITEREG). All function blocks use one data transaction path and require multiple cycles to complete a job.

If several CWRITEREG function blocks are used within an application, they must at least differ in the values of their NO_REG or REG_WRITE parameters.

The complete routing information must be separated into two parts:

- into the NODEADDR of the destination node (regardless of whether it is located in the local segment or in another segment) and
- the routing path, in case there is a link via network bridges.

The destination address arising from this is made from these two items of information.

The routing path is a DINT data type, which is interpreted as a sequence of two-digit information units. Appended "00" are not required (e.g. both routing paths 4711 and 47110000 are valid, for NODEADDR 34 the result is destination reference 47.11.34.00.00).

Note: This function block puts a heavy load on the network. The network load must therefore be carefully monitored. If the network load is too high, the program logic should be reorganized to work with the WRITEREG function block, which is a variant of this function block that does not operate in continuous mode, but is command driven.

Parameter description

NODEADDR

Identifies the node address within the target segment.

The parameter can be specified as direct address, located variable, unlocated variable or Literal.

ROUTPATH

Identifies the routing path to the target segment. The two-digit information units run from 01 … 64 (see *Function mode, p. 36*). If the slave resides in the local network segment, ROUTPATH must be set to "0" or must be left unconnected.

The parameter can be specified as direct address, located variable, unlocated variable or Literal.

SLAVEREG

Start of the destination area in the addressed slave to which the source data are written. The source area always resides within the 4x register area. SLAVEREG expects the destination reference as offset within the 4x area. The initial "4" must be omitted (e.g. 59 (contents of the variable or value of the literal) = 40059).

The parameter can be specified as direct address, located variable, unlocated variable or Literal.

NO_REG

Number of registers to be written to slave processor (1 ... 100).

The parameter can be specified as direct address, located variable, unlocated variable or Literal.

REG_WRIT

This word parameter addresses the first register in a series of NO_REG successive registers used as source data area.

The parameter must be specified as a direct address or located variable.

STATUS

Reports MSTR error code, see *Runtime errors, p. 95*

The parameter can be specified as direct address, located variable or unlocated variable.

IBS_READ: Reading variables via INTERBUS

6

Brief description

Function description

You can use this function block to read data into the status RAM of the PLC from a PCP slave connected over the INTERBUS.

Note: EN and ENO should not be used in conjunction with this EFB, otherwise output parameters may become frozen.

Detailed Description

The detailed description for the function block can be found in the NOA 622 User Manual.

IBS_SEND_REQ: Diagnostic query on the INTERBUS Master 140 NOA 622 00

7

Brief description

Function description

You can use this function block to request data from a specified INTERBUS Master NOA 622 00 and store it in the status RAM of the PLC.

Note: EN and ENO should not be used in conjunction with this EFB, otherwise output parameters may become frozen.

Detailed Description

The detailed description for the function block can be found in the NOA 622 User Manual.

IBS_WRITE: Writing variables to INTERBUS PCP nodes

8

Brief description

Function description

You can use this function block to write data from the status RAM of the PLC to a PCP slave connected over the INTERBUS.

Note: EN and ENO should not be used in conjunction with this EFB, otherwise output parameters may become frozen.

Detailed Description

The detailed description for the function block can be found in the NOA 622 User Manual.

ICNT: Connect/disconnect an INTERBUS communication

9

Overview

Introduction

This chapter describes the ICNT block.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function Description

The function block is used to create or break a communication connection. This is done using the context management services Initiate and Abort.

As additional parameters, EN and ENO can be configured.

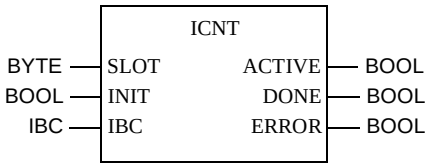
Note: This PCP communication block cannot be used with CPUs 140 CPU 434 12 and 140 CPU 534 14. When using these types of CPUs, please use the LL984 instruction ICNT in a LL984 section.

This LL984 instruction is not a part of the Concept delivery and must be added in Concept as loadable. You can find this loabable on our homepage **http:// www.schneiderautomation.com** → **Support & Services** → **Other Networks** → **Software Library**.

Representation

Symbol

Block representation



Parameter description

Description of parameters:

Parameter	Data type	Meaning
SLOT	BYTE	The Concept slot address corresponds to the appropriate INTERBUS Master NOA 611 10.
INIT	BOOL	Is an edge controlled signal (0/1). If INIT = 0/1 and SERVICE = 1 (element in datastructure IBC) the connection to the INTERBUS PCP slaves is established by using the Initiate service. If INIT = 0/1 and SERVICE = 0 the connection will be severed by using the Abort service, and internal bits are deleted (equivalent to RESET in ICOM). If an Abort request is received the function block tries to re-establish the connection, provided a new 0/1 signal is available at the INIT input.
IBC	IBC	For a description of the data structure, see *IBC data structure, p. 48*
ACTIVE	BOOL	The setting of this binary output continues at 1 during the execution of the specified service.
DONE	BOOL	Confirms that the service has been executed without any errors. In case of Abort it should be reset to DONE = 0.
ERROR	BOOL	This binary output is set to 1 if a negative response has been received, the link has been cancelled, or a parameterizing error of the user has occured. The remaining error information err_cd and err_cl in the data structure IBC will be deleted after correcting the error.

IBC data structure

IBC is a data structure with the following elements:

Element	Element type	Meaning
service	BYTE	Specifies the selected service (1: Initiate, 0: Abort)
err_cd	BYTE	Error number, see *Err_cd (error code) when error class is 0, p. 49*
err_cl	BYTE	Error class, see *Err_cl (error class), p. 49*
cr	BYTE	Communication reference on the PCP slave
size	BYTE	not used
e_par	BYTE	is for special Error messages of the function block
index	WORD	not used
subindex	BYTE	not used
fillbyte_1	BYTE	not used
fillword_1... fillword_5	WORD	Contains sections of the error message and is sent if: 1. If no connection could be established 2. If a connection is to be established, even though one already exists The following table shows how the error message should be read. Further information regarding the error message can be found both in the description of the data structure elements err_cd and er_cl and in the documentation of the PCP nodes.
fillword_6	WORD	For internal use only

Reading the error message:

Element	Meaning for a failed connect attempt (High value byte/Low value byte)	Meaning at failed connection attempt during existing connection (High byte/Low byte)
fillword_1	0000/Additional code	Locally generated/Abort ID
fillword_2	Additional code/Send buffer	Reason code/Abort detail
fillword_3	Send buffer/Receive buffer	0/0
fillword_4	Receive buffer/Service	0/0
fillword_5	818C Hex	81AD Hex

Runtime errors

Introduction

Information on runtime errors that have occurred is to be found in the following elements of the IBC data structure:

- Err_cl (error class)
- Err_cd (error code)
- e_par (error parameters)

Err_cl (error class)

Error class key:

Error class	Meaning
0	This type of error is registered with Initiate Request in case of an error during connection establishment.
5	This type of error is registered in case of a service error.
6	This type of error is registered in case of an access error.
8	This type of error is registered in case of module-specific errors.

Err_cd (error code) when error class is 0

Meaning of error codes when error class is 0:

Error code	Meaning	Action
1	The sizes of the transmit buffer and receive buffer of both communication devices do not agree.	Using Receive CRL Request, adjust the buffer size of the master module to that of the INTERBUS node.
2	The services supported by the two communication devices do not correspond.	Using Receive CRL Request, change the supported services of the master module.
4	This error message is module-specific.	Refer to the module description for details.

Err_cd (error code) when error class is 5

Meaning of error codes when error class is 5:

Error code	Meaning	Action
1	This error only occurs during start or stop. A start or stop command has been transmitted twice. Since the start or stop has already been executed, it cannot be executed again.	No action necessary.
5	This error only occurs during the "Get OD" service: An illegal value has been entered in the Access Specification parameter.	Look up the valid values in the module description and send the service again.

Err_cd (error code) when error class is 6

Meaning of error codes when error class is 6:

Error code	Meaning	Action
2	Access to the module is not possible due to a hardware error. Example: power supply not available.	Correct the hardware error.
3	Limited access rights exist for the module: e.g. read-only (write protected), password-protected.	Look up the access rights in the module description.
5	A service parameter has been given an illegal value. For example, wrong length or illegal subindex.	Using the module description, check the parameters and send the service again with the corrected values.
6	The service in use cannot be performed in this module. For example, a program sequence can be started or stopped, but not read.	Look up the permissible services in the description for this module.
7	Module does not exist. Probably a typing mistake with the index.	Using the module description, check the module index and re-initialize the service.

Err_cd (error code) when error class is 8

Meaning of error codes when error class is 8:

Error code	Meaning	Action
0	Module-specific error	For details refer to the module description.

e_par (error parameters)

Error parameter key:

Code (Hex)	Meaning
F9	Internal error
FB	INTERBUS Master not operational. NOA 611 10 faulty or not plugged in.
FC	INTERBUS master has not been configured
FD	Internal error
FE	Internal error
FF	Internal error
E1	Wrong number in IBC service word
E2	Wrong slot for NOA 611 10
E3	Wrong CR (<2 or >64)
E4	Internal error
E5	Timeout reached (over 24 sec after start of a service, e.g. initialize, abort, read, write)
E6	No connection (if ICNT Enable = 0 and ICOM Enable = 1)
E8	Internal error
E9	Internal error
EA	Error abort
EC	Framing error (e.g. size, index, subindex)

ICOM: Data transfer

10

Overview

Introduction

This chapter describes the ICOM block.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function Description

The function block is used for normal data transfer with the services "Read" and "Write" between the signal memory on the PLC and the INTERBUS-PCP-Slave.

As additional parameters, EN and ENO can be configured.

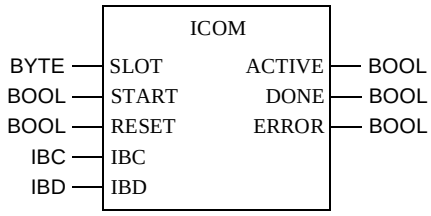
Note: This PCP communication block cannot be used with CPUs 140 CPU 434 12 and 140 CPU 534 14. When using these types of CPUs, please use the LL984 instruction ICOM in a LL984 section.

This LL984 instruction is not a part of the Concept delivery and must be added in Concept as loadable. You can find this loabable on our homepage **http:// www.schneiderautomation.com** → **Support & Services** → **Other Networks** → **Software Library**.

Representation

Symbol

Block representation



Parameter description

Description of parameters:

Parameter	Data type	Meaning
SLOT	BYTE	The Concept slot address corresponds to the appropriate INTERBUS Master NOA 611 10.
START	BOOL	is an edge-controlled signal. In case of START = 0/1 and SERVICE = 2 or 3 (element in datastructure IBC) data will be send or received to the INTERBUS PCP slaves. During RESET = 0/1 no communication services are executed and the EFB is waiting for a new signal.
RESET	BOOL	is an edge-controlled signal. RESET = 0/1 is used to reset the function block in the default status of the internal state machine.
IBC	IBC	For a description of the data structure, see *IBC data structure, p. 56*
IBD	IBD	For a description of the data structure, see *IBD data structure, p. 57*
ACTIVE	BOOL	This binary output is set to 1 as long as the specified service is being executed.
DONE	BOOL	Signals that the treatment of the service is finished without any failures. DONE=1 is set, only in case of an errorless Read/Write service. In case of a Reset it will be set as DONE=0.
ERROR	BOOL	This binary output is set to 1 when a negative response has been received, the service execution has been canceled through the RESET signal, or a parameterization failure of the user has been occurred. The error output is reset as soon as a new service has been issued.

The input START is an edge controlled signal (0->1), but RESET has priority.

IBC data structure

IBC is a data structure with the following elements:

Element	Data type	Meaning
service	BYTE	specifies the selected service (READ = 2, WRITE = 3)
err_cd	BYTE	Error number, see ICNT runtime error (See *Err_cd (error code) when error class is 0, p. 49*)
err_cl	BYTE	Error class, see ICNT runtime error (See *Err_cd (error code) when error class is 0, p. 49*)
cr	BYTE	Communications reference on the PCP slave
size	BYTE	contains the number of data bytes used within the "Data" register area (max. 256)
e_par	BYTE	is not used for special Error messages of the function blocks
index	WORD	equivalent to the Index of the data object within the INTERBUS PCP slave
subindex	BYTE	equivalent to the Subindex of the data object within the INTERBUS PCP slave (The Index and the Subindex should be taken out of the user manual of the Interbus PCP slave!)
fillbyte_1	BYTE	not used
fillword_1 ... fillword_5	WORD	Contains sections of the error message and is sent when: 1. If no connection could be established 2. If a connection is to be established, even though one already exists The following table shows how the error message is read. Further information regarding the error message can be found within the data structure's elements err_cd and er_cl as well as within the PCP node's documentation.
fillword_6	WORD	For internal use only

Reading the error message:

Element	Meaning for read or write fault(High value byte/Low value byte)	Meaning for service denial(High value byte/Low value byte)
fillword_1	0/Additional code	Detected here/Original invoke ID
fillword_2	Additional code/0	Reject PDU type/ Reject code
fillword_3	0/0	0/0
fillword_4	0/0	0/0
fillword_5	8181 or 8182 Hex	81AE Hex

IBD data structure

IBD is a datastructure with the following elements:

Element	Element type	Meaning
IBD	ARRAY (1 .. 128) OF WORD	The datastructure IBD consists of 128 WORD elements. The number 256 relays to the parameter size within the datastructure IBC.

Runtime error

Runtime error

See ICNT description (See *Runtime errors, p. 49*)

MBP_MSTR: Modbus Plus Master

11

Overview

Introduction

This chapter describes the MBP_MSTR block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

With this Function block, it is possible to select one of 12 available network communication operations.

Note: As this function block supports 12 different network communication operations, its parameterization is very complicated. Because of this, simplified EFBs are available for reading and writing registers (READ_REG, CREAD_REG, WRITE_REG, CWRITE_REG).

EN and ENO can be configured as additional parameters.

Note: You must be familiar with the routing procedures of your network when programming an MSTR function. Modbus Plus routing path structures are described in detail in the "Modbus Plus Network Planning and Installation Guide". If TCP/TP or SY/MAX EtherNet is implemented, standard Ethernet IP router products must be used. The "Quantum Ethernet TCP/IP Module User Guide" provides a complete description of TCP/IP routing.

