Concept 2.6

IEC Block Library
Part: SYSTEM

12/2010

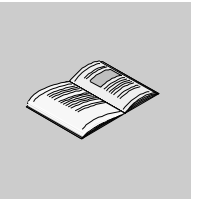
33002219.05

www.schneider-electric.com

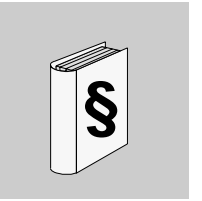
Schneider Electric

© 2010 Schneider Electric. All rights reserved.

Table of Contents



Safety Information



Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

CAUTION

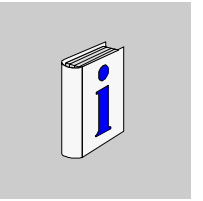
CAUTION, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage.

PLEASE NOTE

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

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

About the Book



At a Glance

Document Scope

This documentation will help you to configure the functions and function blocks.

Validity Note

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

NOTE: Additional up-to-date tips can be found in the README file for Concept.

Related Documents

You can download these technical publications and other technical information from our website at *www.telemecanique.com*

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

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

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

General Information on the SYSTEM Function Block Library

I

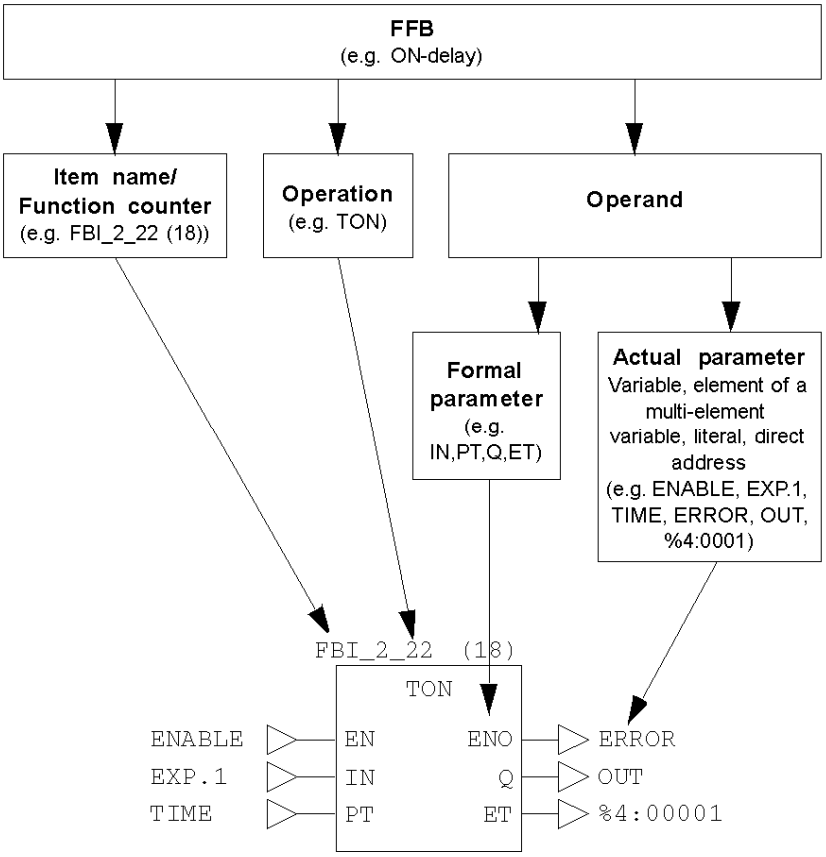
Parameterizing functions and function blocks

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.

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

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

Introduction

These EFB descriptions are documented in alphabetical order.

NOTE: The number of inputs of some EFBs can be increased to a max. of 32 through vertical size alteration of the FFB symbol. See the description of the individual EFBs to determine which EFBs.

What's in this Part?

This part contains the following chapters:

DIOSTAT: Module health status (DIO)

2

Introduction

This section describes function block DIOSTAT.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function provides the health state for I/O modules of an I/O drop (DIO).

Each I/O drop module (slot) is characterised by an output "status" bit. The bit on the far left side in "status" corresponds to the slot on the far left side of the I/O drop.

NOTE: If a module of the I/O drop is configured and works correctly, the corresponding bit is set to "1".

Additional parameters EN and ENO can be defined.

Representation

Symbol

Block representation:

Parameter description

Block parameter description:

Parameter	Data type	Meaning
LINK	UINT	Link No. (0...2)
DROP	UINT	Drop no. (1...64)
OUT	WORD	Status bit pattern of a drop

FREERUN: Free-running timer

3

Introduction

This section describes function block FREERUN.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function enables an independent counter, which can be used for run time measurement of sections and application programs.

Additional parameters EN and ENO can be defined.

Run time determination of a section

Run time determination of a section:

Step	Action
1	Place one FREERUN function at the start of the section and one at the end.
2	Via the execution sequence make sure that the FREERUN function at the start of the section is carried out first, and the one at the end of the section is carried out last.
3	Calculate delta of the two values obtained. Delta indicates the run time of the section in microseconds.

Run time determination of a program

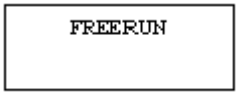
Run time determination of a program:

Step	Action
1	Place a FREERUN function at the start of the program and one at the end of the last section.
2	Via the execution sequence make sure that the FREERUN function at the start of the first section is carried out first, and the one at the end of the last section is carried out last.
3	Calculate the delta of the two values obtained. This delta indicates the run time of the program in microseconds.

Description

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
OUT	DINT	Shows the time measured since the program started in microseconds.

GET_IEC_INF: Read the IEC Status Flags

4

At a glance

This chapter describes the function block GET_IEC_INF.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block shows the new IEC system error flags from the PLC on the outputs. The average and maximum interrupt execution time in proportion to the total execution time (cycle time) of the application is also given. The average of the last 8 cycles are evaluated and shown.

All values for IEC interrupt processing are only valid for the following Quantum CPUs:

- 140 CPU 434 12
- 140 CPU 534 14

When using the simulator, the proportional and maximum execution times shown are not valid values.

Additional parameters EN and ENO can be defined.

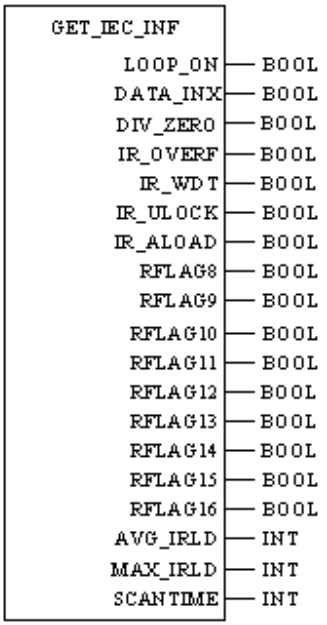
Runtime error

With runtime error -2801, the EFB function is not available if the firmware does not support this service.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
LOOP_ON	BOOL	With "1": The control loop *(see Concept, Context Help,)* ends, logic is partially not executed.
DATA_INX	BOOL	With "1": Range exceeded *(see Concept 2.6, User Manual,)*, invalid access of structured data.
DIV_ZERO	BOOL	With "1": Division by zero *(see Concept, Context Help,)* in inline code (option "fastest code").
IR_OVERF	BOOL	With "1": Overflow in one or more interrupt sections.
IR_WDT	BOOL	With "1": The 20 ms Watchdog Timer has run out for one or more interrupt sections.

Parameters	Data type	Meaning
IR_ULOCK	BOOL	With "1": One or more disables still exist at the end of the cycle time. (The enable did not take place.) **Note:** Does not function when using the simulator.
IR_ALOAD	BOOL	With "1": The maximum cycle time for interrupt sections exceed the limit of 50 % of the total cycle time. The output MAX_IRLD > 50. **Note:** Does not function when using the simulator.
RFLAG8	BOOL	Reserved flag for later use.
RFLAG9	BOOL	Reserved flag for later use.
RFLAG10	BOOL	Reserved flag for later use.
RFLAG11	BOOL	Reserved flag for later use.
RFLAG12	BOOL	Reserved flag for later use.
RFLAG13	BOOL	Reserved flag for later use.
RFLAG14	BOOL	Reserved flag for later use.
RFLAG15	BOOL	Reserved flag for later use.
RFLAG16	BOOL	Reserved flag for later use.
AVG_IRLD	INT	Percent value of the average cycle time of interrupt sections, measured over the total cycle time [0...100]. Resetting takes place by carrying out a complete load or PLC start. **Note:** Does not function when using the simulator.
MAX_IRLD	INT	Percent value of the maximum average cycle time of interrupt sections, measured over the total cycle time [0...100]. If this exceeds 50 %, the IR_ALOAD flag is set. Resetting takes place by carrying out a complete load or PLC start. **Note:** Does not function when using the simulator.
SCAN_TIME	INT	Value of the current cycle time in milliseconds, calculated like the PLC (average of the last 8 cycles).

GET_TOD: Reading the hardware clock (Time Of Day)

5

Introduction

This section describes function block GET_TOD.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block searches (together with the other function blocks in the HSBY group) the configuration of the respective PLC for the necessary components. These components apply to actual connected hardware.

Therefore the correct functioning of this function block on the simulators cannot be guaranteed.