Restrictions

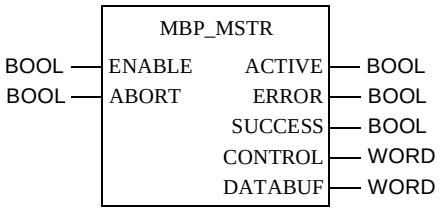
Note the following restrictions:

- Although a large number of MBP_MSTR function blocks can be programmed, only four of them can be active at the same time. All function blocks use one data transaction path and require multiple cycles to complete a job.
- A TCP/IP communication between a Quantum PLC (NOE 211 00) and a Momentum PLC (all TCP/IP CPUs and all TCP/IP I/O modules) is only possible if only **one**read or write job is carried out in every cycle. If several jobs are sent per PLC cycle, the communication stops without generating an error message in the status register of the function block.
- In FBD and LD sections, the function block can only be used on the program level, i.e. not in Derived Function Blocks (DFBs).
- For technical reasons, the function block does not allow the use of ST and IL programming languages.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
ENABLE	BOOL	Enable MSTR function
ABORT	BOOL	Cancel active MSTR operation
ACTIVE	BOOL	Operation is active
ERROR	BOOL	Faulty operation
SUCCESS	BOOL	Operation completed successfully
CONTROL	WORD	First 4x register of the MSTR control block
DATABUF	WORD	First 4x register of the data field

Function mode

Function mode of MBP_MSTR blocks

Using the MBP_MSTR block, one of 12 available network communication operations can be triggered via the network. Each operation receives a code. Whether the operations are available depends on the type of network used.

Valid function codes

Valid function codes:

Code	Function	Modbus Plus	TCP/IP Ethernet	SY/MAX Ethernet
1	Write data	X	X	X
2	Read data	X	X	X
3	Get local statistics	X	X	-
4	Clear local statistics	X	X	-
5	Write global data	X	-	-
6	Read global data	X	-	-
7	Get remote statistics	X	X	-
8	Clear remote statistics (See *Clear remote statistics, p. 80*)	X	X	-
9	Peer Cop Status (Peer Cop Health)	X	-	-
10	Reset optional module	-	X	X
11	Read CTE (Config extension)	-	X	X
12	Write CTE (Config extension)	-	X	X

Legend:

X	Yes
-	No

Parameter description

ENABLE	When ON, the operation specified in the first CONTROL register is enabled.
ABORT	When ON, the currently active operation is aborted.
ACTIVE	ON, if the operation is active.
ERROR	ON, if the operation was aborted without success.
SUCCESS	ON, if the operation concluded successfully.
DATABUF	The 4x register specified is the first in a group of successive output/marker words, making up the data field. For operations providing data, e.g. the write operation, the data field is the data source. For operations receiving data, e.g. the read operation, the data field is the data sink. In the case of Ethernet CTE Read and Write operations, the middle input stores the contents of the Ethernet configuration extension table in a series of registers.
CONTROL	This word parameter addresses the first of several successive 4x registers. The control block is contained in these registers. The first register displayed contains a number from 1 to 12, which provides the operation code of the Modbus operation to be performed. The contents of the sequence registers are determined by the operation. The structure of the control block differs according to the network used: • Modbus Plus • TCP/IP Ethernet • SY/MAX Ethernet

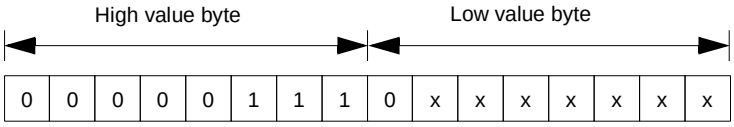
Control block for Modbus Plus

Control block for Modbus Plus:

Register	Contents
4x	indicates one of the Operations which are valid for Modbus Plus.
4x + 1	indicates the Error status.
4x +2	indicates the length (number of registers transferred)
4x +3	indicates MSTR operation-dependent information
4x +4	Routing register 1 is used to specify the address (routing path address one of five) of the destination node during a network transfer. The last byte in the routing path that is not zero, is the destination mode.
4x +5	Routing register 2
4x +6	Routing register 3
4x +7	Routing register 4
4x +8	Routing register 5

Routing register 1 (4x + 4) in Modbus Plus

If a Modbus Plus network option module (NOM) in the rack of a Quantum controller is addressed as destination node, the value in the high value byte represents the physical slot of the NOM, i.e. if the NOM is inserted in slot 7 of the rack, the high value byte of control register 1 looks as follows:



High value byte Slots 1 to 16

Low value byte Destination address (binary value between 1 and 64 (normal) or 65 > 255 (extended))

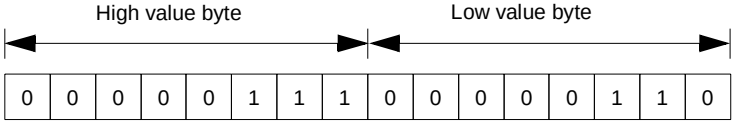
Control block for TCP/IP Ethernet

Control block for TCP/IP Ethernet:

Register	Contents
4x	indicates one of the Operations which are valid for TCP/IP.
4x + 1	indicates the Error status (See *Runtime errors, p. 95*).
4x +2	indicates the length (number of registers transferred)
4x +3	indicates MSTR operation-dependent information
4x +4	Routing register, Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slot of the NOE module
4x +5	Byte 4 (MSB) of the 32-bit destination IP address
4x +6	Byte 3 of the 32-bit destination IP address
4x +7	Byte 2 of the 32-bit destination IP address
4x +8	Byte 1 (LSB) of the 32-bit destination IP address

Routing register (4x + 4) in TCP/IP Ethernet

If a NOE in the rack of a Quantum controller is addressed as destination node, the value in the high value byte represents the physical NOE slot and the value in the low value byte represents the MBP on Ethernet (MET) mapping index, i.e. if the NOE is plugged in at Slot 7 of the rack and the MET mapping index is 6, the first element of the data structure appears as follows:



High value byte Slots 1 to 16

Low value byte MBP on Ethernet Transporter (MET) mapping index

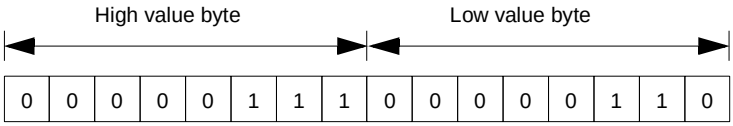
Control block for SY/MAX Ethernet

Control block for SY/MAX Ethernet:

Register	Contents
4x	indicates one of the Operations which are valid for SY/MAX.
4x + 1	indicates the Error status.
4x +2	indicates the length (number of registers transferred)
4x +3	indicates MSTR operation-dependent information
4x +4	Routing register, Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slot of the NOE module
4x +5	Destination drop number (or set to FF hex)
4x +6	Terminator (set to FF hex)

Routing register (4x + 4) in SY/MAX Ethernet

If a NOE in the rack of a Quantum controller is addressed as destination node, the value in the high value byte represents the physical NOE slot and the value in the low value byte represents the MBP on Ethernet (MET) mapping index, i.e. if the NOE is plugged in at Slot 7 of the rack and the MET mapping index is 6, the first element of the data structure appears as follows:



High value byte Slots 1 to 16

Low value byte MBP on Ethernet Transporter (MET) mapping index

Write data

Brief description

The write operation transfers data to an addressed node. The transaction utilizes a master transaction path and may require several cycles.

An attempt to program the MBP_MSTR in such a way that it writes to its own station address will generate an error in the 4x+1 register of the block. However, it is possible to perform a write operation to a non-existing slave register. The slave detects the status logs it. This can last for several cycles.

Network implementation

The write operation can be performed on Modbus Plus, TCP/IP Ethernet and SY/MAX Ethernet networks.

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	1 = Write data
4x+1	indicates the Error status.
4x+2	Number of registers sent to slave
4x+3	Determines the 4x starting register in the slave to which the data must be written (e.g. 1 = 40001, 49=40049)
4x+4 … 4x+8	Routing register 1 is used to specify the address (routing path address one of five) of the destination node during a network transfer. The last byte in the routing path that is not zero, is the destination mode.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	1 = Write data
4x+1	indicates the Error status.
4x+2	Number of registers sent to slave
4x+3	Determines the 4x starting register in the slave to which the data must be written (e.g. 1 = 40001, 49=40049)
4x+4	Routing register, Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Network adapter module slot
4x+5 … 4x+8	Each register contains one byte of the 32-bit IP address

Use of control blocks for SY/MAX Ethernet (CONTROL)

Control block for SY/MAX Ethernet (CONTROL)

Register	Meaning
4x	1 = Write data
4x+1	indicates the Error status.
4x+2	Number of registers sent to slave
4x+3	Determines the 4x starting register in the slave to which the data must be written (e.g. 1 = 40001, 49=40049)
4x+4	Routing register, Slot ID Low value byte: Destination drop number High value byte: Network adapter module slot
4x+5 … 4x+8	Terminator: FF hex

Read data

Brief description

The read operation transfers data from a specified node on the network. The transaction utilizes a master transaction path and may require several cycles.

An attempt to program the MBP_MSTR in such a way that it reads from its own station address will generate an error in the 4x+1 register of the block. But it is possible to perform a read operation on a non-existing register of the slave. The slave detects the status logs it. This can last for several cycles.

Network implementation

The read operation can be performed on Modbus Plus, TCP/IP Ethernet and SY/MAX Ethernet networks.

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	2 = Read data
4x+1	indicates the Error status.
4x+2	Number of registers to be read from the slave
4x+3	Determines the 4x starting register in the slave from which the data must be read (e.g. 1 = 40001, 49 = 40049)
4x+4 … 4x+8	Routing register 1 is used to specify the address (routing path address one of five) of the destination node during a network transfer. The last byte in the routing path that is not zero, is the destination mode.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	2 = Read data
4x+1	indicates the Error status.
4x+2	Number of registers to be read from the slave
4x+3	Determines the 4x starting register in the slave from which the data must be read (e.g. 1 = 40001, 49 = 40049)
4x+4	Routing register, Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Network adapter module slot
4x+5 … 4x+8	Each register contains one byte of the 32-bit IP address

Use of control blocks for SY/MAX Ethernet (CONTROL)

Control block for SY/MAX Ethernet (CONTROL)

Register	Meaning
4x	2 = Read data
4x+1	indicates the Error status.
4x+2	Number of registers to be read from the slave
4x+3	Determines the 4x starting register in the slave to which the data must be written (e.g. 1 = 40001, 49=40049)
4x+4	Routing register, Slot ID Low value byte: Destination drop number High value byte: Network adapter module slot
4x+5 … 4x+8	Terminator: FF hex

Read local statistics

Brief description

This operation reads the data from the local node. The operation is carried out in one scan and does not require a master transaction path.

Network implementation

The write operation can be performed on Modbus Plus, TCP/IP Ethernet and SY/MAX Ethernet networks:

- List of available Modbus Plus network statistics (See *Modbus Plus network statistics, p. 89*)
- List of TCP/IP Ethernet network statistics (See *TCP/IP Ethernet network statistics, p. 94*)

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	3 = Read local statistics
4x+1	indicates the Error status.
4x+2	Number of registers to be read from the local statistics (1...32)
4x+3	First register from which the statistics table must be read (Reg1=0)
4x+4	Routing register 1 is used to specify the address (routing path address one of five) of the destination node during a network transfer. The last byte in the routing path that is not zero, is the destination mode.

Note: If your controller does not support any Modbus Plus option modules (S985s or NOMs), the High value byte of the 4x+4 register will not be used and the bits of the High value byte must all be set to 0.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL)

Register	Meaning
4x	3 = Read local statistics
4x+1	indicates the Error status.
4x+2	Number of registers to be read from the local statistics (1...32)
4x+3	First register from which the statistics table must be read (Reg1=0)
4x+4	Routing register, High value byte: Network adapter module slot
4x+5 … 4x+8	no significance

Clear local statistics

Brief description

This operation deletes the statistics concerning the local node. The operation is carried out in one scan and does not require a master transaction path.

Note: If the "Clear local statistics" operation is edited, only the words 13 to 22 in the statistics table are cleared.

Network implementation

The operation can be performed on Modbus Plus and TCP/IP Ethernet networks.

- List of available Modbus Plus network statistics (See *Modbus Plus network statistics, p. 89*)
- List of TCP/IP Ethernet network statistics (See *TCP/IP Ethernet network statistics, p. 94*)

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	4 = Clear local statistics
4x+1	indicates the Error status.
4x+2	Reserved
4x+3	Reserved
4x+4	Routing register 1 is used to specify the address (routing path address one of five) of the destination node during a network transfer. The last byte in the routing path that is not zero, is the destination mode.

Note: If your controller does not support any Modbus Plus option modules (S985s or NOMs), the High value byte of the 4x+4 register will not be used and the bits of the High value byte must all be set to 0.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	4 = Clear local statistics
4x+1	indicates the Error status.
4x+2	Reserved
4x+3	Reserved
4x+4	Routing register, High value byte: Network adapter module slot
4x+5 … 4x+8	Reserved

Write global data

Brief description

This operation transfers data to the communication processor of the current node, so that it can be sent via the network, as soon as the node receives the token. This data can be received by all nodes connected to the local network. The operation is carried out in one scan and does not require a master transaction path.

Network implementation

The operation can only be performed on Modbus Plus networks.

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	5 = Write global data
4x+1	indicates the Error status.
4x+2	Number of registers to be sent from State RAM into global data memory (comm processor) (1...32)
4x+3	Reserved
4x+4	Routing address 1 If this is the second of two local nodes, set the value of the High value byte to 1.

Note: If your controller does not support any Modbus Plus option modules (S985s or NOMs), the High value byte of the 4x+4 register will not be used and the bits of the High value byte must all be set to 0.

Read global data

Brief description

This operation reads data from the communications processor of any node connected to the network that sends out global data. The operation can take several cycles, if the global data is not currently available with the nodes called. If global data is available, the operation runs down in one cycle. A master transaction path is not required.

Network implementation

The operation can only be performed on Modbus Plus networks.

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	6 = Read global data
4x+1	indicates the Error status.
4x+2	Number of registers to be sent from global data memory (comm processor) (1...32)
4x+3	Display of registers available in scanned node (will be automatically updated)
4x+4	Routing register 1 is used to specify the address (routing path address one of five) of the destination node during a network transfer. The last byte in the routing path that is not zero, is the destination mode.

Note: If your controller does not support any Modbus Plus option modules (S985s or NOMs), the High value byte of the 4x+4 register will not be used and the bits of the High value byte must all be set to 0.

Get remote statistics

Brief description

This operation reads the data referring to remote nodes on the network (see *Modbus Plus network statistics, p. 89* and *TCP/IP Ethernet network statistics, p. 94*). This operation can last for several cycles and does not require a master data transaction path.

With each query, the remote communications processor supplies a complete statistics table even if the query does not refer to the entire table. MBP_MSTR will then copy only those words into the identified 4x registers that you queried.

Network implementation

The operation can be performed on Modbus Plus and TCP/IP Ethernet networks.

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	7 = Get remote statistics
4x+1	indicates the Error status.
4x+2	Number of registers to be read from the statistics data field (1...54) The size of the data field may not be exceeded.
4x+3	First register from which the node statistics must be read. The number of available statistics registers may not be exceeded.
4x+4 …	Routing address 1 … 5 of the node.
4x+8	Refers to routing path addresses one to five. The last byte if the routing path that is different from zero is the destination node.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	7 = Get remote statistics
4x+1	indicates the Error status.
4x+2	Number of registers to be read from the statistics data field (1...54) The size of the data field may not be exceeded.
4x+3	First register from which the node statistics must be read. The number of available statistics registers may not be exceeded.
4x+4	Routing register, High value byte: Network adapter module slot
4x+5 … 4x+8	Each register contains one byte of the 32-bit IP address

Clear remote statistics

Brief description

This operation clears the statistics concerning remote nodes on the network from the data field of the local node. This operation can last for several cycles and employs one single master data transaction path.

Note: If the "Clear remote statistics"operation is edited, only the words 13 through 22 of the statistics table (see *Modbus Plus network statistics, p. 89* and *TCP/IP Ethernet network statistics, p. 94*) will be deleted.

Network implementation

The write operation can be performed on Modbus Plus and TCP/IP Ethernet networks.

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	8 = Clear remote statistics
4x+1	indicates the Error status.
4x+2	Reserved
4x+3	Reserved
4x+4 … 4x+8	Routing register 1 is used to specify the address (routing path address one of five) of the destination node during a network transfer. The last byte in the routing path that is not zero, is the destination mode.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	8 = Clear remote statistics
4x+1	indicates the Error status.
4x+2	Reserved
4x+3	Reserved
4x+4	Routing register, High value byte: Network adapter module slot
4x+5 … 4x+8	Each register contains one byte of the 32-bit IP address

Peer cop health

Brief description

This operation reads the selected data from the peer cop communications health table and downloads the respective data into the specified 4x registers of State RAM. The Peer cop communications health table is 12 words long, MBP_MSTR indexes all words with 0 through 11.

Network implementation

The operation can only be performed on Modbus Plus networks.

Use of control blocks for Modbus Plus (CONTROL)

Control block for Modbus Plus (CONTROL):

Register	Meaning
4x	9 = Peer cop health
4x+1	indicates the Error status.
4x+2	Number of words wanted by the peer cop table (1..0,12)
4x+3	First word to be read from the peer cop table (0...11; 0=first word in peer cop table and 11=last word in peer cop table)
4x+4	Routing address 1 If this is the second of two local nodes, set the High value byte to 1.

Note: If your controller does not support any Modbus Plus option modules (S985s or NOMs), the High value byte of the 4x+4 register will not be used and the bits of the High value byte must all be set to 0.

Optional module reset

Brief description

The "Reset option module" operation leads a Quantum NOE option module to start a reset cycle to reset its working environment.

Network implementation

The write operation can be performed on TCP/IP Ethernet and SY/MAX Ethernet networks.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	10 = Optional module reset
4x+1	indicates the Error status.
4x+2	no significance
4x+3	no significance
4x+4	Routing register, The number shown in the High value byte in area 1 through 16 indicates the slot where the option module is located.
4x+5 … 4x+8	no significance

Use of control blocks for SY/MAX Ethernet (CONTROL)

Control block for SY/MAX Ethernet (CONTROL)

Register	Meaning
4x	10 = Optional module reset
4x+1	indicates the Error status.
4x+2	no significance
4x+3	no significance
4x+4	Routing register, High value byte: Network adapter module slot
4x+5 … 4x+8	no significance

Read CTE (Config extension table)

Brief description

The "Read CTE" operation reads a given number of bytes from the Ethernet configuration extension table in the specified buffer in the PLC memory. The bytes to be read start with a byte offset at the start of the CTE. The contents of the Ethernet CTE table is displayed on output DATABUF.

Network implementation

The write operation can be performed on TCP/IP Ethernet and SY/MAX Ethernet networks.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	11 = Read CTE (Config extension table)
4x+1	Indicates the Error status.
4x+2	No significance
4x+3	No significance
4x+4	Routing register, Low value byte = mapping index Either a value displayed in the byte of the register or is not used. or High value byte = slot ID Network adapter module slot
4x+5 … 4x+8	The number shown in the Low value byte in area 1 through 16 indicates the slot where the option module is located.

Use of control blocks for SY/MAX Ethernet (CONTROL)

Control block for SY/MAX Ethernet (CONTROL)

Register	Meaning
4x	11 = Read CTE (Config extension table)
4x+1	Indicates the Error status.
4x+2	Number of words transferred
4x+3	Byte offset in the PLC register structure, specifying from where the CTE bytes are read.
4x+4	Routing register, High value byte: Slots of the NOE module
4x+5 … 4x+8	Terminator: FF hex

CTE Indicator implementation (DATABUF)

The values in the Ethernet configuration extension table (CTE) are displayed in a series of registers on output DATABUF when a CTE read operation is implemented. DATABUF contains the first of 11 contiguous 4x registers. The registers display the following CTE data:

CTE Indicator implementation (DATABUF)

Parameter	Register	Contents
Frame type	4x	1 = 802.3 2 = Ethernet
IP address	4x+1	First byte of the IP address
	4x+2	Second byte of the IP address
	4x+3	Third byte of the IP address
	4x+4	Fourth byte of the IP address
Lower netmask	4x+5	High word:
	4x+6	Low word:
Gateway	4x+7	First byte of the gateway
	4x+8	Second byte of the gateway
	4x+9	Third byte of the gateway
	4x+10	Fourth byte of the gateway

Write CTE (Config extension table)

Brief description

The "Write CTE" operation writes the CTE configuration table from the specified data (DATABUF) to a specified Ethernet configuration extension table or to a specific slot.

Network implementation

The write operation can be performed on TCP/IP Ethernet and SY/MAX Ethernet networks.

Use of control blocks for TCP/IP Ethernet (CONTROL)

Control block for TCP/IP Ethernet (CONTROL):

Register	Meaning
4x	12 = Write CTE (Config extension table)
4x+1	indicates the Error status.
4x+2	No significance
4x+3	No significance
4x+4	Routing register, Low value byte = mapping index Either a value displayed in the byte of the register or is not used. or High value byte = slot ID Network adapter module slot
4x+5 … 4x+8	The number shown in the Low value byte in area 1 through 16 indicates the slot where the option module is located.

Use of control blocks for SY/MAX Ethernet (CONTROL)

Control block for SY/MAX Ethernet (CONTROL)

Register	Meaning
4x	12 = Write CTE (Config extension table)
4x+1	indicates the Error status.
4x+2	Number of words transferred
4x+3	Byte offset in the PLC register structure specifying where the CTE bytes are written.
4x+4	Routing register, High value byte = slot ID Slots of the NOE module Low value byte = Destination drop number
4x+5	Terminator: FF hex
4x+5… 4x+8	No significance

CTE Indicator implementation (DATABUF)

The values in the Ethernet extension table (CTE) are displayed in a series of registers on output DATABUF when a CTE write operation is implemented. DATABUF contains the first of 11 contiguous 4x registers. The registers are used to transfer the following CTE data:

CTE Indicator implementation (DATABUF)

Parameter	Register	Contents
Frame type	4x	1 = 802.3 2 = Ethernet
IP address	4x+1	First byte of the IP address
	4x+2	Second byte of the IP address
	4x+3	Third byte of the IP address
	4x+4	Fourth byte of the IP address
Lower netmask	4x+5	High word:
	4x+6	Low word:
Gateway	4x+7	First byte of the gateway
	4x+8	Second byte of the gateway
	4x+9	Third byte of the gateway
	4x+10	Fourth byte of the gateway

Peer cop communications health status

Peer cop communications health status

The table containing the Peer cop status information fills 12 consecutive registers, which can be indexed with the numbers 0 to 11 in an MBP_MSTR operation. Each individual bit of the table words is used to present one aspect of communications health that refers to a specific node on the Modbus Plus network.

Relation bit network node

The bits of the words 0 to 3 represent the health at the global communications input of nodes 1 to 64. The bits of words 4 to 7 represent the health of the output of a specific node.

The bits in words 8 to 11 represent the health of the input of a specific node.

Status type	Word index	Relation bit network node
Global input	0	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	1	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	2	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	3	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
Specific output	4	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	5	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	6	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	7	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
Specific input	8	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
	9	32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
	10	48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
	11	64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49

Health bit status

The status of the peer cop health bit indicates the current communications status of its assigned node. A health bit will be set when the associated node accepts input for its peer cop data block or when it receives a signal that another node has accepted specific output data from its peer cop output data block. A health bit will be deleted when the associated data block did not take up any communication within the configured peer cop health timeout period.

All health bits will be deleted when interface command "put peer cop" is executed during PLC startup. The table values become valid when the Token has been completely bypassed, after the interface command "put peer cop" has been carried out. The health bit of a specific node is always zero when the assigned peer cop input is zero.

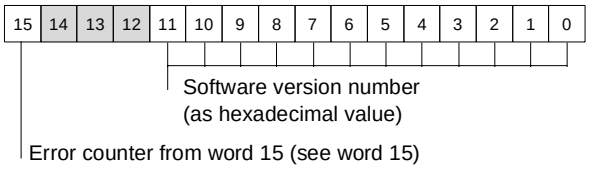
Modbus Plus network statistics

Modbus Plus network statistics

The following table shows the statistics available on Modbus Plus. You can obtain this data by running the corresponding MBP_MSTR operation (Modbus function codes 8).

Note: If you edit the "Clear local statistics" or "Clear remote statistics" operation, only words 13 to 22 in the statistics table are cleared.

Modbus Plus network statistics:

Word	Bits	Meaning
00		Node type ID
	0	Unknown node type
	1	PLC node
	2	Modbus bridge node
	3	Host computer node
	4	Bridge Plus node
	5	Peer I/O node
01	0 ... 11	Software version number as hexadecimal value (to read this, isolate bits 12-15 from the word)
	12 ... 14	Reserved
	15	Defines error counters from word 15. The high bit defines the use of error counters in word 15. The low half of the high value byte together with the low value byte contain the software version. 
02		Network address of this station

Word	Bits	Meaning
03		MAC status variable:
	0	Startup status
	1	Offline status indicator signals
	2	Duplicated offline status
	3	Idle status
	4	Token utilization status
	5	Work response status
	6	Token transfer status
	7	Response request status
	8	Status check of transfer
	9	Token request status
	10	Response request status
04		Peer status (LED code); indicates status of this device relative to the network:
	0	Monitor connect operation
	32	Normal connect operation
	64	Never receives token
	96	Single station
	128	Duplicate station
05		Token transfer counter; incremented every time this station receives the token
06		Token cycle time in ms
07	LOW	Bitmap data master failure during token ownership
	HIGH	Bitmap program master failure during token ownership
08	LOW	Bitmap activity token ownership of the data master
	HIGH	Bitmap activity token ownership of the program master
09	LOW	Bitmap activity token ownership of the data slave
	HIGH	Bitmap activity token ownership of the program slave
10	LOW	
	HIGH	Bitmap transfer request command data slave/slave poll
11	LOW	Bitmap response transfer request program master/master poll
	HIGH	Bitmap transfer request command program slave/slave poll
12	LOW	Bitmap connect status of the program master
	HIGH	Bitmap automatic logout of program slave

Word	Bits	Meaning
13	LOW	Pretransfer delay error counter
	HIGH	Receive buffer DMA overrun error counter
14	LOW	Reception count repeat command
	HIGH	Error counter data block size
15		If bit 15 of word 1 is not set, word 15 has the following significance:
	LOW	Receiver error count collision abort
	HIGH	Receive error count alignment
		If bit 15 of word 1 is set, word 15 has the following significance:
	LOW	Data block error on cable B
	HIGH	Data block error on cable B
16	LOW	CRC receiver error count
	HIGH	Error counter wrong packet length
17	LOW	Error counter wrong link address
	HIGH	Error counter DMA underflow transfer buffer storage
18	LOW	Error counter wrong internal packet length
	HIGH	Error counter wrong MAC function code
19	LOW	Communication retry counter
	HIGH	Error counter communication failed
20	LOW	Counter package receipt successful
	HIGH	Error counter no response receipt
21	LOW	Error counter unexpected response receipt
	HIGH	Error counter unexpected path
22	LOW	Error counter unexpected response
	HIGH	Error counter skipped transaction
23	LOW	Bitmap active station table, nodes 1 through 8
	HIGH	Bitmap active station table, nodes 9 through 16
24	LOW	Bitmap active station table, nodes 17 through 24
	HIGH	Bitmap active station table, nodes 25 through 32
25	LOW	Bitmap active station table, nodes 33 through 40
	HIGH	Bitmap active station table, nodes 41 through 48
26	LOW	Bitmap active station table, nodes 49 through 56
	HIGH	Bitmap active station table, nodes 57 through 64
27	LOW	Bitmap token station table, nodes 1 through 8
	HIGH	Bitmap token station table, nodes 9 through 16