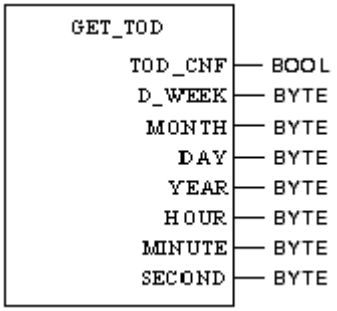
The function block GET_TOD reads the hardware system clock, if relevant registers are provided with this configuration. If these registers are not present, the output TOD_CNF is set to "0".

Additional parameters EN and ENO can be defined.

Description

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
TOD_CNF	BOOL	"1" = 4x-register for hardware system clock was found and the clock is operational. "0" = time is set at the moment. In this case the other outputs keep their values.
D_WEEK	BYTE	Weekday, 1 = Sunday .. 7 = Saturday
MONTH	BYTE	Month 1..12
DAY	BYTE	Day 1..31
YEAR	BYTE	Year 0..99
HOUR	BYTE	Hour 0..23
MINUTE	BYTE	Minute 0..59
SECOND	BYTE	Second 0..59

HSBY_RD: Reading the Hot Standby command register

6

Introduction

This section describes function block HSBY_RD.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block allows you to use the IEC Hot Standby functionality. It searches (together with the other function blocks in the HSBY group) the configuration of the respective PLC for the necessary components. These components apply to actual connected hardware.

Therefore the correct functioning of this function block on the simulators cannot be guaranteed.

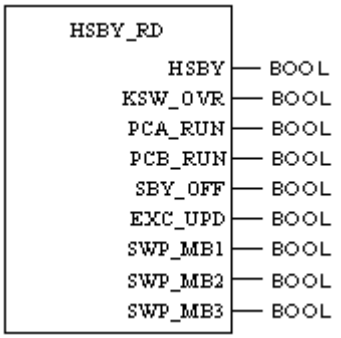
The function block HSBY_RD independently checks if a Hot Standby configuration exists. In that case the contents of the command register will be communicated, and the HSBY output is set to "1". If there is no Hot Standby configuration, the HSBY output is set to "0".

Additional parameters EN and ENO can be defined.

Display

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
HSBY	BOOL	"1" = Hot Standby configuration found
KSW_OVR	BOOL	Keyswitch override "1" = ● Position `Off Line` on the rotary switch of the Hot Standby module (CHSxxx) is deactivated using the software. ● Position `Xfer` on the rotary switch of the Hot Standby module (CHSxxx) is activated via the software. ● Position `Run` on the rotary switch of the Hot Standby-module (CHSxxx) is deactivated using the software. ● The Hot Standby system is in running mode (Run-LED an). "0" = All positions on the rotary switch of the Hot Standby module (CHSxxx) are activated via the software.

Parameter	Data type	Meaning
PCA_RUN	BOOL	PLC A running "1" = The PLC with the Hot Standby module in switch position A on the local rack, is in running mode (Run-LED of PLC and Standby-/Primary-LED of the Hot Standby module on). This is of importance only if the keyswitch override is activated.
PCB_RUN	BOOL	PLC B running "1" = The PLC with the Hot Standby module in switch position B on the local rack is in the running mode (Run-LED of PLC and Standby-/Primary-LED of the Hot Standby module on). This is of importance only if the keyswitch override is activated.
SBY_OFF	BOOL	Standby Off on logic mismatch "1" = Logic difference is permitted (although both PLCs contain different programs, the Hot Standby-SPS stays in Online mode.) "0" = Logic difference is not permitted (The standby PLC switches to offline mode as soon as both PLCs contain different programs.)
EXC_UPD	BOOL	Exec Update "1" = Exec Update in the Standby-PLC is possible with the primary PLC still running. (After Exec Update the standby PLC changes back to the online mode.)
SWP_MB1	BOOL	Swap address Modbus Port 1 "1" = Swap address Modbus Ports 1 activated
SWP_MB2	BOOL	Swap address Modbus Port 2 "1" = Swap address Modbus Ports 2 activated
SWP_MB3	BOOL	Swap address Modbus Port 3 "1" = Swap address Modbus Ports 3 activated

HSBY_ST: Reading the Hot Standby status register

7

Introduction

This section describes function block HSBY_ST.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block allows you to use the IEC Hot Standby functionality. It searches (together with the other function blocks in the HSBY group) the configuration of the respective PLC for the necessary components. These components apply to actual connected hardware.

Therefore the correct functioning of this function block on the simulators cannot be guaranteed.

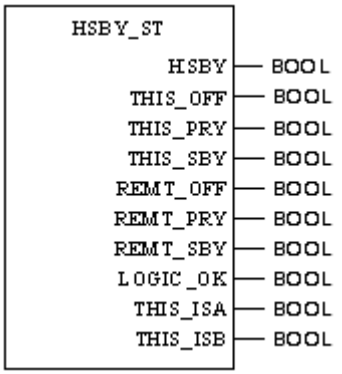
This function block is used to read the IEC Hot Standby status registers. If there is no Hot Standby configuration or if the existing Hot Standby configuration does not have a "non transfer" area containing the status register, the HSBY output is set to "0".

Additional parameters EN and ENO can be defined.

Description

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
HSBY	BOOL	"1" = Hot Standby configuration found, and a "non-transfer"-area was entered to it.
THIS_OFF	BOOL	"1" = This PLC is offline
THIS_PRY	BOOL	"1" = This PLC is the primary PLC
THIS_SBY	BOOL	"1" = This PLC is the standby PLC
REMT_OFF	BOOL	"1" = The other (remote) PLC is offline
REMT_PRY	BOOL	"1" = The other PLC is the primary PLC
REMT_SBY	BOOL	"1" = The other PLC is the standby PLC
LOGIC_OK	BOOL	"1" = Both PLC programs are identical
THIS_ISA	BOOL	"1" = This PLC has the Hot Standby module with switch position A in the local rack.
THIS_ISB	BOOL	"1" = This PLC has the Hot Standby module with switch position B in the local rack.

HSBY_WR: Writing the Hot Standby command register

8

Introduction

This section describes function block HSBY_WR.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block allows you to use the IEC Hot Standby functionality. It searches (together with the other function blocks in the HSBY group) the configuration of the respective PLC for the necessary components. These components apply to actual connected hardware.

Therefore the correct functioning of this function block on the simulators cannot be guaranteed.

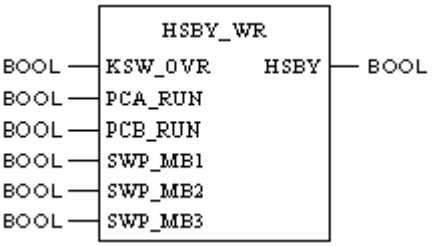
The function block HSBY_WR is used to set various Hot Standby modes allowed for IEC Hot Standby. Setting the respective modes means a change in the Hot Standby command register, which is carried out automatically by the function block. If there is no Hot Standby configuration, the HSBY output is set to "0", otherwise it is set to "1".

Additional parameters EN and ENO can be defined.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
KSW_OVR	BOOL	Keyswitch override "1" = Switch at Hot Standby module (CHSxxx) will be disabled.
PCA_RUN	BOOL	PLC A running "1 -> 0" = And Keyswitch override (KSW_OVR) causes the PLC with the Hot Standby module with switch position A in its local rack to be forced into offline mode.
PCB_RUN	BOOL	PLC B running "1 -> 0" = And Keyswitch override (KSW_OVR) causes the PLC with the Hot Standby module with switch position B in its local rack to be forced into offline mode.
SWP_MB1	BOOL	Swap address Modbus Port 1 "0 -> 1" = The Modbus address at Port 1 of the NEW primary PLC changes if a switchover has occurred. (new primary PLC address = old address + 128 new standby PLC address = old address -128)
SWP_MB2	BOOL	Swap address Modbus Port 2 "0 -> 1" = The Modbus address at Port 2 of the NEW primary PLC changes if a switchover has occurred. (new primary PLC address = old address + 128 new standby PLC address = old address -128).

Parameter	Data type	Meaning
SWP_MB3	BOOL	Swap address Modbus Port 3 "0 -> 1" = The Modbus address at Port 3 of the NEW primary PLC changes if a switchover has occurred. (new primary PLC address = old address + 128 new standby PLC address = old address -128).
HSBY	BOOL	"1" = Hot Standby configuration found.

ISECT_ON: Unlocking a specific interrupt section

9

Introduction

This chapter describes the ISECT_ON block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block can unlock a specific time event section, after it has previously been locked using the ISECT_OFF *(see page 51)* block.

An unlocked section is initiated as soon as the respective hardware signal (I/O event section) or time interval (time event section) is triggered. This also increments the event and activations counter. A possible interrupt causes an interruption during the processing of the section, it will be continued afterwards.

The input pin SECT_CTRL returns the control variable of a specific section. This variable contains the section name.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:

Parameter description

Function block parameter description:

Parameter	Data type	Meaning
SECTCTRL	SECT_CTRL	Control variable, contains the section names.

ISECT_OFF: Locking a specific interrupt section

10

Introduction

This chapter describes the ISECT_OFF block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block can lock a specific time event section or I/O event section, and can be unlocked using the ISECT_ON *(see page 47)* block.