Word	Bits	Meaning
28	LOW	Bitmap token station table, nodes 17 through 24
	HIGH	Bitmap token station table, nodes 25 through 32
29	LOW	Bitmap token station table, nodes 33 through 40
	HIGH	Bitmap token station table, nodes 41 through 48
30	LOW	Bitmap token station table, nodes 49 through 56
	HIGH	Bitmap token station table, nodes 57 through 64
31	LOW	Global data existence bitmap table, nodes 1 through 8
	HIGH	Global data existence bitmap table, nodes 9 through 16
32	LOW	Global data existence bitmap table, nodes 17 through 24
	HIGH	Global data existence bitmap table, nodes 25 through 32
33	LOW	Global data existence bitmap table, nodes 33 through 40
	HIGH	Global data existence bitmap table, nodes 41 through 48
34	LOW	Global data existence bitmap table, nodes 49 through 56
	HIGH	Global data existence bitmap table, nodes 57 through 64
35	LOW	Bitmap receive buffer used, buffers 1 through 8
	HIGH	Bitmap receive buffer used, buffers 9 through 16
36	LOW	Bitmap receive buffer used, buffers 17 through 24
	HIGH	Bitmap receive buffer used, buffers 25 through 32
37	LOW	Bitmap receive buffer used, buffers 33 through 40
	HIGH	Counter of activated processed commands for station administration
38	LOW	Counter for command activation, output path 1 of data master
	HIGH	Counter for command activation, output path 2 of data master
39	LOW	Counter for command activation, output path 3 of data master
	HIGH	Counter for command activation, output path 4 of data master
40	LOW	Counter for command activation, output path 5 of data master
	HIGH	Counter for command activation, output path 6 of data master
41	LOW	Counter for command activation, output path 7 of data master
	HIGH	Counter for command activation, output path 8 of data master
42	LOW	Counter for command processing, input path 41 of data slave
	HIGH	Counter for command processing, input path 42 of data slave
43	LOW	Counter for command processing, input path 43 of data slave
	HIGH	Counter for command processing, input path 44 of data slave
44	LOW	Counter for command processing, input path 45 of data slave
	HIGH	Counter for command processing, input path 46 of data slave

Word	Bits	Meaning
45	LOW	Counter for command processing, input path 47 of data slave
	HIGH	Counter for command processing, input path 48 of data slave
46	LOW	Counter for command activation, output path 81 of program master
	HIGH	Counter for command activation, output path 82 of program master
47	LOW	Counter for command activation, output path 83 of program master
	HIGH	Counter for command activation, output path 84 of program master
48	LOW	Counter for command activation, output path 85 of program master
	HIGH	Counter for command activation, output path 86 of program master
49	LOW	Counter for command activation, output path 87 of program master
	HIGH	Counter for command activation, output path 88 of program master
50	LOW	Counter for command processing, input path C1 of program slave
	HIGH	Counter for command processing, input path C2 of program slave
51	LOW	Counter for command processing, input path C3 of program slave
	HIGH	Counter for command processing, input path C4 of program slave
52	LOW	Counter for command processing, input path C5 of program slave
	HIGH	Counter for command processing, input path C6 of program slave
53	LOW	Counter for command processing, input path C7 of program slave
	HIGH	Counter for command processing, input path C8 of program slave

TCP/IP Ethernet network statistics

TCP/IP Ethernet network statistics

A TCP/IP Ethernet plugboard replies to the "Get local statistics" and "Set local statistics" commands using the following information:

Word	Meaning
00 - 02	MAC address e.g. MAC address 00 00 54 00 12 34 is displayed as follows: **Word** **Contents** 00 00 00 01 00 54 02 34 12
03	Plugboard status: 0x0001 = Running 0x4000 = APPI LED (1=ON, 0 = OFF) 0x8000 = LED Connection
04 and 05	Number of receiver interrupts
06 and 07	Number of transfer interrupts
08 and 09	Transfer timeout error count
10 and 11	Collision detection error count
12 and 13	Omitted packets
14 and 15	Memory error count
16 and 17	Number of groove restarts performed by the driver
18 and 19	Receive framing error count
20 and 21	Overflow error count receiver
22 and 23	Receive CRC error counter
24 and 25	Receive buffer error counter
26 and 27	Transfer buffer error counter
28 and 29	Transfer bin underflow counter
30 and 31	Late collision counter
32 and 33	Lost carrier counter
34 and 35	Number of retries

Word	Meaning
36 and 37	IP address e.g. the IP address 198.202.137.113 (or c6 CA 89 71) is represented as follows: **Word** **Contents** 36 89 71 37 C6 CA

Runtime errors

Runtime errors

In the event that an error occurs during an MSTR operation, a hexadecimal error code is displayed in the 4x+1 register of the control block (CONTROL).

Function error codes are network-specific:

- Modbus Plus and SY/MAX Ethernet Error Codes (See *Modbus Plus and SY/MAX Ethernet Error Codes, p. 96*)
- SY/MAX-specific error codes (See *SY/MAX-specific error codes, p. 98*)
- TCP/IP Ethernet error codes (See *TCP/IP Ethernet error codes, p. 100*)
- CTE error codes for SY/MAX and TCP/IP Ethernet (See *CTE error codes for SY/MAX and TCP/IP Ethernet, p. 103*)

Modbus Plus and SY/MAX Ethernet Error Codes

Form of the function error code

Function error codes for Modbus Plus and SY/MAX Ethernet transactions appear as **Mmss**, where:

- **M** is the high code
- **m** is the low code
- **ss** is a subcode

Hexadecimal error code

Hexadecimal error code for Modbus Plus and SY/MAX Ethernet:

Hex. error code	Meaning
1001	Abort by user
2001	An operation type that is not supported was 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	Illegal value in the length field of the control block
2004	Illegal value in the offset field of the control block
2005	Illegal 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 own address
200A	Attempting to retrieve more global data words than available
30ss	Unusual response by Modbus slave (See *ss hexadecimal value in 30ss error code, p. 97*)
4001	Inconsistent response by Modbus slave
5001	Inconsistent response by network
6mss	Routing path error (See *ss hexadecimal value in 6mss error code, p. 97*) Subfield m shows where the error occurred (a 0 value means local node, 2 means 2nd device in route, etc).

ss hexadecimal value in 30ss error code

ss hexadecimal value in 30ss error code:

ss hex. value	Meaning
01	Slave does not support requested operation
02	Non-existent slave registers were requested
03	An unauthorized data value was requested
05	Slave has accepted a lengthy program command
06	Function cannot currently be carried out: lengthy command running
07	Slave has rejected lengthy program command

ss hexadecimal value in 6mss error code

Note: Subfield m in error code 6mss is an index in the routing information that shows where an error has been detected (a 0 value indicates the local node, 2 means the second device in the route, etc.).

The ss subfield in error code 6mss is as follows:

ss hexadecimal value	Meaning
01	No response receipt
02	Access to program denied
03	Node out of service and unable to communicate
04	Unusual response received
05	Router-node data path busy
06	Slave out of order
07	Wrong destination address
08	Unauthorized node type in routing path
10	Slave has rejected the command
20	Slave has lost an activated transaction
40	Unexpected master output path received
80	Unexpected response received
F001	Wrong destination node specified for MSTR operation

SY/MAX-specific error codes

SY/MAX-specific error codes

When utilizing SY/MAX Ethernet, three additional types of errors may appear in the 4x+1 register of the control block (CONTROL).

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 HEX error code

SY/MAX-specific HEX error code:

Hex. error code	Meaning
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 read only register found by the SY/MAX remote device
F710	Receiver overrun 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 timeout 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 read only register found by the SY/MAX server
F720	Receiver overrun 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 timeout 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
7301	Invalid opcode in an MSTR block request from the Quantum translator
7303	Read/Write QSE module status (200 route address out of range)

Hex. error code	Meaning
7309	Attempt to write to a read only 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, you must reset the QSE module in order for the modifications to become effective. ● When using an MSTR instruction no valid slot or drop has been specified

TCP/IP Ethernet error codes

TCP/IP Ethernet error codes

An error in an MSTR routine via TCP/IP Ethernet may produce one of the following errors in the MSTR control block:

The error code appears as **Mmss**, where:

- **M** is the high code
- **m** is the low code
- **ss** is a subcode

HEX error codes TCP/IP Ethernet

HEX error codes TCP/IP Ethernet:

Hex. Error code	Meaning
1001	Abort by user
2001	An operation type that is not supported was 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
3000	Generic Modbus failure code
30ss	Exceptional response by Modbus slave (See *ss hexadecimal value in 30ss error code, p. 100*)
4001	Inconsistent response by Modbus slave

ss hexadecimal value in 30ss error code

ss hexadecimal value in 30ss error code:

ss hex. Value	Meaning
01	Slave does not support requested operation
02	Non-existing slave registers were requested
03	An unauthorized data value was requested
05	Slave has accepted a lengthy program command
06	Function cannot currently be carried out: lengthy command running
07	Slave has rejected lengthy program command

HEX error codes TCP/IP Ethernet network

An error on the TCP/IP Ethernet network itself may produce one of the following errors in the 4x+1 register of the control block (CONTROL).

HEX error codes TCP/IP Ethernet network:

Hex. Error code	Meaning
5004	Interrupted system invocation
5005	I/O error
5006	No such address
5009	The socket descriptor is not valid
500C	Not enough storage space
500D	Authorization denied
5011	Entry exists
5016	An argument is not valid
5017	An internal table has no more space
5020	There is interference on the connection
5023	This operation would be blocking and the socket is non-blocking
5024	The socket is non-blocking and the connection cannot be closed down
5025	The socket is non-blocking and a previous connection attempt has not been concluded
5026	Socket operation 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 at socket
502E	Protocol family not supported
F502	Address family not supported
5030	Address is already in use
5031	Address not available
5032	Network is out of order
5033	Network cannot be reached
5034	Network shut down the connection during reset
5035	The connection was terminated by the peer
5036	The connection was reset by the peer

Hex. Error code	Meaning
5037	An internal buffer is required, but cannot be assigned
5038	The socket is already connected
5039	The socket is not connected
503A	Cannot transmit after the socket has been shut off
503B	Too many references; cannot splice
503C	Connection timed out
503D	The connection attempt was denied
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	The MTU is not valid
5048	The hardware length is not valid
5049	The route specified cannot be found
504A	Collision when invoking Select; these conditions have already been selected by another job
504B	The job ID is not valid
F001	In reset mode

CTE error codes for SY/MAX and TCP/IP Ethernet

CTE error codes for SY/MAX and TCP/IP Ethernet

The following error codes are displayed in the 4x+1 register of the control block (CONTROL) if there is a problem with the Ethernet configuration extension table (CTE) in your program configuration.

CTE error codes for SY/MAX and TCP/IP Ethernet:

Hex. Error code	Meaning
7001	There is no Ethernet configuration extension
7002	The CTE is not available for access
7003	The offset is not valid
7004	Offset + length are not valid
7005	Bad data field in the CTE

MODBUSP_ADDR: Modbus Plus Address

12

Overview

At a Glance

This chapter describes the MODBUSP_ADDR block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block enables the input of Modbus Plus addressed for the READ_REG, CREAD_REG, WRITE_REG and CWRITE_REG function blocks. The address is transferred in the form of a data structure.

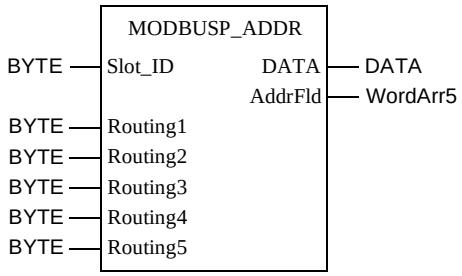
EN and ENO can be projected as additional parameters.

Note: You must be familiar with your network when programming the MODBUSP_ADDR function block. Modbus Plus routing path structures are described in detail in "Modbus Plus Network Planning and Installation Guide".

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
Slot_ID	BYTE	Slot ID Slots of the NOM module
Routing1	BYTE	Routing 1 is used for address specification (routing path addresses one of five) of the destination node during network transfer. The last byte in the routing path that is not zero is the destination node.
Routing2	BYTE	Routing2
Routing3	BYTE	Routing3
Routing4	BYTE	Routing4
Routing5	BYTE	Routing5
AddrFld	WordArr5	Data structure used to transfer the Modbus Plus address

Elementary description of WordArr5

Elementary description for WordArr5

Element	Data type	Meaning
WordArr5[1]	WORD	Routing register 1 Low value byte: used for address specification (routing path addresses one of five) of a destination node during network transfer. The last byte in the routing path that is not zero is the destination node. High value byte: Slot of the network adapter module (NOM), if any.
WordArr5[2]	WORD	Routing register 2
WordArr5[3]	WORD	Routing register 3
WordArr5[4]	WORD	Routing register 4
WordArr5[5]	WORD	Routing register 5

Detailed Description

Slot_ID

If a Modbus Plus network option module (NOM) in the rack of a Quantum controller is addressed as the destination node, the value at the Slot_ID input represents the physical NOM slot, i.e. if the NOM is plugged in at Slot 7 of the rack, the value appears as follows:

0	0	0	0	0	1	1	1

Routing x

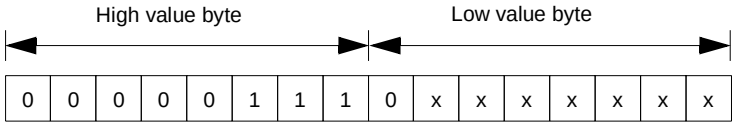
The Routing x input is used for address specification (routing path addresses one from five) of the destination node during network transfer. The last byte in the routing path that is not zero is the destination node.

0	x	x	x	x	x	x	x

Destination address (binary value between 1 and 64 (normal) or 65 > 255 (extended))

Routing register 1

If a Modbus Plus network option module (NOM) in the rack of a Quantum controller is addressed as destination node, the value in the more significant byte represents the physical slot of the NOM, i.e. if the NOM is inserted in slot 7 of the rack, the more significant byte of control register 1 looks as follows:



High value byte Slots 1 ... 16

Low value byte Destination address (binary value between 1 and 64 (normal) or 65 > 255 (extended))

PORTSTAT: Modbus Port Status

13

Overview

Introduction

This chapter describes the PORTSTAT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is used to read the status information of a local Modbus port.

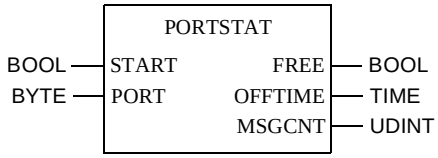
This provides the following information:

- Counter status
- Availability of the Modbus ports

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
START	BOOL	1 (TRUE) = Status information about the selected Ports (PORT) is given to the outputs. 0 (FALSE) = Outputs are set to 0.
PORT	BYTE	1 = Local Modbus port No. 1 (for Quantum, Compact, Momentum) 2 = Local Modbus port No. 2 (only for Momentum) **Note:** Other values are invalid, the outputs are set to 0 in this case.
FREE	BOOL	1 (TRUE) = Port is inactive, i.e. not in use. 0 (FALSE) = Port is in use, e.g. by a XXMIT or RTXMIT block; or is it currently in use as a communication interface to an external Modbus master (MMI, SCADA, ...).
OFFTIME	TIME	Gives the elapsed time (in ms) during which the port was continuously inactive.
MSGCNT	UDINT	Number of external Modbus Master requests

READ_REG: Read register

14

Overview

Introduction

This chapter describes the READ_REG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

If requested, this function block will read a register area once (rising edge of the REQ input). It reads data from an addressed slave via Modbus Plus, TCP/IP-Ethernet or SY/MAX-Ethernet.

Note: You must be familiar with the routing procedures of your network when programming a READ_REG function. Modbus Plus routing path structures will be described in detail in "Modbus Plus Network Planning and Installation Guide". If TCP/IP or SY/MAX EtherNet is im

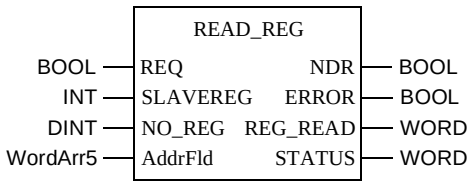
Note: For technical reasons, this function block does not allow use of the programming languages ST and IL.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
REQ	BOOL	Start read operation once
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be read from
NO_REG	INT	Number of registers to be read from slave
AddrFld	WordArr5	Data structure describing the Modbus Plus-address, TCP/IP address or SY/MAX-IP address.
NDR	BOOL	Set to "1" for one cycle after reading new data
ERROR	BOOL	Set to "1" for one scan in case of error
STATUS	WORD	Error code, see *Runtime errors, p. 95*
REG_READ	WORD	First 4x area register for read values

Elementary description for WordArr5 in Modbus Plus

Elementary description for WordArr5 in Modbus Plus:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: Routing register 1 is used for address specification (routing path addresses one of five) of the destination node during network transfer. The last byte in the routing path that is not zero is the destination node. High value byte: Slot of the network adapter module (NOM), if any.
WordArr5[2]	WORD	Routing register 2
WordArr5[3]	WORD	Routing register 3
WordArr5[4]	WORD	Routing register 4
WordArr5[5]	WORD	Routing register 5

Elementary description for WordArr5 with TCP/IP EtherNet

Elementary description for WordArr5 with TCP/IP EtherNet:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slot of the NOE module
WordArr5[2]	WORD	Byte 4 (MSB) of the 32-bit destination IP address
WordArr5[3]	WORD	Byte 3 of the 32-bit destination IP address
WordArr5[4]	WORD	Byte 2 of the 32-bit destination IP address
WordArr5[5]	WORD	Byte 1 (LSB) of the 32-bit destination IP address

Elementary description for WordArr5 with SYMAX EtherNet

Elementary description for WordArr5 with SYMAX EtherNet:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: MBP on Ethernet Transporter (MET) mapping index High value byte: Slot of the NOE module
WordArr5[2]	WORD	Destination drop number (or set to FF hex)
WordArr5[3]	WORD	Terminator (set to FF hex)
WordArr5[4]	WORD	No significance
WordArr5[5]	WORD	No significance

Function mode

Function mode of READ_REG blocks

Although a large number of READ_REG function blocks can be programmed, only four read operations may be active at the same time. In such a case it is insignificant whether they are the result of this function block or of other read operations (e.g. MBP_MSTR, MSTR, CREAD_REG). All function blocks use one data transaction path and require multiple cycles to complete a job.

Note: A TCP/IP communication between a Quantum PLC (NOE 711 00) and a Momentum PLC (all TCP/IP CPUs and all TCP/IP I/O modules) is only possible, when only **one**read or write job is carried out in every cycle. If several jobs are sent per PLC cycle, the communication stops without generating an error message in the status register of the function block.

The entire routing information is contained in data structure WordArr5 of input AddrFld. The type of function block connected to this input and thus the contents of the data structure depend on the network used.

Please use:

- Modbus Plus for function block MODBUSP_ADDR
- TCP/IP Ethernet: the function block TCP_IP_ADDR
- SY/MAX Ethernet: the function block SYMAX_IP_ADDR

Note: For experts:

The WordArr5 data structure can also be used with constants.

Parameter description

REQ	A rising edge triggers the read transaction. The parameter can be specified as direct address, located variable, unlocated variable or Literal.
SLAVEREG	Start of the area in the addressed slave from which the source data is read. The source area always resides within the 4x register area. SLAVEREG expects the source reference as offset within the 4x area. The leading "4" must be omitted (e.g. 59 (contents of the variables or value of the literal) = 40059). The parameter can be specified as direct address, located variable, unlocated variable or Literal.
NO_REG	Number of registers to be read from the addressed slave (1 ... 100). The parameter can be specified as direct address, located variable, unlocated variable or Literal.
NDR	Transition to ON state for one program cycle signifies receipt of new data ready to be processed. The parameter can be specified as direct address, located variable or unlocated variable.
ERROR	Transition to ON state for one program cycle signifies detection of a new error. The parameter can be specified as direct address, located variable or unlocated variable.
REG_READ	This word parameter addresses the first register in a series of NO_REG registers lying in series used as destination data area. The parameter must be entered as a direct address or located variable.
STATUS	Error code, see *Runtime errors, p. 95* The parameter can be specified as direct address, located variable or unlocated variable.

READREG: Read register

15

Overview

Introduction

This chapter describes the READREG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

If requested, this Function block will read a register area once (rising edge of the REQ input). It reads data from an addressed slave via Modbus Plus.

EN and ENO can be configured as additional parameters.

Note: It is necessary to be familiar with the routing procedures of your network when programming a READREG function. Modbus Plus routing path structures are described in detail in "Modbus Plus Network Planning and Installation Guide".

Note: This function block only supports the local Modbus Plus interface (no NOM).

If using a NOM please work with the block CREAD_REG.

Note: This function block does not support TCP/IP- or SY/MAX-Ethernet.

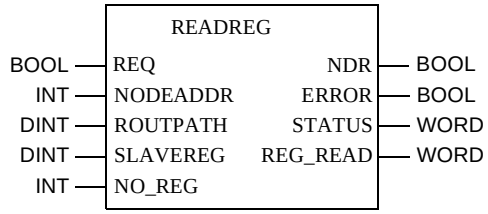
If TCP/IP- or SY/MAX-Ethernet is needed, please use the block CREAD_REG.

Note: For technical reasons, this function block does not allow the use of ST and IL programming languages.

Representation

Symbol

Block representation



Parameter description

Description of parameters:

Parameter	Data type	Meaning
REQ	BOOL	Start read operation once
NODEADDR	INT	Device address within the target segment
ROUTEPATH	DINT	Routing path to target segment
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be read from
NO_REG	INT	Number of registers to be read from slave
NDR	BOOL	Set to "1" for one cycle after reading new data
ERROR	BOOL	Set to "1" for one cycle in case of error
STATUS	WORD	Error code, see (See *Runtime errors, p. 95*)
REG_READ	WORD	First 4x area register for read values

Function mode

READREG block Function mode

Although a large number of READREG function blocks can be programmed, only four read operations may be active at the same time. In such a case it is insignificant whether they are the result of this function block or of other read operations (e.g. MBP_MSTR, MSTR, CREAD_REG). All function blocks use one data transaction path and require multiple cycles to complete a job. The status signals NDR and ERROR report the function block state to the user program.

The complete routing information must be separated into two parts:

- into the NOEADDR of the destination node (regardless of whether it is located in the local segment or in another segment) and
- the routing path, in case there is a link via bridges.

The destination address arising from this is made from these two parts of information.

The routing path is a DINT data type, which is interpreted as a sequence of two-digit information units. Appended "00" are not required (e.g. both routing paths 4711 and 47110000 are valid, for NODEADDR 34 the result is destination reference 47.11.34.00.00).

Parameter description

REQ	A rising edge triggers the read transaction. The parameter can be specified as direct address, located variable, unlocated variable or literal.
NODEADDR	Identifies the node address within the target segment. The parameter can be specified as direct address, located variable, unlocated variable or literal.
ROUTPATH	Identifies the routing path to the target segment. The two-digit information units run from 01 … 64 (see *Function mode, p. 122*). If the slave resides in the local network segment, ROUTPATH must be set to "0" or must be left unconnected. The parameter can be specified as direct address, located variable, unlocated variable or literal.
SLAVEREG	Start of the area in the addressed slave from which the source data is read. The source area always resides within the 4x register area. SLAVEREG expects the source reference as offset within the 4x area. The leading "4" must be omitted (e.g. 59 (contents of the variables or value of the literal) = 40059). The parameter can be specified as direct address, located variable, unlocated variable or literal.
NO_REG	Number of registers to be read from slave processor (1 .... 100). The parameter can be specified as direct address, located variable, unlocated variable or literal.
NDR	Transition to ON state for one program scan signifies receipt of new data ready to be processed. The parameter can be specified as direct address, located variable or unlocated variable.
ERROR	Transition to ON state for one program scan signifies detection of a new error. The parameter can be specified as direct address, located variable or unlocated variable.

STATUS	Error code, see *Runtime errors, p. 95* The parameter can be specified as direct address, located variable or unlocated variable.
REG_READ	This word parameter addresses the first register in a series of NO_REG registers lying in series used as destination data area. The parameter must be entered as a direct address or located variable.

RTXMIT: Full duplex Transfer (Compact, Momentum, Quantum)

16

At a Glance

Introduction

This chapter describes the RTXMIT function block.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function Description

The function block provides full duplex communication through the local Modbus port.

On Momentum PLCs the second local Modbus port is supported as well. The function block combines two main functions into one, these are simple message reception and simple message transmission.

Note: The RTXMIT does NOT support Modbus protocol or modem functions.

Note: EN and ENO should NOT be used with the RTXMIT, otherwise the output parameters may freeze.

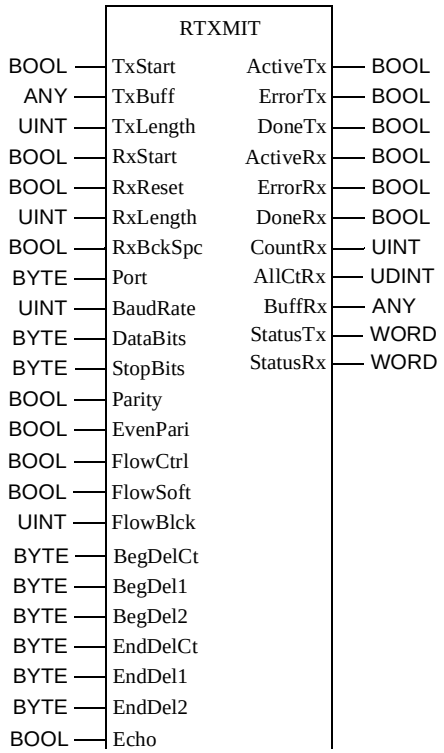
Detailed Description

The detailed description for the RTXMIT function block can be found in the XMIT-IEC User Manual.

Representation

Symbol

Representation of the Block



Parameter Description

Description of the block parameter

Parameters	Data type	Significance
TxStart	BOOL	On a rising edge (FALSE->TRUE) the EFB begins with the send operation. This operation would work concurrently to an ongoing reception. If this parameter transitions from TRUE to FALSE an ongoing transmission will be aborted without any error being generated. After a transmission process completed (with or without success) a new process won't be triggered before the next rising edge happening to TxStart.
TxBuff	ANY	A variable of any datatype, it contains the 'to be sent' character stream in Intel format.
TxLength	UINT	This parameter specifies the full amount of characters to be sent from TxBuff. Without the use of data flowcontrol (RTS/CTS or XON/XOFF), the amount of characters to be sent from TxBuff may not exceed 1024. With data flow control being activated TxLength may go as high as 2^16, as FlowBlck specifies the number of characters being transmitted with one message frame.
RxStart	BOOL	On a rising edge (FALSE->TRUE) the EFB begins with the receive operation. This operation would work concurrently to an ongoing transmission. In case this parameter carries the value TRUE after the reception process completed (DoneTx = TRUE), following characters being received won't be stored in RxBuff anymore. A new reception process won't be triggerd before the next rising edge happing to RxStart.
RxReset	BOOL	If TRUE, the following stream of characters being received will be stored at the begin of BuffRx. Also output parameter CountRx will be set to zero. At the same time current values of input parameters RxLength, Strt_Cnt, Strt_Dl1, Strt_Dl2, End_Cnt, End_Dl1, End_Dl2, RxBckSpc will be used from then on.
RxLength	UINT	Max. number of characters to be received. In case this value exceeds the size of RxBuff no error will be generated, but the size of RxBuff will be used instead. After the given number of characters has been received the output parameter DoneRx transitions to TRUE, and the receive operation will end at that time.

Parameters	Data type	Significance
RxBckSpc	BOOL	While this parameter is being set to TRUE a received character of value 8 (backspace) will cause the one character being received before the backspace to be overwritten by the character being received after the backspace. Also, in this mode the output CountRx will decrease its value with each backspace being received, till it's 0. The EFB will consider the value of RxBckSpc only while RxStart transitions from FALSE to TRUE or while RxReset is TRUE (whereby RxStart needs to be TRUE at that time).
Port	BYTE	Local port number (1 or 2) The 2nd port is supported on Momentum PLCs only. **Note:** On Momentum PLCs the EFB will switch to RS485 if the assigned port has been configured as such, otherwise the port will be run in RS232 mode.
Baudrate	UINT	Bits per second for transmission and reception, allowed values are: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200
DataBits	BYTE	Databits per transmitted and received character (8 or 7)
StopBits	BYTE	Stopbits per transmitted and received character (1 or 2)
Parity	BOOL	If TRUE, parity check will be enabled (odd or even depends on EvenPari). If FALSE no parity check will be used.
EvenPari	BOOL	If TRUE and Parity = TRUE, even parity check will be used. If FALSE and Parity = TRUE, odd parity check will be used.
FlowCtrl	BOOL	If TRUE, the next triggered transmission will consider either RTS/CTS or XON/XOFF (depends on FlowSoft)for data flow control. Receive operations won't use data flow control, since the PLC internal buffer is big enough (512 byte) to avoid losing any character between two PLC scans.
FlowSoft	BOOL	If TRUE, the data flow of transmissions will be controled by using the XON/XOFF handshaking method.