Locking means only the section cannot be processed. The event counter counts the incoming hardware signal and time interval of the locked section. The activation counter only counts processed or unlocked sections. A possible interrupt on an interrupt section has no effect.

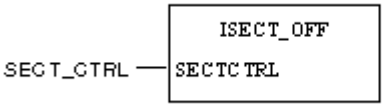
The input pin SECT_CTRL returns the control variable of a specific section. This variable contains the section name.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Function block parameter description:

Parameter	Data type	Meaning
SECTCTRL	SECT_CTRL	Control variable, contains the section names.

ISECT_STAT: Interrupt Section Status

11

Introduction

This chapter describes the ISECT_STAT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block reads the internal states of an interrupt section and copies this data to the data structures that the respective outputs are assigned to.

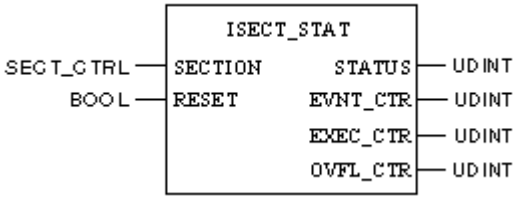
NOTE: You can also see the status table information using the menu command **Online →Event sections**.

The RESET input pin resets all outputs to 0.

Representation

Symbol

Block representation:



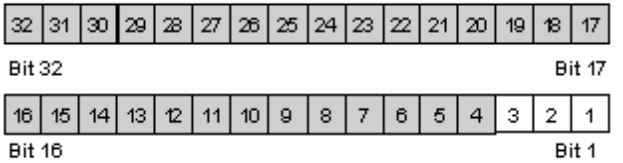
Parameter description

Function block parameter description:

Parameter	Data type	Meaning
SECTION	SECT_CTRL	Control variable = Section name, whose status should be requested.
RESET	BOOL	Resets the outputs to 0.
STATUS	UDINT	Contains the interrupt section status (see section "Interrupt Section Status")
EVNT_CTR	UDINT	Contains the number of all events.
EXEC_CTR	UDINT	Contains the number of all executed events.
OVFL_CTR	UDINT	Contains the number of all events that could not be triggered as they were triggered during the processing of a section.

Interrupt Section Status

Bit allocation:



Bit	Allocation
1	Overflow has occurred, an event could not be processed.
2	Watchdog Timer is timed out.
3	The locked Interrupt sections were all unlocked, but the lock counter was not 0.
4-8	reserved
9-16	Internal use
17-32	Reserved

I_UNLOCK: Unlocking all interrupt sections

12

Introduction

This chapter describes the I_UNLOCK block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

General information

The blocks I_LOCK and I_UNLOCK are used to encapsulate logic that may not be interrupted during the execution of an Event section. This is the case if commonly used variables are accessed for example. The *I_MOVE: Interrupt protected move, page 75* block is used for copy operations that cannot be interrupted.

I_LOCK and I_UNLOCK can be used in cyclical sections as well as in Event sections.

Function description

This function block can unlock all Timer Event sections or I/O Event sections at the same time, after they have previously been locked using the I_LOCK *(see page 63)* block.

Locking means only the immediate processing of Event sections is locked. If the locking (e.g. by calling the I_UNLOCK block) is unlocked, only the Events accumulated after that date are processed, the respective Event sections are then executed. A maximum of 1 event per Event section (therefore a maximum of 16 Timer Events and 64 I/O Event) may occur while the lock is active. Further incoming events increment the overflow counter of the respective Event section, but does not cause any further execution of the respective Event section.

If it is guaranteed that the lock is not active for longer than the minimum time interval between two events of an Event section, then no event is lost due to the locking function. However, the respective Event section execution is delayed. To prevent permanent locking of Event sections, the output pin LOCKCTR is set to 0 at the end of an Event section (I_LOCK call within the Event section) or after each cycle (I_LOCK call within a cyclic section), i.e. the lock is automatically unlocked.

The output pin LOCKCTR returns the current value of the general lock counter. This counter value is reduced every time the I_UNLOCK block is called. The counter value 0 means that the sections are unlocked and their logic is executed. If the value of the output pin LOCKCTR is no 0 at the end of an Event section (I_UNLOCK no present in the section), Bit 3 of the Event-Section-Status is additionally set.

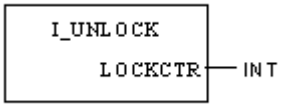
The activation counter only counts processed or unlocked sections.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Function block parameter description:

Parameter	Data type	Meaning
LOCKCTR	INT	Current value of the general lock counter. The value is incremented every time the I_LOCK block is called and decremented every time the I_UNLOCK block is called. The value 0 means that the sections are unlocked.

I_LOCK: Locking all interrupt sections

13

Introduction

This chapter describes the I_LOCK block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

General information

The blocks I_LOCK and I_UNLOCK are used to encapsulate logic that may not be interrupted during the execution of an Event section. This is the case if commonly used variables are accessed for example. The *I_MOVE: Interrupt protected move, page 75* block is used for copy operations that cannot be interrupted.

I_LOCK and I_UNLOCK can be used in cyclical sections as well as in Event sections.

Function description

This function block can lock all Timer Event sections or I/O Event sections at the same time, and can be unlocked using the I_UNLOCK *(see page 59)* block.

Locking means only the immediate processing of Event sections is locked. If the locking (e.g. by calling the I_UNLOCK block) is unlocked, only the Events accumulated after that date are processed, the respective Event sections are then executed. A maximum of 1 event per Event section (therefore a maximum of 16 Timer Events and 64 I/O Event) may occur while the lock is active. Further incoming events increment the overflow counter of the respective Event section, but does not cause any further execution of the respective Event section.

If it is guaranteed that the lock is not active for longer than the minimum time interval between two events of an Event section, then no event is lost due to the locking function. However, the respective Event section execution is delayed. To prevent permanent locking of Event sections, the output pin LOCKCTR is set to 0 at the end of an Event section (I_LOCK call within the Event section) or after each cycle (I_LOCK call within a cyclic section), i.e. the lock is automatically unlocked.

The output pin LOCKCTR returns the current value of the general lock counter. This counter value is incremented every time the I_LOCK block is called and decremented every time the I_UNLOCK block is called. The counter value 0 means that the sections are unlocked and their logic is executed. If the value of the output pin LOCKCTR is no 0 at the end of an Event section (I_UNLOCK no present in the section), Bit 3 of the Event-Section-Status is additionally set.

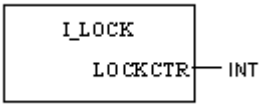
The activation counter only counts processed or unlocked sections.

Additional parameters EN and ENO can be defined.

Representation

Symbol

Block representation:



Parameter description

Function block parameter description:

Parameter	Data type	Meaning
LOCKCTR	INT	Current value of the general lock counter. The value is incremented every time the I_LOCK block is called and decremented every time the I_UNLOCK block is called. The value 0 means that the sections are unlocked.

LOOPBACK: Re-entry

14

Introduction

This section describes function block LOOPBACK.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

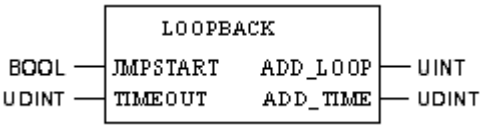
This function block triggers a jump to the start of the application program (restart of application program).

Additional parameters EN and ENO can be defined.

Description

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
JMPSTART	BOOL	1 = Jump is executed
TIMEOUT	UDINT	Watchdog time in microseconds
ADD_LOOP	UINT	Number of additional loop cycles
ADD_TIME	UDINT	Time for additional cycles in microseconds

Detailed description

Triggering the jump

As long as the input JMPSTART is set to "0" (FALSE), the function block triggers no function. If the input JMPSTART is set to "1" (TRUE), the jump is executed at the start of the application program, as long as the time stated at the input TIMEOUT has **not** expired.

Adapt watchdog time

The jump is only executed if an appropriate watchdog time is set at input TIMEOUT. Appropriate means that the watchdog time must be longer than the actual execution time of the application program.

NOTE: Please note that the watchdog time is measured in units of **microseconds**(10 000 are equal to 10 milliseconds). If the value at the input TIMEOUT is "0", no jump is executed.

Loop cycle display

The output ADD_LOOP shows the additional loop cycles executed by the application program.

Display of additional cycle time

The output ADD_TIME shows the time in microseconds needed for the additional cycles. This output can show unexpected values if the TIMEOUT pre-settings are small. Therefore this value should only be taken as general information (e.g. for diagnostics). It should not be used for further calculations.

Summary

Jumps to the start of the application program are only executed if:

- Value "1" is set at the input JMPSTART.
- An appropriate watchdog time (microseconds) is set at the input TIMEOUT (watchdog time > execution time of application program).
- The defined watchdog time at the TIMEOUT input has **not** yet expired.

M1HEALTH: Module health status for M1

15

Introduction

This section describes function block M1HEALTH.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block provides the health status for I/O modules, which are operated together with the PLC M1/Momentum.

16 I/O modules are allocated to an output "STATUSx". Each module is characterised by a bit of the corresponding output "STATUSx". The bit allocation is defined through the wiring of the I/O modules. The furthest bit on the left in "STATUSx" corresponds to the I/O module which is closest to the PLC (in relation to each of the 16 I/O modules).

The local module connected to the PLC is characterised by the output ATIDROP.