Parameters	Data type	Significance
FlowBlck	UINT	Used only if FlowCtrl equals TRUE! This parameter specifies the number of characters being sent as one frame as soon as the transmitter obtains permission to sent through the selected data flow control mechanism. If FlowBlck is set to 0 the EFB will internally use 1 instead, as this is the minimum amount of characters to be sent in one frame. If FlowBlck is set to a higher value than TxLength the EFB will internally use TxLength instead, as this is the maximum amount of characters to be sent in one frame. In order to increase data throughput (only one frame can be transmitted per PLC scan) the value assigned to FlowBlck needs to be increased.
BegDelCt	BYTE	Number of start delimiter. This parameter assigns how many characters are being used for the start delimiter. Allowed values are: 0, 1, 2. In case the value exceeds 2 the EFB won't generate an error, but would use the max. of 2 instead.
BegDel1	BYTE	This is the first (of max. 2) character of the start delimiter.
BegDel2	BYTE	This is the second (of max. 2) character of the start delimiter.
EndDelCt	BYTE	Number of end delimiter. This parameter assigns how many characters are being used for the end delimiter. Allowed values are: 0, 1, 2. In case the value exceeds 2 the EFB won't generate an error, but would use the max. of 2 instead.
EndDel1	BYTE	This is the first (of max. 2) character of the end delimiter.
EndDel2	BYTE	This is the second (of max. 2) character of the end delimiter.
Echo	BOOL	If TRUE, all characters being received during transmission will be discarded. In RS485 2-wire mode this parameter would need to be set TRUE, otherwise each just-transmitted character would be received immediately afterwards.
ActiveTx	BOOL	If TRUE, a previously initiated send operation is still ongoing.
ErrorTx	BOOL	If TRUE, a previously initiated send operation failed, StatusTx. In such case StatusTx will carry an error code that helps to identify the reason for a failure.
DoneTx	BOOL	If TRUE, a previously initiated send operation finsihed with success.
ActiveRx	BOOL	If TRUE, a previously initiated receive operation is still ongoing.
ErrorRx	BOOL	If TRUE, a previously initiated receive operation failed. In such case StatusRx will carry an error code that helps to identify the reason for a failure.

Parameters	Data type	Significance
DoneRx	BOOL	If TRUE, a previously initiated receive operation finsihed with success.
CountRx	UINT	Number of characters being received since last initiated receive operation. This output parameter will be set back to 0 after RxReset has been set to TRUE. Also this number does decrease upon reception of a backspace character in case RxBckSpc is set to TRUE.
AllCtRx	UDINT	Number of ALL characters being received since the last rising edge happened at RxStart. This output will also stay at its value after RxReset has been set to TRUE.
BuffRx	ANY	A variable of any datatype, it is used to store the received characters in Intel format.
StatusTx	WORD	Will be 0 if there's no error for the send operation, otherwise error code (See *Runtime Errors, p. 132*).
StatusRx	WORD	Will be 0 if there's no error for the receive operation, otherwise error code (See *Runtime Errors, p. 132*).

Port-Parameters

New port parameters being assigned to input parameters Port, Baudrate, DataBits, StopBits, Parity and EvenPari will only be used after both parts of the EFB (receiver and transmitter) have been shutdown (TxStart = FALSE and RxStart = FALSE) and at least one of them has been (re-)started again.

Runtime Errors

Error code (at StatusTx and StatusRx

Error code (at StatusTx and StatusRx

Error Code	Description
0	No error, either EFB is turned off completely (TxStart and RxStart are FALSE) or the ongoing process works properly.
8003 (hex)	The assigned Modbus port does not exist (>1 on Quantum and Compact, >2 on Momentum). or Another EFB is using the assigned Modbus port already.
8304 (hex)	The assigned Modbus port is used by a 984-Loadable (like XMIT).
8305 (hex)	Illegal baudrate being assigned.
8307 (hex)	Illegal number of data bits being assigned.
8308 (hex)	Illegal number of stop bits being assigned.

SYMAX_IP_ADDR: SY/MAX IP Address

17

Overview

Introduction

This chapter describes the SYMAX_IP_ADDR block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function Block enables the input of SY/MAX IP addressed for the REAG_REG, CREAD_REG, WRITE_REG and CWRITE_REG Function Blocks. The address is transferred in the form of a data structure.

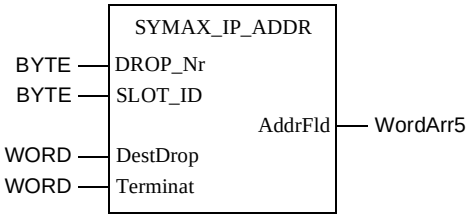
The parameters EN and ENO can additionally be projected.

Note: You must be familiar with your network when programming the SYMAX_IP_ADDR function block.

Representation

Symbol

Block representation



Parameter description

Description of parameters:

Parameter	Data type	Meaning
Drop_No	BYTE	MBP on Ethernet Transporter (MET) mapping index
Slot_ID	BYTE	Slots of the NOE module
DestDrop	WORD	Destination drop number (or set to FF hex)
Terminat	WORD	Terminator (set to FF hex)
AddrFld	WordArr5	Data structure used to transfer the SY/MAX IP address

Elementary description of WordArr5

Elementary description for WordArr5

Element	Data type	Meaning
WordArr5[1]	WORD	High value byte: Slots of the NOE module Low value byte: MBP on Ethernet Transporter (MET) mapping index
WordArr5[2]	WORD	Destination drop number (or set to FF hex)
WordArr5[3]	WORD	Terminator (set to FF hex)
WordArr5[4]	WORD	No significance
WordArr5[5]	WORD	No significance

Detailed description

Drop_No

The MBP to Ethernet Transporter (MET) mapping index is given at the Drop_Nr input, i.e. if MET is 6, the value appears as follows:

0	0	0	0	0	1	1	0

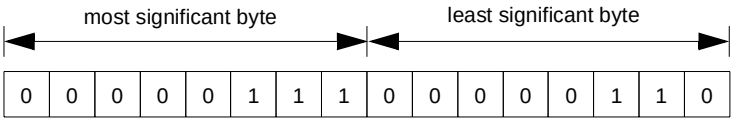
Slot_ID

If an NOE in the rack of a Quantum controller is addressed as a destination node, the value at the Slot_ID input represents the physical NOE slot, i.e. if the NOE is plugged in at Slot 7 of the rack, the value appears as follows:

0	0	0	0	0	1	1	1

AddrFld

If an NOE in the rack of a Quantum controller is addressed as a destination node, the value in the High value byte represents the physical slot of the NOE and the Low value byte represents the MBP on Ethernet Transporter (MET) mapping index, i.e. if the NOE is inserted in slot 7 of the rack and the MET mapping index is 6, the first element of the data structure looks as follows:



High value byte Slots 1 to 16

Low value byte MBP on Ethernet Transporter (MET) mapping index

TCP_IP_ADDR: TCP/IP Address

18

Overview

Introduction

This chapter describes the TCP_IP_ADDR block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function Block enables the input of TCP/IP addresses for the READ_REG, CREAD_REG, WRITE_REG and CWRITE_REG Function Blocks. The address is transferred in the form of a data structure.

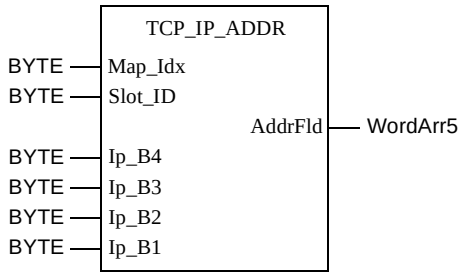
The parameters EN and ENO can additionally be projected.

Note: You must be familiar with your network when programming the TCP_IP_ADDR Function Block. The "Quantum Ethernet TCP/IP Module User Guide" provides a complete description of the TCP/IP routing.

Representation

Symbol

Block representation:



Parameter description

Description of parameters:

Parameter	Data type	Meaning
Map_Idx	BYTE	Map index MBP on Ethernet Transporter (MET) mapping index
Slot_ID	BYTE	Slot ID Slot of the NOE module
Ip_B4	BYTE	Byte 4 (MSB) of the 32-bit destination IP address
Ip_B3	BYTE	Byte 3 of the 32-bit destination IP address
Ip_B2	BYTE	Byte 2 of the 32-bit destination IP address
Ip_B1	BYTE	Byte 1 (LSB) of the 32-bit destination IP address
AddrFld	WordArr5	Data structure used to transfer the TCP/IP address

Elementary description of WordArr5

Elementary description for WordArr5

Element	Data type	Meaning
WordArr5[1]	WORD	High value byte: Slot of the NOE module Low value byte: MBP on Ethernet Transporter (MET) mapping index
WordArr5[2]	WORD	Byte 4 of the 32-bit destination IP address
WordArr5[3]	WORD	Byte 3 of the 32-bit destination IP address
WordArr5[4]	WORD	Byte 2 of the 32-bit destination IP address
WordArr5[5]	WORD	Byte 1 of the 32-bit destination IP address

Detailed Description

Map_Idx

The MBP on Ethernet Transporter (MET) mapping index is given at the Map_Idx input, i.e. if MET is 6, the value appears as follows:

0	0	0	0	0	1	1	0

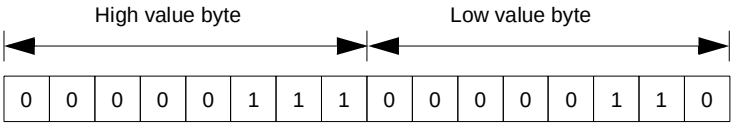
Slot_ID

If an NOE in the rack of a Quantum controller is addressed as destination node, the value at the Slot_ID input represents the physical NOE slot, i.e. if the NOE is plugged in at Slot 7 of the rack, the value appears as follows:

0	0	0	0	0	1	1	1

AddrFld

If an NOE in the rack of a Quantum controller is addressed as a destination node, the value in the High value byte represents the physical slot of the NOE and the Low value byte represents the MBP on Ethernet Transporter (MET) mapping index, i.e. if the NOE is inserted in slot 7 of the rack and the MET mapping index is 6, the first element of the data structure looks as follows:



High value byte Slots 1 ... 16

Low value byte MBP on Ethernet Transporter (MET) mapping index

WRITE_REG: Write register

19

Overview

Introduction

This chapter describes the WRITE_REG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

If requested, this Function block will write a register area once (rising edge of the REQ input). It transfers data from the PLC via Modbus Plus, TCP/IP Ethernet or SY/MAX Ethernet to an addressed slave.

EN and ENO can be configured as additional parameters.

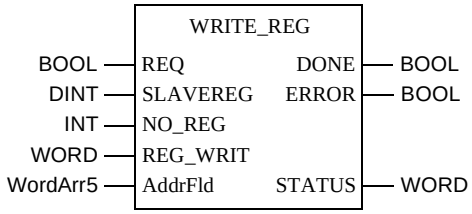
Note: You must be familiar with the routing procedures of your network when programming a WRITE_REG function. Modbus Plus routing path structures will be described in detail in "Modbus Plus Network Planning and Installation Guide".

Note: For technical reasons, this function block does not allow the use of ST and IL programming languages.

Representation

Symbol

Block representation:



Parameter description

Description of parameters:

Parameter	Data type	Meaning
REQ	BOOL	Start write operation once
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be written to
NO_REG	INT	Number of registers to be written from slave
AddrFld	WordArr5	Data structure transferring the Modbus Plus-address, TCP/IP address or SY/MAX-IP address.
REG_WRIT	WORD	First 4x register of the source data area
DONE	BOOL	Set to "1" for one scan after writing data
ERROR	BOOL	Set to "1" for one scan in case of error
STATUS	WORD	Error code, see *Runtime errors, p. 95*

Elementary description for WordArr5 in Modbus Plus

Elementary description for WordArr5 in Modbus Plus:

Element	Data type	Meaning
WordArr5[1]	WORD	Low value byte: Routing register 1 is used for address specification (routing path addresses one of five) of the destination node during network transfer. The last byte in the routing path that is not zero is the destination node. High value byte: Slot of the network adapter module (NOM), if any.
WordArr5[2]	WORD	Routing register 2
WordArr5[3]	WORD	Routing register 3
WordArr5[4]	WORD	Routing register 4
WordArr5[5]	WORD	Routing register 5

Elementary description for WordArr5 with TCP/IP EtherNet

Elementary description for WordArr5 with TCP/IP EtherNet:

Element	Data type	Meaning
WordArr5[1]	WORD	High value byte: Slot of the NOE module Low value byte: MBP on Ethernet Transporter (MET) mapping index
WordArr5[2]	WORD	Byte 4 (MSB) of the 32-bit destination IP address
WordArr5[3]	WORD	Byte 3 of the 32-bit destination IP address
WordArr5[4]	WORD	Byte 2 of the 32-bit destination IP address
WordArr5[5]	WORD	Byte 1 (LSB) of the 32-bit destination IP address

Elementary description for WordArr5 with SYMAX EtherNet

Elementary description for WordArr5 with SYMAX EtherNet:

Element	Data type	Meaning
WordArr5[1]	WORD	High value byte: Slot of the NOE module Low value byte: MBP on Ethernet Transporter (MET) mapping index
WordArr5[2]	WORD	Destination drop number (or set to FF hex)
WordArr5[3]	WORD	Terminator (set to FF hex)
WordArr5[4]	WORD	No significance
WordArr5[5]	WORD	No significance

Function mode

Function mode of the WRITE_REG module

Although a large number of WRITE_REG function blocks can be programmed, only four write operations may be active at the same time. In such a case it is insignificant whether they are the result of this function block or of other write operations (e.g. MBP_MSTR, MSTR, CWRITE_REG). All function blocks use one data transaction path and require multiple cycles to complete a job.

If several WRITE_REG function blocks are used within an application, they must at least differ in the values of their NO_REG or REG_WRITE parameters.

Note: A TCP/IP communication between a Quantum PLC (NOE 711 00) and a Momentum PLC (all TCP/IP CPUs and all TCP/IP I/O modules) is only possible, when only **one**read or write job is carried out in every cycle. If several jobs are sent per PLC cycle, the communication stops without generating an error message in the status register of the function block.