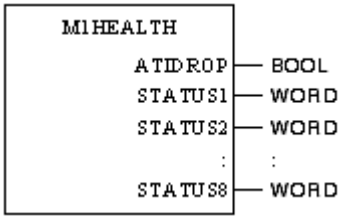
NOTE: If a module has been configured and works correctly, the corresponding bit is set to "1".

Additional parameters EN and ENO can be defined.

Description

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
ATIDROP	BOOL	Status bit of the local station (ATI=Adaptable I/O Interface)
STATUS1	WORD	Status bits of the I/O modules 1 ... 16
STATUS2	WORD	Status bits of the I/O modules 17 ... 32
STATUS3	WORD	Status bits of the I/O modules 33 ... 48
STATUS4	WORD	Status bits of the I/O modules 49 ... 64
STATUS5	WORD	Status bits of the I/O modules 65 ... 80
STATUS6	WORD	Status bits of the I/O modules 81 ... 96
STATUS7	WORD	Status bits of the I/O modules 97 ... 112
STATUS8	WORD	Status bits of the I/O modules 113 ... 128

I_MOVE: Interrupt protected move

16

Introduction

This chapter describes the I_MOVE block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function assigns the input value to the output and is therefore interrupt protected.

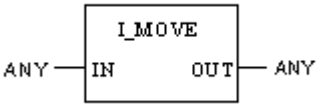
This block is used if the variable at the block is used simultaneously in cyclically processed sections and interrupt sections (time event and I/O event section). The value assignment is therefore protected from an interruption by a time event or I/O event section.

The MOVE block constructed in the same way, however, the value assignment is not interrupt protected.

Representation

Symbol

Block representation:



Formulas

OUT = IN

Parameter description

Description of the block parameter:

Parameter	Data type	Meaning
IN	ANY	Input value
OUT	ANY	Output value

ONLEVT: Online event

17

Introduction

This section describes function block ONLEVT.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function description

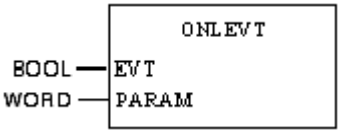
With this function block unexpected program conditions can be entered into the fallback buffer for the online event display. For this the fallback recognition "E_EFB_ONLEVT" is used. Additionally, the parameter is transferred at the input PARAM. EVT "1" results in an entry into the fallback buffer.

Additional parameters EN and ENO can be defined.

Representation

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
EVT	BOOL	"1": Entry into the online event display fallback buffer.
PARAM	WORD	Parameter transferred to the online event display fallback buffer.

PLCSTAT: PLC health status

18

Introduction

This section describes function block PLCSTAT.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block reads the Quantum PLC internal statuses and error bits and copies this data to the data structures allocated to the respective outputs.

NOTE: This function block was basically designed for the Quantum product family only. However, it can also be used, within certain limitations, for the product families Compact, Momentum and Atrium.

NOTE: Status chart information can also be viewed via the menu command **Online →Controller status**.

Additional parameters EN and ENO can be defined.

Only data with the input bit (PLC_READ, RIO_READ, DIO_READ) set to "1" will be read.

Evaluation for Quantum

The evaluation of PLC_STAT (PLC status), RIO_STAT (I/O status) and DIO_STAT (I/O communications status) is possible for the Quantum PLC type.

NOTE: The name of the output DIO_STAT is confusing. This output only relates to the remote I/O Drop Status Information (S908) and not to the Distributed I/O status. In order to read-out the Distributed I/O status use the function block DIOSTAT *(see page 19)*.

Evaluation for Compact

The evaluation of PLC_STAT (PLC status), RIO_STAT (I/O status) and DIO_STAT (I/O communications status) is possible for the Compact PLC type.

Evaluation for Momentum

The evaluation of PLC_STAT (PLC status) and RIO_STAT (I/O bus status) is possible for the Momentum PLC type.

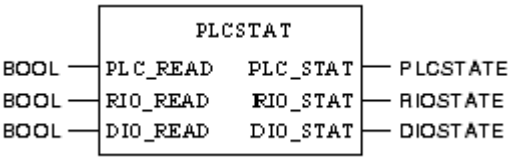
Evaluation for Atrium

Only the evaluation of PLC_STAT (PLC status) is possible for the Atrium PLC type.

Representation

Symbol

Function block description:



Parameter description PLCSTAT

Function block parameter description PLCSTAT:

Parameters	Data type	Meaning
PLC_READ	BOOL	1 = copies the PLC status from the status chart to the output PLC_STAT.
RIO_READ	BOOL	1 = copies the RIO status from the status chart to the output RIO_STAT.
DIO_READ	BOOL	1 = copies the DIO status from the status chart to the output DIO_STAT.
PLC_STAT	PLCSTATE	Contains the PLC status.
RIO_STAT	RIOSTATE	Contains the RIO status (I/O status) for Quantum/B800-hardware or contains the I/O status for Compact or contains the I/O status for Momentum
DIO_STAT	DIOSTATE	Contains the DIO status (I/O communication status) for Quantum or contains the global I/O status and the repetition status for Compact **Note:** The name of this output is confusing. This output only relates to the remote I/O Drop Status Information (S908) and not to the Distributed I/O status. To read out the distributed I/O status use the function block DIOSTAT *(see page 19)*.

Element description PLCSTATE

Element description PLCSTATE:

Element	Data type	Meaning
word1	WORD	CPU status
word2	WORD	Hot-Standby status (Quantum only)

Element	Data type	Meaning
word3	WORD	PLC status
word4	WORD	RIO status (Quantum, Momentum only)
word5	WORD	PLC stop status
word6	WORD	Number of ladder logic segments (as decimal number)
word7	WORD	End of Logic (EOL) pointer address
word8	WORD	RIO redundancy and timeout (Quantum, Momentum only)
word9	WORD	ASCII message status (Quantum only)
word10	WORD	RUN/LOAD/DEBUG status
word11	WORD	Reserve

Description of RIOSTATE element for Quantum

Description of RIOSTATE elements for Quantum:

Element	Data type	Meaning
word1	WORD	I/O drop 1, rack 1
word2	WORD	I/O drop 1, rack 2
...	...	...
word5	WORD	I/O drop 1, rack 5
word6	WORD	I/O drop 1, rack 2
word7	WORD	I/O drop 2, rack 2
...	...	...
word171	WORD	I/O drop 32, rack 5

Description of RIOSTATE element for Compact

Description of RIOSTATE elements for Compact:

Element	Data type	Meaning
word1	WORD	I/O status, rack 1
word2	WORD	I/O status, rack 2
word3	WORD	I/O status, rack 3
word4	WORD	I/O status, rack 4
word5	WORD	not used
...	...	...
word160	WORD	not used

Description of RIOSTATE element for Momentum

Description of RIOSTATE elements for Momentum:

Element	Data type	Meaning
word1	WORD	Functional ability of local Momentum I/O
word2	WORD	Functional ability of bus I/O
word3	WORD	Functional ability of bus I/O
...	...	...
word9	WORD	Functional ability of bus I/O
word10	WORD	not used
...	...	...
word160	WORD	not used

Description of DIOSTATE element for Quantum

Description of DIOSTATE elements for Quantum:

Element	Data type	Meaning
word172	WORD	Switch on error numbers: This word is always 0 when the system is running. If an error occurs, the PLC does not start but generates a stop code
word173	WORD	Cable A error
word174	WORD	Cable A error
word175	WORD	Cable A error
word176	WORD	Cable B error
word177	WORD	Cable B error
word178	WORD	Cable B error
word179	WORD	Global communication errors
word180	WORD	Global cumulative error counter for cable A
word181	WORD	Global cumulative error counter for cable B
word182	WORD	I/O drop 1 health status and repetition counter (first word)
word183	WORD	I/O drop 1 health status and repetition counter (second word)
word184	WORD	I/O drop 1 health status and repetition counter (third word)
word185	WORD	I/O drop 2 health status and repetition counter (first word)
...	...	...

Element	Data type	Meaning
word275	WORD	I/O drop 32 health status and repetition counter (first word)
word276	WORD	I/O drop 32 health status and repetition counter (second word)
word277	WORD	I/O drop 32 health status and repetition counter (third word)

Description of DIOSTATE element for Compact

Description of DIOSTATE elements for Compact:

Element	Data type	Meaning
word1	WORD	not used
...	...	...
word10	WORD	not used
word11	WORD	Global I/O status
word12	WORD	I/O error counter
word13	WORD	PAB repetition counter
word14	WORD	not used
...	...	...
word106	WORD	not used

PLC status (PLC_STAT) for Quantum, Compact, Momentum and Atrium

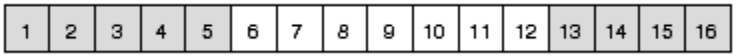
General information

NOTE: Information corresponds to status table words 1 to 11 in the dialog **Controller status**.

The conditions are true when the bits are set to 1.

PLC status (PLCSTATE: word1)

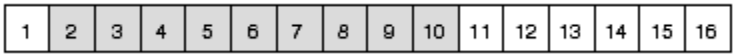
Bit allocation:



Bit	Allocation
6	Enable constant sweep
7	Enable single sweep delay
8	1 = 16 bit user logic 0 = 24 bit user logic
9	Alternating current ON
10	Run light OFF
11	Memory protect OFF
12	Battery failed
13-16	Reserved

Hot Standby status (PLCSTATE: word2) (Quantum only)

Bit allocation:

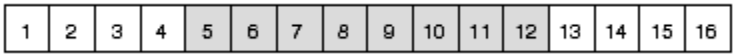


Bit	Allocation
1	CHS 110/S911/R911 present and OK
11	0 = CHS shift switch set to A 1 = CHS shift switch set to B
12	0 = PLCs have equal logic 1 = PLCs have unequal logic

Bit	Allocation
13, 14	Remote system condition Dec binary 1 0 1 = Offline 2 1 0 = Primary 3 1 1 = Standby
15, 16	Local system condition Dec binary 1 0 1 = Offline 2 1 0 = Primary 3 1 1 = Standby

PLC status (PLCSTATE: word3)

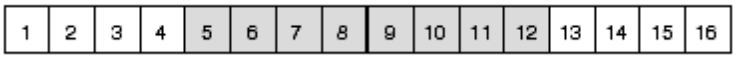
Bit allocation:



Bit	Allocation
1	First cycle
2	Start command not yet executed
3	Constant scan times exceeded
4	Quit Dim Awareness
13-16	Single cycles

RIO status (PLCSTATE: word2) (Quantum)

Bit allocation:



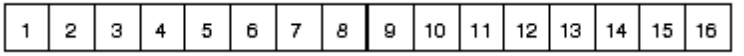
Bit	Allocation
1	IOP defect
2	IOP timeout
3	IOP Loopback
4	IOP memory disturbance
13-16	00 IO has not responded 01 no response 02 Loopback defect

RIO status (PLCSTATE: word4 (Momentum)

With Momentum, this word contains the number (in hex format) of the first faulty module at the bus.

PLC stop status (PLCSTATE: word5)

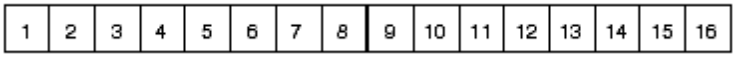
Bit allocation:



Bit	Allocation
1	Peripheral port stop
2	Extended memory parity error (for housing mounted controllers) or traffic COP/Quantum/S908 error (for other controllers). If bit = 1 in a 984B controller, the error was detected in the extended memory, and the controller is running If for another PLC the bit = 1, then either a traffic cop error was detected or the Quantum/S908 is missing from a multi I/O drop configuration.
3	PLC in Dim awareness
4	Illegal peripheral intervention
5	Segment scheduler invalid
6	Node start did not start the segment
7	State RAM test failed
8	Traffic cop invalid
9	Watchdog timer expired
10	Real time clock error
11	CPU logic solve failed (for housing mounted controller) or Coil Use Table (for other controller). If bit = 1 in a housing mounted controller the internal diagnostics detected a CPU failure. If the bit = 1 in another controller, the Coil Use Table does not comply with the coil in the user logic.
12	IOP disturbance
13	Invalid node
14	Logic checksum
15	Coil disabled in RUN
16	Incorrect configuration

PLC stop status (PLCSTATE: word6)

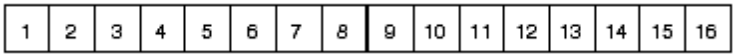
Word 6 displays the number of segments; a binary number is displayed:



Bit	Allocation
1-16	Number of segments (as decimal number)

PLC stop status (PLCSTATE: word7)

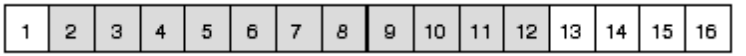
Word 7 displays the End of Logic (EOL) pointer address:



Bit	Allocation
1-16	EOL pointer address

RIO redundancy and timeout (PLCSTATE: word8) (Quantum, Momentum only)

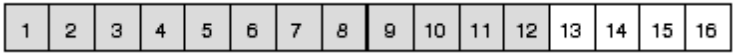
Bit allocation:



Bit	Allocation
1	RIO redundancy cable? 0 = No 1 = Yes
13-16	RIO timeout constant

ASCII message status (PLCSTATE: word9) (Quantum only)

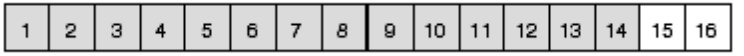
Bit allocation:



Bit	Allocation
13	Number of messages and pointer do not correspond
14	Message pointer invalid
15	Message invalid
16	Message checksum error

RUN/LOAD/DEBUG status (PLCSTATE: word10)

Bit allocation:



Bit	Allocation
15, 16	Local system condition Dec Debug = 0 0 0 Running = 0 1 1 Load = 1 0 2

PLCSTATE: word11

Reserved for extensions

RIO Status (RIO_STAT) for Quantum / B800 Hardware, Part I

General information

NOTE: The information corresponds to status table words 12 to 171 in the PLC status dialog box.

The words show the I/O modules' function status.

Five words are reserved for each I/O station (there can be a maximum of 32 I/O stations). Each one of these words corresponds to one of up to 2 (Quantum) or 5 (B800) possible racks in each I/O station.

Function indicator for Quantum hardware

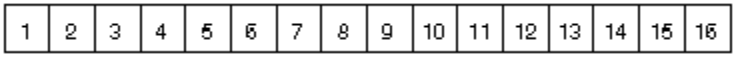
Each rack used for Quantum hardware can contain up to 15 I/O modules (except for the first rack, which can contain a maximum 14 I/O modules). Bits 1– 16 of each word are used as a function indicator for the corresponding I/O module in the racks.

Function indicator for B800 hardware

Each rack used for B800 hardware can contain up to 11 I/O modules. Bits 1– 11 of each word are used as a function indicator for the corresponding I/O module in the racks.

I/O module function status

Bit assignment:



Bit	Assignment
1	Slot 1
2	Slot 2
...	...
16	Slot 16

Conditions for a correct function indicator

Four conditions must be met for an I/O module to be able to provide a correct function indicator:

- The data traffic on the slot has to be monitored.
- The slot must be valid for the equipped module.
- Valid communications must be established between the module and the RIO interface on RIO stations.
- Valid communications must be established between the I/O processor in the PLC and the RIO interface on the RIO station.

Note: Please note that the health bits for analog input modules will be set to 1 after the following scenarios even if it takes an extended period of time (1 to 2 seconds) for the values to become valid.

- Loading the project onto the CPU and starting the CPU (start / stop does not require any additional time)
- Replacing modules during ongoing operation (module hot swapping)
- Switching local racks on/off again

The following are input modules:

- 140 ACI 030 00
- 140 ACI 040 00
- 140 AVI 030 00
- 140 AMM 090 00
- 140 ATI 030 00
- 140 ARI 030 10
- 140 AII 330 00
- 140 AII 330 10
- 140 DDI 364 00

Status words for the MMI user controls

The status of the 32 element button controls and PanelMate units on a RIO network can also be monitored with an I/O function status word. The button controls are located on slot 4 in an I/O rack and can be monitored through bit 4 of the corresponding status word. A PanelMate on RIO is located on slot 1 in rack 1 of the I/O station and can be monitored through bit 1 of the first status word for the I/O station.

NOTE: The ASCII keyboard's communication status can be monitored with the error numbers in the ASCII read/write instructions.

RIO status (DIO_STAT) for Quantum, Part II

General information

NOTE: Information corresponds to status words 172 to 277 in the PLC status dialogue.

The words contain the I/O communication status (DIO status) Words 1 to 10 are global status words. Of the remaining 96 words, three words are allocated to each of the up to 32 I/O drops.

Word 1 saves the Quantum switch on error numbers. This word is always 0 when the system is running. If an error occurs, the PLC does not start but generates an PLC stop status (word 5 of PLC_STAT).

The conditions are true when the bits are set to 1.

Switch on error numbers (DIOSTATE: word1)

The conditions are true when the bits are set to 1.

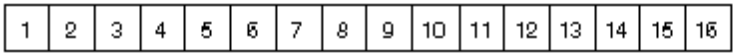
Quantum switch on error numbers:

Code	Error	Meaning (location of error)
01	BADTCLEN	Traffic cop length
02	BADLNKNUM	RIO link number
03	BADNUMDPS	I/O drop number in traffic cop
04	BADTCSUM	Traffic cop checksum
10	BADDDLEN	I/O drop descriptor length
11	BADDRPNUM	I/O drop number
12	BADHUPTIM	I/O drop stop time
13	BADASCNUM	ASCII port number
14	BADNUMODS	Module number in an I/O drop
15	PRECONDRP	I/O drop is already configured
16	PRECONPRT	Port is already configured
17	TOOMNYOUT	More than 1024 output locations
18	TOOMNYINS	More than 1024 input locations
20	BADSLTNUM	Module slot address
21	BADRCKNUM	Rack address
22	BADOUTBC	Number of output bytes
23	BADINBC	Number of input bytes
25	BADRF1MAP	First reference number
26	BADRF2MAP	Second reference number

Code	Error	Meaning (location of error)
27	NOBYTES	No input or output bytes
28	BADDISMAP	EI/O flag bit not at 16 bit limit
30	BADODDOUT	Unmated, odd output module
31	BADODDIN	Unmated, odd input module
32	BADODDREF	Unmated odd module reference
33	BAD3X1XRF	1x-reference after 3x-register
34	BADDMYMOD	Dummy module reference already in use
35	NOT3XDMY	3x-module is no dummy module
36	NOT4XDMY	4x-module is no dummy module
40	DMYREAL1X	Dummy module, then real 1x-module
41	REALDMY1X	Real, then 1x-dummy module
42	DMYREAL3X	Dummy module, then real 3x-module
43	REALDMY3X	Real, then 3x-dummy module