The status signals DONE and ERROR report the function block state to the user program.

The entire routing information is contained in data structure WordArr5 of input AddrFld. The type of function block connected to this input and thus the contents of the data structure depend on the network used.

Please use:

- Modbus Plus for function block MODBUSP_ADDR (See *MODBUSP_ADDR: Modbus Plus Address, p. 105*)
- TCP/IP Ethernet: the function block TCP_IP_ADDR (See *TCP_IP_ADDR: TCP/IP Address, p. 137*)
- SY/MAX Ethernet: the function block SYMAX_IP_ADDR (See *SYMAX_IP_ADDR: SY/MAX IP Address, p. 133*)

Note: For experts:

The WordArr5 data structure can also be used with constants.

Parameter description

REQ	A rising edge triggers the write transaction. The parameter can be specified as Direct address, Located variable, Unlocated variable or Literal.
SLAVEREG	Start of the destination area in the addressed slave to which the source data is written. The source area always resides within the 4x register area. SLAVEREG expects the destination reference as offset within the 4x area. The leading "4" must be omitted (e.g. 59 (contents of the variables or value of the literal) = 40059). The parameter can be specified as Direct address, Located variable, Unlocated variable or Literal.
NO_REG	Number of registers to be written to slave processor (1 ... 100). The parameter can be specified as Direct address, Located variable, Unlocated variable or Literal.
REG_WRIT	This word parameter addresses the first register in a series of NO_REG registers lying in series used as source data area. The parameter must be entered as a direct address or located variable.
DONE	Transition to ON state for one program scan signifies data have been transferred. The parameter can be specified as Direct address, Located variable or Unlocated variable .
ERROR	Transition to ON state for one program scan signifies detection of a new error. The parameter can be specified as Direct address, Located variable or Unlocated variable.
STATUS	Error code, see *Runtime errors, p. 95* The parameter can be specified as Direct address, Located variable or Unlocated variable.

WRITEREG: Write register

20

Overview

Introduction

This chapter describes the WRITEREG block.

What's in this Chapter?

This chapter contains the following topics:

Short description

Function description

If requested, this function block will write a register area once (rising edge of the REQ input). It transfers data from the PLC via Modbus Plus to an addressed slave.

EN and ENO can be configured as additional parameters.

Note: It is necessary to be familiar with the routing procedures of your network when programming a WRITEREG function. Modbus Plus routing path structures will be described in detail in "Modbus Plus Network Planning and Installation Guide".

Note: This function block only supports the local Modbus Plus interface (no NOM).

If using a NOM please work with the block WRITE_REG.

Note: This function block does not support TCP/IP or SY/MAX Ethernet.

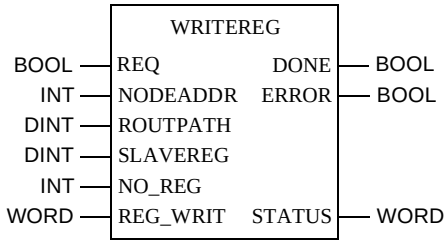
If TCP/IP- or SY/MAX-Ethernet is needed, please use the block WRITE_REG.

Note: For technical reasons use of the programming languages ST and IL is not allowed by this function block

Representation

Symbol

Representation of the block:



Parameter description

Description of the block parameter:

Parameter	Data type	Meaning
REQ	BOOL	Start write operation once
NODEADDR	INT	Device address within the target segment
ROUTEPATH	DINT	Routing path to target segment
SLAVEREG	DINT	Offset address of the first 4x register in the slave to be written to
NO_REG	INT	Number of registers to be written from slave
REG_WRIT	WORD	First 4x register of the source data area
DONE	BOOL	Set to "1" for one scan after writing data
ERROR	BOOL	Set to "1" for one scan in case of error
STATUS	WORD	Error code, see *Runtime errors, p. 95*

Function mode

Function mode of WRITEREG blocks

Although a large number of WRITEREG function blocks can be programmed, only four write operations may be active at the same time. In such a case it is insignificant whether they are the result of this function block or of other write operations (e.g. MBP_MSTR, MSTR, CWRITE_REG). All function blocks use one data transaction path and require multiple cycles to complete a job.

If several WRITEREG function blocks are used within an application, they must at least differ in the values of their NO_REG or REG_WRITE parameters.

The status signals DONE and ERROR report the function block state to the user program.

The complete routing information must be separated into two parts:

- into the NODEADDR of the destination node (regardless of whether it is located in the local segment or in another segment) and
- the routing path, in case there is a link via network bridges.

The destination address arising from this is made from these two items of information.

The routing path is a DINT data type, which is interpreted as a sequence of two-digit information units. Appended "00" are not required (e.g. both routing paths 4711 and 47110000 are valid, for NODEADDR 34 the result is destination reference 47.11.34.00.00).

Parameter description

REQ	A rising edge triggers the write transaction. The parameter can be specified as direct address, located variable, unlocated variable or Literal.
NODEADDR	Identifies the node address within the target segment. The parameter can be specified as direct address, located variable, unlocated variable or Literal.
ROUTPATH	Identifies the routing path to the target segment. The two-digit information units run from 01 … 64 (see *Function mode, p. 150*). If the slave resides in the local network segment, ROUTPATH must be set to "0" or must be left unconnected. The parameter can be specified as direct address, located variable, unlocated variable or Literal.
SLAVEREG	Start of the destination area in the addressed slave to which the source data is written. The source area always resides within the 4x register area. SLAVEREG expects the destination reference as offset within the 4x area. The leading "4" must be omitted (e.g. 59 (contents of the variables or value of the literal) = 40059). The parameter can be specified as direct address, located variable, unlocated variable or Literal.
NO_REG	Number of registers to be written to slave processor (1 ... 100). The parameter can be specified as direct address, located variable, unlocated variable or Literal.
REG_WRIT	This word parameter addresses the first register in a series of NO_REG registers lying in series used as source data area. The parameter must be entered as a direct address or located variable.
DONE	Transition to ON state for one program scan signifies data have been transferred. The parameter can be specified as direct address, located variable or unlocated variable.

ERROR	Transition to ON state for one program scan signifies detection of a new error. The parameter can be specified as direct address, located variable or unlocated variable.
STATUS	Error code, see (See *Runtime errors, p. 95*) The parameter can be specified as direct address, located variable or unlocated variable.

XMIT: Transmit (Momentum)

21

At a Glance

Introduction

This chapter describes the XMIT function block.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function Description

The XMIT (Transmit) function block sends Modbus messages from a "master" PLC to multiple slave PLCs or sends ASCII character strings from the PLC's Modbus slave port#1 or port#2 to ASCII printers and terminals. XMIT sends these messages over telephone dialup modems, radio modems, or simply direct connection.XMIT comes with three modes: a communication mode, port status mode and a conversion mode. XMIT performs general ASCII input functions in the communication mode including simple ASCII and terminated ASCII. You may import and export ASCII or binary data into your PLC and convert it into various binary data or ASCII to send to DCE devices based upon the needs of your application. The block has builtin diagnostics that checks to make sure no other XMIT blocks are active in the PLC on the same port. Within the XMIT block a control table allows you to control the communications link between the PLC and DCE (Data Communication Equipment) devices attached to Modbus port #1 or port#2 of the PLC. The XMIT block does NOT activate the port LED when it is transmitting data.Remember, the Modbus protocol is a "master/slave" protocol. Modbus is designed to have only one master polling multiple slaves. Therefore, when using the XMIT block in a network with multiple masters, contention resolution and collision avoidance is your responsibility and may easily be addressed through ladder logic programming. paragraph of overview block.

EN and ENO can be configured as additional parameters

Using Modbus

Remember, the Modbus protocol is a "master/slave" protocol. Modbus is designed to have only one master polling multiple slaves. Therefore, when using the XMIT block in a network with multiple masters, contention resolution and collision avoidance is your responsibility and may easily be addressed through user logic programming.

Restrictions

This function block controls Modbus port #1 and #2 of the Momentum CPUs.It can be used with the stripped exec onlyThe XMIT function block works just as its LL984 counterpart, but without ASCII string conversion, copy and compare functions and without the Port Status functions.

Software and Hardware Required

When using the Momentum PLCs the XMIT function block it is a builtin.

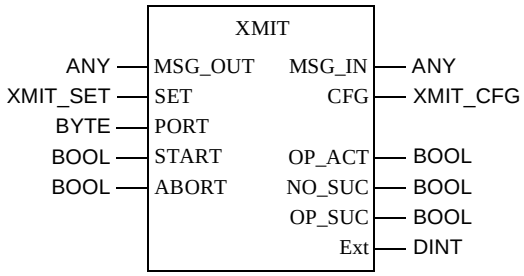
Detailed Description

The detailed description for the XMIT function block can be found in the XMIT-IEC User Manual.

Representation

Symbol

Representation of the block



Parameter Description

Description of the block parameter

Parameters	Data type	Meaning
SET	XMIT_SET	Data structure for the XMIT configuration
MSG_OUT	ANY	Message to be sent (must be in 4x range)
PORT	BYTE	Selection of communications interface
START	BOOL	1: Starts XMIT operation
ABORT	BOOL	1: Aborts current XMIT operation
MSG_IN	ANY	Incoming message (must be in 4x range)
CFG	XMIT_CFG	Data structure with all components of the XMIT configuration, including the automatically set and not used variables. Only for display and must be in 4x range.
OP_ACT	BOOL	1: XMIT operation in progress
NO_SUC	BOOL	1: There is an error or the current XMIT operation is aborted.
OP_SUC	BOOL	1: XMIT operation successfully completed
Ext	DINT	not presently in use

XMIT_SET Data Structure

Description of data structure

Element	Data type	Meaning
BaudRate	WORD	This component corresponds to the 4x+3 register (data rate) of the LL984 XMIT instruction.
DataBits	BYTE	This component corresponds to the 4x+4 register (data bits) of the LL984 XMIT instruction.
Parity	BYTE	This component corresponds to the 4x+5 register (parity) of the LL984 XMIT instruction.
StopBits	BYTE	This component corresponds to the 4x+6 register (stop bits) of the LL984 XMIT instruction.
CommandWord	WORD	This component corresponds to the 4x+8 register (command word) of the LL984 XMIT instruction.
MessageLen	WORD	This component corresponds to the 4x+10 register (message length) of the LL984 XMIT instruction. (In case of a terminated ASCII receipt, this component will be set automatically.)
RespTimeOut	WORD	This component corresponds to the 4x+11 register (response time-out (ms)) of the LL984 XMIT instruction.
RetryLimit	WORD	This component corresponds to the 4x+12 register (retry limit) of the LL984 XMIT instruction.
XmStartDelay	WORD	This component corresponds to the 4x+13 register (start of transmission delay (ms)) of the LL984 XMIT instruction.
XmEndDelay	WORD	This component corresponds to the 4x+14 register (end of transmission delay (ms)) of the LL984 XMIT instruction.

XMIT_CFG Data Structure

Description of data structure

Element	Data type	Meaning
FaultStatus	WORD	This component corresponds to the 4x+1 register (fault status) of the LL984 XMIT instruction.
UserAvail_1	WORD	This component corresponds to the 4x+2 register (available to user) of the LL984 XMIT instruction.
BaudRate	WORD	This component corresponds to the 4x+3 register (data rate) of the LL984 XMIT instruction.
DataBits	WORD	This component corresponds to the 4x+4 register (data bits) of the LL984 XMIT instruction.
Parity	WORD	This component corresponds to the 4x+5 register (parity) of the LL984 XMIT instruction.
StopBits	WORD	This component corresponds to the 4x+6 register (stop bits) of the LL984 XMIT instruction.
UserAvail_2	WORD	This component corresponds to the 4x+7 register (available to user) of the LL984 XMIT instruction.
CommandWord	WORD	This component corresponds to the 4x+8 register (command word) of the LL984 XMIT instruction.
MessagePtr	WORD	This component corresponds to the 4x+9 register (message pointer) of the LL984 XMIT instruction.
MessageLen	WORD	This component corresponds to the 4x+10 register (message length) of the LL984 XMIT instruction.
RespTimeOut	WORD	This component corresponds to the 4x+11 register (response time-out (ms)) of the LL984 XMIT instruction.
RetryLimit	WORD	This component corresponds to the 4x+12 register (retry limit) of the LL984 XMIT instruction.
XmStartDelay	WORD	This component corresponds to the 4x+13 register (start of transmission delay (ms)) of the LL984 XMIT instruction.
XmEndDelay	WORD	This component corresponds to the 4x+14 register (end of transmission delay (ms)) of the LL984 XMIT instruction.
CurrentRetry	WORD	This component corresponds to the 4x+15 register (current retry) of the LL984 XMIT instruction.

XXMIT: Transmit (Compact, Momentum, Quantum)

22

At a Glance

Introduction

This chapter describes the XXMIT function block.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function Description

The XXMIT (Transmit) function block sends Modbus messages from a "master" PLC to multiple slave PLCs or sends ASCII character strings from the PLC's Modbus slave port#1 (on Momentum PLCs also port#2 is supported) to ASCII printers and terminals. XXMIT sends these messages over telephone dialup modems, radio modems, or simply direct connections. XXMIT performs general ASCII input functions in the communication mode including simple ASCII and terminated ASCII. You may import and export ASCII or binary data into your PLC. The block has builtin diagnostics that checks to make sure no other XXMIT blocks are active in the PLC on the same port. Within the XXMIT block control inputs allows you to control the communications link between the PLC and DCE (Data Communication Equipment) devices attached to Modbus port #1 or port#2 of the PLC. The XXMIT block does NOT activate the port LED when it is transmitting data.

Note: EN and ENO should NOT be used with the XXMIT, otherwise the output parameters may freeze.

Restrictions

The following restrictions apply to the XXMIT function block:

XXMIT does not support::

- ASCII string conversion
- copy and compare functions
- Port Status functions

Note: Momentum only supports one Stopbit.

Note: Port 2 only supported by Momentum PLCs

Software and Hardware Required

Software

The XXMIT function block requires the following software

- A minimum of Concept 2.2 Service Release 2
- IEC exec version

Hardware

The following hardware is **not** supported by the XXMIT function block:

- PLCs which do not support IEC languages
- Soft PLC
- All Atrium PLCs
- IEC Simulator

Memory Requirements

The usage of one or more XXMIT EFBs in an IEC application consumes approximately 15.5 KByte program (code) memory.For each instance of this EFB included in the user program, additional data memory between 2.5 and 3 Kbyte is allocated.