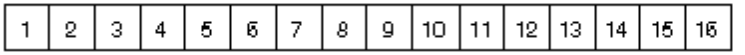
Status of cable A (DIOSTATE: word2, word3, word4)

Bit allocation for word2:



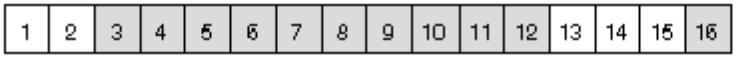
Bit	Allocation
1 - 8	Counts frame fields
9 - 16	Counts DMA receiver overflows

Bit allocation for word3:



Bit	Allocation
1 - 8	Counts receiver errors
9 - 16	Counts I/O drop receiver failures

Bit allocation for word4:

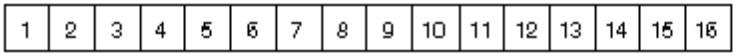


Bit	Allocation
1	1 = frame too short
2	1 = no frame end

Bit	Allocation
13	1 = CRC error
14	1 = alignment error
15	1 = overflow error

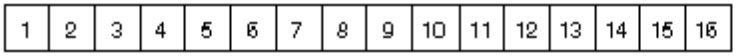
Status of cable B (DIOSTATE: word5, word6, word7)

Bit allocation for word5:



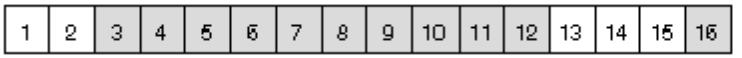
Bit	Allocation
1 - 8	Counts frame fields
9 - 16	Counts DMA receiver overflows

Bit allocation for word6:



Bit	Allocation
1 - 8	Counts receiver errors
9 - 16	Counts I/O drop receiver failures

Bit allocation for word7:

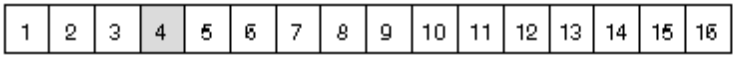


Bit	Allocation
1	1 = frame too short
2	1 = no frame end
13	1 = CRC error
14	1 = alignment error
15	1 = overflow error

Global communication status (DIOSTATE: word8)

The conditions are true when the bits are set to 1.

Bit allocation for word8:

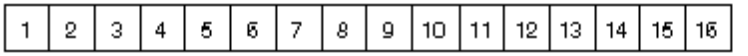


Bit	Allocation
1	Comm. health display
2	Cable A status
3	Cable B status
5 - 8	Communication counter lost
9 - 16	Cumulative repetition counter

Global cumulative error counter for cable A (DIOSTATE: word9)

The conditions are true when the bits are set to 1.

Bit allocation for word9:

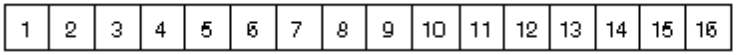


Bit	Allocation
1 - 8	Counts recognised errors
9 - 16	Counts zero responses

Global cumulative error counter for cable B (DIOSTATE: word10)

The conditions are true when the bits are set to 1.

Bit allocation for word10:

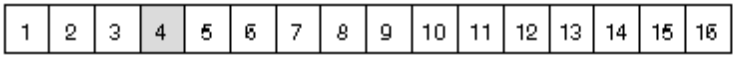


Bit	Allocation
1 - 8	Counts recognised errors
9 - 16	Counts zero responses

RIO status (DIOSTATE: word11 to word106)

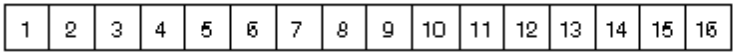
Words 11 to 106 are used to describe the RIO station status, three status words are planned for each I/O drop.

The **first** word in each group of three shows the communication status for the corresponding I/O drop:



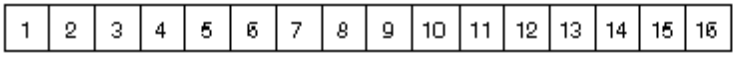
Bit	Allocation
1	Communication health
2	Cable A status
3	Cable B status
5 - 8	Counter for lost communications
9 - 16	Cumulative repetition counter

The **second** word in each group of three is the cumulative I/O drop error counter at cable A for the corresponding I/O drop:



Bit	Allocation
1 - 8	Minimum one error in words 2 to 4
9 - 16	Counts zero responses

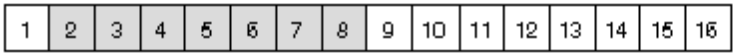
The **third** word in each group of three is the cumulative I/O drop error counter at cable B for the corresponding I/O drop:



Bit	Allocation
1 - 8	Minimum one error in words 5 to 7
9 - 16	Counts zero responses

NOTE: For a PLC where the I/O drop 1 is reserved for the local I/O, words 11 to 13 are allocated as follows:

word11 shows the global I/O drop status:



Bit	Allocation
1	All modules OK
9 - 16	Counts, how often a module is regarded as not OK, counter overflow is at 255

word12 is used as a 16 bit I/O bus error counter.

word13 is used as a 16 bit I/O repetition counter.

I/O status (RIO_STAT) for Compact

I/O status (RIO_STAT: word1 – 4)

I/O status for word1 to word4:

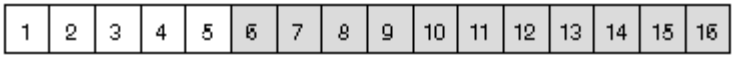
word1	Module rack 1
word2	Module rack 2
word3	Module rack 3
word3	Module rack 4

The words show the I/O module health status in the max. 4 racks.

Each word contains the health status of up to five A120 I/O modules. The bit with the highest value (left) represents the module health status in slot 1 of the rack.

If a module is entered into the I/O module and activated, the corresponding bit is set to value "1". If a module is not entered into the I/O module or not activated, the corresponding bit is set to value "0".

Bit allocation:



Bit	Allocation
1	Slot 1
2	Slot 2
3	Slot 3
4	Slot 4
4	Slot 5

NOTE: Slots 1 and 2 in the module rack 1 (word 1) are not used because the CPU itself uses both slots.

I/O status (RIO_STAT: word5 -160)

not used

I/O bus status (RIO_STAT) for Momentum

General information

NOTE: Information corresponds to status words 12 to 20 of the PLC status dialogue.

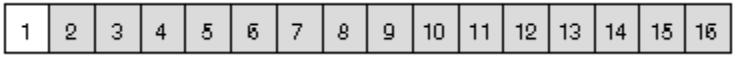
The words show the I/O module health status.

The first word represents the health of the local Momentum module. The following 8 words represent the health of the up to 128 bus modules.

The conditions are true when the bits are set to 1.

RIO status (RIOSTATE: word1)

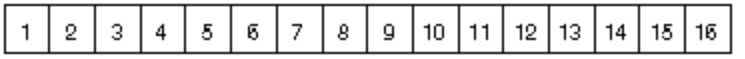
Bit allocation:



Bit	Allocation
1	Local module operative

RIO status (RIOSTATE: word2 -9)

Bit allocation:



Bit	Allocation
1	Module 1
2	Module 2
...	...
16	Module 16

Health of bus module:

word2	Shows the health of bus modules 1 - 16
word3	Shows the health of bus modules 17 -32
word4	Shows the health of bus modules 33 -48
word5	Shows the health of bus modules 49 -64
word6	Shows the health of bus modules 65 -80
word7	Shows the health of bus modules 81 -96
word8	Shows the health of bus modules 97 -112
word9	Shows the health of bus modules 113 -128

I/O status (RIO_STAT: word10 -160)

not used

Global I/O status and the repetition status (DIO_STAT) for Compact

General information

NOTE: Information corresponds to status words 172 to 277 in the PLC status dialogue.

The words "word11" to "word13" contain health status and communication information on the I/O modules installed. The words "word1" to "word10" and "word14" to "word106" are not used.

The conditions are true when the bits are set to 1.

DIOSTATE: word1 - 10 and word14 - 106

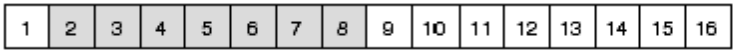
not used

Global I/O status (DIOSTATE: word11)

Bit 1 is set when all modules are able to function.

Bits 9 to 16 work as a counter to show how often an I/O module failed. Counter overrun is at 255.

Bit allocation for word11:

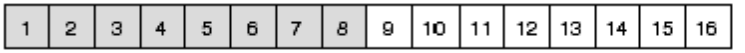


Bit	Allocation
1	All modules OK
9 - 16	Counts, how often a module is regarded as not OK.

I/O error counter (DIOSTATE: word12)

Bits 9 to 16 work as a counter to show during how many cycles an I/O module failed. Counter overrun is at 255.

Bit allocation for word12:



Bit	Allocation
9 - 16	Counts, how often a module is regarded as not OK.

PAB repetition counter (DIOSTATE: word13)

This word shows the communication status of the PAB (Parallel System Bus). Normally this word shows the value "0". An error is displayed, when the bus error is still detected after 5 repetitions. In this case the PLC is stopped and the error code "10" is displayed. Errors can be caused e.g. by a short within the rack or by noise.

PRJ_VERS: Project Name/Version

19

Introduction

This chapter describes the PRJ_VERS block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

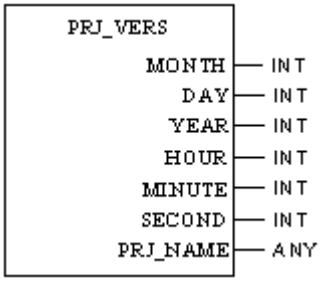
The block gives both the project names as well as the project versions on its output pins.

The project version consists of a time/date stamp, the project name contains a maximum character length of 8 characters/bytes.

Representation

Symbol

Block representation:



Parameter description

Description of the Block Parameter:

Parameter	Data type	Meaning
MONTH	INT	Month: 1-12 (January - December)
DAY	INT	Day: 1-31
YEAR	INT	Year (only two decimal places available, e.g. 2001 = 01)
HOUR	INT	Hour: 0-23
MINUTE	INT	Minute: 0-59
SECOND	INT	Seconds: 0-59
PRJ_NAME	ANY	Project name with a maximum of 8 characters/bytes **Note:** The project name depends on the character/byte size of the variable entered. This means that if a variable less than 8 bytes long is entered, the project name can only appear to be this length as well.

RES_IEC_INF: Resetting the IEC Status Flags

20

At a glance

This chapter describes the function block RES_IEC_INF.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

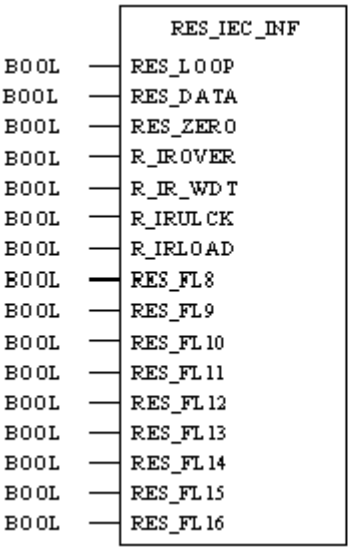
With this function block, you can reset the set IEC system error flags individually (see function block *GET_IEC_INF: Read the IEC Status Flags, page 27*).

Additional parameters EN and ENO can be defined.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
RES_LOOP	BOOL	With "1": Flag LOOP_ON *(see page 29)* is reset.
RES_DATA	BOOL	With "1": Flag DATA_INX *(see page 29)* is reset.
RES_ZERO	BOOL	With "1": Flag DIV_ZERO *(see page 29)* is reset.
R_IROVER	BOOL	With "1": Flag IR_OVERF *(see page 29)* is reset.
R_IR_WDT	BOOL	With "1": Flag IR_WDT *(see page 29)* is reset.
R_IRULCK	BOOL	With "1": Flag IR_ULOCK *(see page 29)* is reset.
R_IRALOAD	BOOL	With "1": Flag IR_ALOAD *(see page 29)* is reset.
RES_F8	BOOL	With "1": Flag RFLAG8 *(see page 29)* is reset. (Reserved flag for later use.)
RES_F9	BOOL	With "1": Flag RFLAG9 *(see page 29)* is reset. (Reserved flag for later use.)
RES_F10	BOOL	With "1": Flag RFLAG10 *(see page 29)* is reset. (Reserved flag for later use.)

Parameters	Data type	Meaning
RES_F11	BOOL	With "1": Flag RFLAG11 *(see page 29)* is reset. (Reserved flag for later use.)
RES_F12	BOOL	With "1": Flag RFLAG12 *(see page 29)* is reset. (Reserved flag for later use.)
RES_F13	BOOL	With "1": Flag RFLAG13 *(see page 29)* is reset. (Reserved flag for later use.)
RES_F14	BOOL	With "1": Flag RFLAG14 *(see page 29)* is reset. (Reserved flag for later use.)
RES_F15	BOOL	With "1": Flag RFLAG15 *(see page 29)* is reset. (Reserved flag for later use.)
RES_F16	BOOL	With "1": Flag RFLAG16 *(see page 29)* is reset. (Reserved flag for later use.)

REV_XFER: Writing and reading the two reverse transfer register

21

Introduction

This section describes function block REV_XFER.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block allows you to use the IEC Hot Standby functionality. It searches (together with the other function blocks in the HSBY group) the configuration of the respective PLC for the necessary components. These components always apply to actually connected hardware.

Therefore the correct functioning of this function block on the simulators cannot be guaranteed.

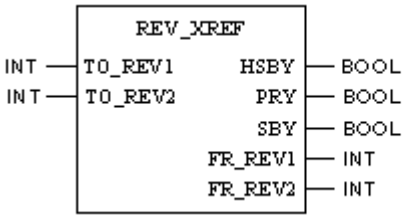
The function block REV_XFER provides the option of transmitting two 16 bit words (4x register) from the standby PLC to the primary PLC. This is possible only with an existing hot standby configuration including the non-transfer area. The two registers transmitted through this function block are the first two 4x registers in the non-transfer area (reverse transfer registers).

Additional parameters EN and ENO can be defined.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
TO_REV1	INT	Describes the first reverse transfer register if this PLC is the standby PLC.
TO_REV2	INT	Describes the second reverse transfer register if this PLC is the standby PLC.
HSBY	BOOL	1 = Hot Standby configuration found and a 'non-transfer' area is entered in it.
PRY	BOOL	1 = This PLC is the primary PLC.
SBY	BOOL	1 = This PLC is the standby PLC.
FR_REV1	INT	Content of first reverse transfer register from standby PLC (Output only if HSBY is "1").
FR_REV2	INT	Content of second reverse transfer register from standby PLC (Output only if HSBY is "1").

RIOSTAT: Module health status (RIO)

22

Introduction

This section describes function block RIOSTAT.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block provides the health status for I/O modules of an I/O drop (local/remote I/O).

Quantum I/O or 800 I/O can be used.

An output "STATx" is allocated to each rack. Each module (slot) of this rack is characterised by a bit of the corresponding "STATx" output. The bit on the far left-hand side in "STATx" corresponds to the slot on the far left-hand side of the rack x.

Use of "STAT1" to "STAT5":

- **Quantum I/O**
  There is only one rack for an I/O drop, e.g. only "STAT1" is used.
- **800 I/O**
  There can be up to 5 racks for an I/O drop, e.g. "STAT1" corresponds to module rack 1, "STAT5" corresponds to module rack 5.

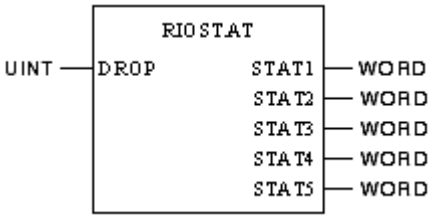
NOTE: If a module of the rack has been configured and works correctly, the corresponding bit is set to "1".

Additional parameters EN and ENO can be defined.

Representation

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
DROP	UINT	Local/remote I/O drop no. (1...32)
STAT1	WORD	Module rack 1 status bit pattern
STAT2	WORD	Module rack 2 status bit pattern (800 I/O only)
...	...	...
STAT5	WORD	Module rack 5 status bit pattern (800 I/O only)

SAMPLETM: Sample time

23

Introduction

This section describes function block SAMPLETM.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

With this function block the function blocks of the control mechanism are released under time control.

To control, the output Q of the function block SAMPLETM is connected with the input ENof the function block to be controlled.

The output Q is activated for one program cycle after the stated time at the input INTERVAL has expired.

The input DELSCAN was created to prevent the start of more than one sample time dependent FFBs which are controlled by various SAMPLETM function blocks. At this input, the number of cycles, by which the activation of Q after a cold start is delayed, is given. Therefore it is possible to release sample time dependent function blocks step by step to reduce the load on the CPU during the start cycle.

Additional parameters EN and ENO can be defined.

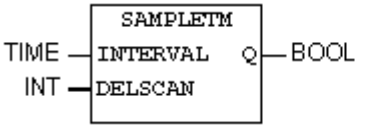
NOTE: If this module is not called up at least 1x during 2 INTERVAL-times, an entry is made in the event viewer.

This can occur when
- the logic of the module is not processed, or
- EN during > 2 INTERVAL-times = 0.

Representation

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
INTERVAL	TIME	Sample time for connected control mechanism function block
DELSCAN	INT	Number of delay cycles after a cold start
Q	BOOL	Release of control mechanism function block

SET_TOD: Setting the hardware clock (Time Of Day)

24

Introduction

This section describes function block SET_TOD.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function searches (together with the other function blocks in the HSBY group) the configuration of the respective PLC for the necessary components. These components always apply to hardware that is actually connected.

Therefore the correct behavior of this function block on the simulators cannot be guaranteed.

The function block sets the hardware system clock, if the corresponding registers are provided with this configuration. If these registers are not present, the output TOD_CNF is set to "0".

The function block reads the input values when a "1" signal occurs on input S_PULSE and transfers them to the hardware clock.

NOTE: As S_PULSE is a static input, the write operation is active when S_PULSE = 1. This means that after the write operation has been executed, S_PULSE must be reset to 0 in order to ensure that the hardware clock functions correctly.