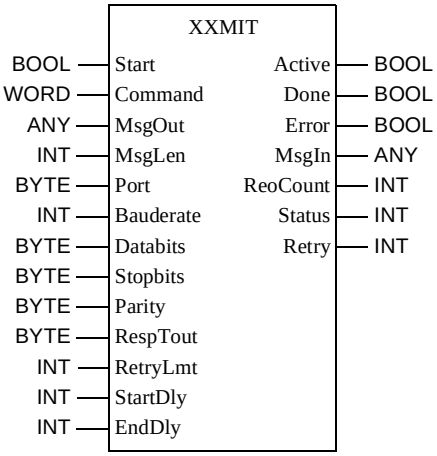
Detailed Description

The detailed description for the XXMIT function block can be found in the XMIT-IEC User Manual.

Representation

Symbol

Representation of the Block

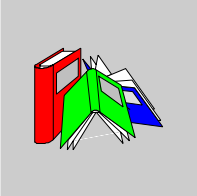


Parameter Description

Description of the block parameter

Parameters	Data type	Significance
Start	BOOL	Value of 1 starts XXMIT operation
Command	WORD	Specifies the command to be performed
MsgOut	ANY	Message to be sent
MsgLen	INT	Message length of output message
Port	BYTE	Selection of communications interface
Baudrate	INT	Baudrate
Databits	BYTE	Databits
Stopbits	BYTE	Stopbits
Parity	BYTE	Parity
RespTout	INT	Time to wait for a valid response
RetryLmt	INT	Number of retries until receiving a valid response
StartDly	INT	Waiting time before message transmit.
EndDly	INT	Waiting time after message transmit
Active	BOOL	Value of 1 indicates that an XXMIT operation is in progress
Done	BOOL	Value of 1 indicates that the XXMIT operation has been completed successfully
Error	BOOL	Value of 1 indicates that an error has ocured or that the current XXMIT operation is terminated
MsgIn	ANY	Incoming message
RecCount	INT	Displaythe number of received characters
Status	INT	Display a fault code generated by the XXMIT block
Retry	INT	Indicates the current number of retry attempts made by the XXMIT block

Glossary



A

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

Actual Parameters — Current connected Input / Output Parameters.

Addresses — (Direct) addresses are memory ranges in the PLC. They are located in the State RAM and can be assigned Input/Output modules.

The display/entry of direct addresses is possible in the following formats:

- Standard Format (400001)
- Separator Format (4:00001)
- Compact format (4:1)
- IEC Format (QW1)

ANL_IN — ANL_IN stands for the "Analog Input" data type and is used when processing analog values. The 3x-References for the configured analog input module, which were specified in the I/O component list, are automatically assigned data types and should therefore only be occupied with Unlocated Variables.

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

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

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

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

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

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

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

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

Argument Synonymous with Actual parameters.

ASCII-Mode The ASCII (American Standard Code for Information Interchange) mode is used to communicate with various host devices. ASCII works with 7 data bits.

Atrium The PC based Controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module has a motherboard (requiring SA85 driver) with two slots for PC104 daughter-boards. In this way, one PC104 daughter-board is used as a CPU and the other as the INTERBUS controller.

B

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

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

Example

16#F_F or 16#FF (decimal 255)

16#E_0 or 16#E0 (decimal 224)

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

Example

2#1111_1111 or 2#11111111 (decimal 255)

2#1110_0000 or 2#11100000 (decimal 224)

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

Example

8#3_77 or 8#377 (decimal 255)

8#34_0 or 8#340 (decimal 224)

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

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

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

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

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

C

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

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

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

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

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

D

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

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

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

DCP I/O drop

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

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs in Windows. The user can also use the DDE interface in the extended monitor to invoke their own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data to the PLC via the server. The user can therefore alter data directly in the PLC, while monitoring and analyzing results. When using this interface, the user can create their own "Graphic Tool", "Face Plate" or "Tuning Tool" and integrate into the system. The tools can be written in any language, i.e. Visual Basic, Visual C++, which supports DDE. The tools are invoked, when the user presses one of the buttons in the Extended Monitor dialog field. Concept Graphic Tool: Configuration signals can be displayed as a timing diagram using the DDE connection between Concept and Concept Graphic Tool.

Declaration

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

Definitions file (Concept-EFB)

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

Derived Data Type

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

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

Derived Function Block (DFB)

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

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

A distinction is made between local and global DFBs.

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

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

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

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

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

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

DP (PROFIBUS) DP = Remote Peripheral

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

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

E

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

Elementary functions/ function blocks (EFB) Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose body for example can not be modified with the DFB editor (Concept-DFB). EFB types are programmed in "C" and are prepared in a pre-compiled form using libraries.

EN / ENO (Enable / Error signal)

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

Error

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

Evaluation

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

F

FFB (Functions/ Function blocks)

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

Field variables

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

FIR Filter

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

Formal parameters

Input / Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.

Function (FUNC)

A program organization unit, which supplies an exact data element when processing. a function has no internal status information. Multiple invocations of the same function using the same input parameters always supply the same output values.

Details of the graphic form of the function invocation can be found in the "Functional block (instance)". In contrast to the invocation of the function blocks, function invocations only have a single unnamed output, whose name is the same as the function. In FBD each invocation is denoted by a unique number via the graphic block, this number is automatically generated and can not be altered.

Function block (Instance) (FB)

A function block is a program organization unit, which correspondingly calculates the functionality values that were defined in the function block type description, for the outputs and internal variable(s), if it is invoked as a certain instance. All internal variable and output values for a certain function block instance remain from one function block invocation to the next. Multiple invocations of the same function block instance with the same arguments (input parameter values) do not therefore necessarily supply the same output value(s).

Each function block instance is displayed graphically using a rectangular block symbol. The name of the function block type is stated in the top center of the rectangle. The name of the function block instance is also stated at the top, but outside of the rectangle. It is automatically generated when creating an instance, but, depending on the user's requirements, it can be altered by the user. Inputs are displayed on the left side of the block and outputs are displayed on the right side. The names of the formal input/output parameters are shown inside the rectangle in the corresponding places.

The above description of the graphic display is especially applicable to the function invocation and to DFB invocations. Differences are outlined in the corresponding definitions.

Function Block Dialog (FBD)

One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.

Function block type

A language element, consisting of: 1. the definition of a data structure, divided into input, output and internal variables; 2. a set of operations, which are performed with elements of the data structure, when a function block type instance is invoked. This set of operations can either be formulated in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (invoked) several times.

Function Number

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

n = section number (current number)

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

G

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

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

Global Data Global data are Unlocated variables.

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

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

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

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

H

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

I

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

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

IEC 61131-3

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

IEC Format (QW1)

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

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

IEC name conventions (identifier)

An identifier is a sequence of letters, numbers and underscores, which must begin with either a letter or underscore (i.e. the name of a function block type, an instance, a variable or a section). Letters of a national typeface (i.e.: ö,ü, é, õ) can be used, except in project and DFB names.

Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as two separate identifiers. Several leading and multiple successive underscores are not allowed.

Identifiers should not contain any spaces. No differentiation is made between upper and lower case, e.g. "ABCD" and "abcd" are interpreted as the same identifier.

Identifiers should not be Keywords.

IEC Program Memory

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

IIR Filter

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

Initial step

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

Initial value

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

Input bits (1x references)

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

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

Input parameter (Input)

Upon invocation of a FFB, this transfers the corresponding argument.

Input words (3x references)

An input word contains information, which originates from an external source and is represented by a 16 bit number. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 300201 signifies an input word at the address 201 in the State RAM.

Input/output marker bits (0x references)

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

Instance Name

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

FBI = Function Block Instance

n = section number (current number)

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

Instancing

Generating an Instance.

Instruction (IL)

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

Instruction (LL984)

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

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

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

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

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

Example

-12, 0, 123_456, +986

INTERBUS (PCP) The new INTERBUS (PCP) I/O drop type is entered into the Concept configurator, to allow use of the INTERBUS PCP channel and the INTERBUS process data pre-processing (PDV). This I/O drop type is assigned the INTERBUS switching module 180-CRP-660-01.

The 180-CRP-660-01 differs from the 180-CRP-660-00 only in the fact that it has a clearly larger I/O range in the control state RAM.

Invocation The process, through which an operation is carried out.

J

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

K

Keywords

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

L

Ladder Diagram (LD)

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

Ladder Logic 984 (LL)

The terms Ladder Logic and Ladder Diagram refer to the word Ladder being executed. In contrast to a circuit diagram, a ladder diagram is used by electrotechnicians to display an electrical circuit (using electrical symbols), which should show the course of events and not the existing wires, which connect the parts with each other. A usual user interface for controlling the actions of automation devices permits a Ladder Diagram interface, so that electrotechnicians do not have to learn new programming languages to be able to implement a control program.

The structure of the actual Ladder Diagram enables the connection of electric elements in such a way that generates a control output, which is dependant upon a logical power flow through used electrical objects, which displays the previously requested condition of a physical electrical device.

In simple form, the user interface is a video display processed by the PLC programming application, which sets up vertical and horizontal grid, in which programming objects are classified. The diagram contains the power grid on the left side, and when connected to activated objects, the power shifts from left to right.

Landscape

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

Language Element

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

Library

Collection of software objects, which are intended for re-use when programming new projects, or even building new libraries. Examples are the libraries of the Elementary function block types.

	EFB libraries can be divided up into Groups.
Link	A control or data flow connection between graphical objects (e.g. steps in the SFC Editor, function blocks in the FBD Editor) within a section, represented graphically as a line.
Literals	Literals are used to provide FFB inputs, and transition conditions etc using direct values. These values can not be overwritten by the program logic (read only). A distinction is made between generic and standardized literals. Literals are also used to allocate a constant, a value or a variable an initial value. Entries are made as base 2 literal, base 8 literal, basis 16 literal, integer literal, real literal or real literal with exponent.
Local derived data types	Local derived data types are only available in a single Concept project and the local DFBs and are placed in the DFB directory under the project directory.
Local DFBs	Local DFBs are only available in a single Concept project and are placed in the DFB directory under the project directory.
Local Link	The local network is the network, which connects the local nodes with other nodes either directly or through bus repeaters.
Local macros	Local macros are only available in a single Concept project and are placed in the DFB directory under the project directory.
Local network nodes	The local node is the one, which is currently being configured.
Located variable	A state RAM address (reference addresses 0x, 1x, 3x,4x) is allocated to located variables. The value of these variables is saved in the state RAM and can be modified online using the reference data editor. These variables can be addresses using their symbolic names or their reference addresses. All inputs and outputs of the PLC are connected to the state RAM. The program can only access peripheral signals attached to the PLC via located variables. External access via Modbus or Modbus Plus interfaces of the PLC, e.g. from visualization systems, is also possible via located variables.

M

Macro	Macros are created with the help of the Concept DFB software.

Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).

A distinction is made between local and global macros.

Macros have the following properties:

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

MMI Man-Machine-Interface

Multi element variables Variables to which a Derived data type defined with STRUCT or ARRAY is allocated.

A distinction is made here between field variables and structured variables.

N

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

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

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

O

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

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

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

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

P

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

PLC Memory programmable controller

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

Print-out Expressions consist of operators and operands.

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

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

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

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

General description for the complete set of programming and configuration information in the project database, which represents the source code that describes the automation of a system.

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

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

R

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

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

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

Example

-12.0, 0.0, +0.456, 3.14159_26

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

Example

-1.34E-12 or -1.34e-12

1.0E+6 or 1.0e+6

1.234E6 or 1.234e6

Reference

Every direct address is a reference that begins with an indicator, which specifies whether it is an input or an output and whether it is a bit or a word. References that begin with the code 6, represent registers in the extended memory of the state RAM.

0x range = Coils

1x range = Discrete inputs

3x range = Input registers

4x range = Output registers

6x range = Register in the extended memory

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

Register in the extended memory (6x-reference)

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

Remote Network (DIO)

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

RIO (Remote I/O)

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

RTU-Mode

Remote Terminal Unit

The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.

Runtime error

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

S

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

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

Section A section can for example be used to describe the mode of functioning of a technological unit such as a motor.

A program or DFB consists of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages may be used within a section at any one time.

Each section has its own document window in Concept. For reasons of clarity, it is however useful to divide a very large section into several small ones. The scroll bar is used for scrolling within a section.

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

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

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

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

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

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

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

Standard Format (400001)

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

Standardized literals

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

Example

INT#15 (Data type: integer, value: 15),

BYTE#00001111 (Data type: byte, value: 00001111)

REAL#23.0 (Data type: real, value: 23.0)

To assign the data type REAL, the value may also be specified in the following manner: 23.0.

Entering a comma will automatically assign the data type REAL.

State RAM

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

Status Bits

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

Step

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

Step name

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

The automatically generated step name is always formed as follows: S_n_m

S = step

n = section number (current number)

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

Structured text (ST)

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

Structured variables

Variables to which a Derived data type defined with STRUCT (structure) is allocated.

A structure is a collection of data elements with generally different data types (elementary data types and/or derived data types).

SY/MAX

In Quantum control devices, Concept includes the providing of I/O-map SY/MAX-I/O modules for remote contolling by the Quantum PLC. The SY/MAX remote backplane has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O System. The SY/MAX-I/O modules are executed for you for labelling and inclusion in the I/O map of the Concept configuration.

T

Template file (Concept-EFB)

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

TIME

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

Time literals

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

Example

t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Token

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

Total IEC memory

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

Traffic Cop

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

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

U

UDEFB User-defined elementary functions/function blocks

Functions or function blocks, which were created in the C programming language, and which Concept provides in libraries.

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

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

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

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

V

Variables Variables are used to exchange data within a section, between several sections and between the program and the PLC.

Variables consist of at least one variable name and one data type.

If a variable is assigned a direct address (reference), it is called a located variable. If the variable has no direct address assigned to it, it is called an unlocated variable. If the variable is assigned with a derived data type, it is called a multi element variable.

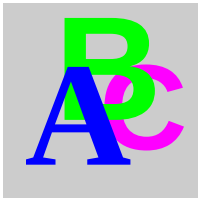
There are also constants and literals.

W

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

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

Index



I

M

P

R

S

T

W

X