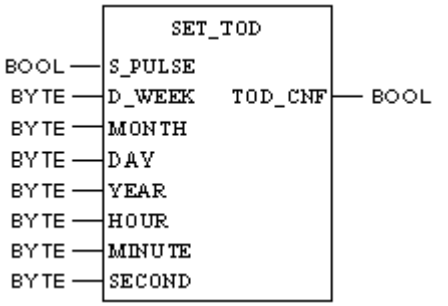
For all input values:
- If the value exceeds the specified maximum value, the maximum is used.
- If the value falls below the specified minimum value, the minimum is used.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
S_PULSE	BOOL	"1" = the input values are accepted and written into the clock.
D_WEEK	BYTE	Day of week, 1 = Sunday 7 = Saturday
MONTH	BYTE	Month 1..12
DAY	BYTE	Day 1..31
YEAR	BYTE	Year 0..99
HOUR	BYTE	Hour 0..23
MINUTE	BYTE	Minute 0..59
SECOND	BYTE	Second 0..59
TOD_CNF	BOOL	"1" = 4x-register for hardware system clock has been found and the clock is ready for operation. "0" = Time is currently being set or hardware clock was not found.

SFCCNTRL: SFC controller

25

Introduction

This section describes function block SFCCNTRL.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is used to control sequence strings.

This function block is used to control the processing of a SFC section. You can skip steps, for example, or turn on/off the editing function of the transition conditions, or reset the string to the initial state.

The function block provides the use of all the control options that are provided by the commands of the online menu and the animation panel. Additionally the function block provides the option to disable the operating mode changes from the online menu/animation panel.

⚠ DANGER

Danger of unsafe, dangerous and destructive tool and process operations.

RESETSFC, DISTRANS, DISACT, STEPUN and STEPDEP should not be used for error detection on controllers for machine tool, process or material handling systems when they are running. This can lead to unsafe, dangerous and destructive tool and process operations.

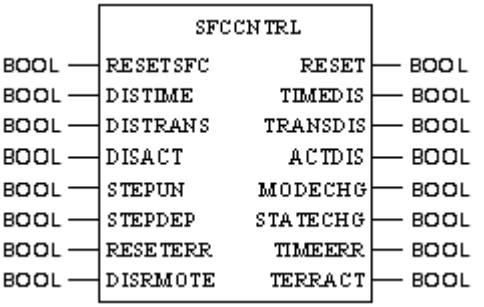
Failure to follow these instructions will result in death or serious injury.

Additional parameters EN and ENO can be defined.

Representation

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
RESETSFC	BOOL	0 -> 1: reset string; 1 -> 0: standardised start of string (set initial step)
DISTIME	BOOL	1: Turn off time controlling (This will not influence animation or output TERRACT.)
DISTRANS	BOOL	1: Turn off evaluation of transitions
DISACT	BOOL	1: Turn off editing of actions and reset all actions of the string
STEPUN	BOOL	0 -> 1: Activate next step, regardless of transition
STEPDEP	BOOL	0 -> 1: Activate next step, when transition condition is fulfilled
RESETERR	BOOL	0 -> 1: Turns off display of all minimum time control errors when animating the SFC section. Time control errors already displayed will be updated. If there are no existing time control errors output TERRACT will be reset.
DISRMOTE	BOOL	1: Prevent controlling of SFC with the help of editing parameters of the online animation controller
RESET	BOOL	1: String is reset.
TIMEDIS	BOOL	1: Time control is disabled
TRANSDIS	BOOL	1: Evaluation of transitions is disabled

Parameters	Data type	Meaning
ACTDIS	BOOL	1: Editing of actions is disabled and all actions of the string are reset
MODECHG	BOOL	1: String operating mode has changed
STATECHG	BOOL	1: String status has changed
TIMEERR	BOOL	1: Error in time control detected (will be displayed for one cycle only)
TERRACT	BOOL	1: Error in time control detected (will be displayed until error is inactive)

Function description

Controlling of a sequence string

Each function block SFCCNTRL corresponds to one SFCsection.

There are 4 possibilities to control a string:
- with the menu commands in the online menu
- with the animation controller (in the online menu)
- with the function block SFCCNTRL
- with the function block XSFCCNTRL

If a sequence string is controlled with different control options at the same time, they are of equal status.

It is possible to lock the editing parameters for the SFC that run with commands of the on-line menu or the animation controller using the function block SFCCNTRL.

NOTE: To allocate the function block to a corresponding SFC section, **the** name of the SFC section should be given as instance name of the function block SFCCNTRL.

Correct function block editing can only be done by placing the function block into a section, which will be processed earlier than the SFC section that needs to be controlled. This is done using the menu command **Project →Execution order...**

Organization of inputs/outputs

Inputs and outputs of the function block are divided into 5 groups:
- Settings of operating modes
  - RESETSFC
  - DISTIME
  - DISTRANS
  - DISACT
- Control commands
  - STEPUN
  - STEPDEP
  - RESETERR
- Locking the SFC online commands
  - DISRMOTE
- Display of operating mode settings
  - RESET
  - TIMEDIS
  - TRANSDIS
  - ACTDIS *(see page 139)*

- General displays
  - MODECHG
  - STATECHG
  - TIMEERR
  - TERRACT

Parameter description

General information

⚠ WARNING

RESETSFC, DISTRANS, DISACT, STEPUN and STEPDEP should not be used for error detection on controllers for machine tool, process or material handling systems when they are running. This can lead to unsafe, dangerous and destructive tool and process operations connected to the controller.

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

RESETSFC

This input enables you to reset the string and perform a standardised start.

- Reset the string
  A 0 -> 1 edge at the input will stop the string and reset all the actions. It is not possible to operate.
- Standardized start of the string
  A 1 -> 0 edge at the input will reset the string, i.e. the initial step will be active.

DISTIME (DISable TIME check)

Signal 1 at the input will disable the time control of the steps. This will not influence animation or output TERRACT.

DISTRANS (DISableTRANSitions)

Signal 1 at the input will disable the evaluation of transition states. The string will remain in the current state, regardless of the signals at the transitions. The string can only be controlled with the commands (RESETSFC, STEPUN, STEPDEP).

DISACT (DISable ACTions)

Signal 1 at the input will disable the editing of the step actions.

STEPUN (STEP UNconditional)

A 0 -> 1 edge at the input activates the next step, regardless of the transition state, but only when the step delay time of the active step is completed.

In simultaneous branching this command always activates every branching; in alternative branching it always activates the branching on the left.

The command STEPDEP is used to activate branching within the process.

STEPDEP (STEP transition DEPendent)

A 0 -> 1 edge at the input will activate the next step if the transition condition is fulfilled.

The use of this control command makes sense only with signal 1 at the input DISTRANS.

The control command freezing the transitions (DISTRANS = 1) enables the user to edit manually step by step the string elements. The transition will be processed in accordance with the transition condition.

RESETERR (RESET ERRor display)

A 0 -> 1 edge at the input will turn off the display of all minimum time control errors in the SFC section animation. Time control errors already displayed will be updated. If there are no existing time control errors output TERRACT will be reset.

DISRMOTE (DISable ReMOTE)

Signal 1 at the input disables controlling the SFC using editing parameters of the online animation controller (set/reset flag, time check lock, transition lock, action lock). The SFC can still be controlled with the function block SFCCNTRL.

RESET (mode of RESET)

The output is set to 1 if the string is stopped with the reset command, regardless of the reset being triggered via the function block (input RESETSFC) or via the SFC on-line commands. Therefore it can happen that the output has a different status from the input RESETSFC.

TIMEDIS (execution mode TIME supervision DISabled)

The output is set to 1 if the time error display is switched off, regardless of the display being switched off via the function block (input DISTIME) or via the SFC on-line commands. Therefore it can happen that the output has a different status from the input DISTIME.

TRANSDIS (execution mode TRANSitions DISabled)

The output is set to 1 if the transition evaluation is stopped, regardless of the display being switched off via the function block (input DISTRANS) or via the SFC online commands. Therefore it can happen that the output has a different status from the input DISTRANS.

ACTDIS (execution mode ACTions DISabled)

The output is set to 1 if the action output is stopped, regardless of the output being switched off via the function block (inputDISACT) or via the SFC online commands. Therefore it can happen that the output has a different status from the input DISACT.

MODECHG (execution MODECHanGe)

The output is set to 1 for a cycle if one or more operation modes of the string are modified, regardless of the modification being made via the function block (input RESESTSFC, DISTIME, DISACT or DISTRANS) or via the SFC online commands.

STATECHG (sfc STATE CHanGe)

The output is set to 1 for a cycle if the state of the string is modified, regardless of the modification being caused by the sequence of the string, via the function block or via the SFC online commands.

TIMEERR (supervision TIME ERROR)

The output is set to 1 for one cycle if one or more time controlling errors occurred.

TERRACT (supervision Time ERRor ACTive)

The output remains at 1 as long as one or more time controlling errors occur.

SKP_RST_SCT_FALSE: Skip rest of section

26

Introduction

This chapter describes function block SKP_RST_SCT_FALSE.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

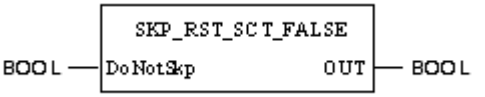
This function block triggers a skip over the logic that follows the function block (dependent on the FFB-execution sequence) in the current section. A "0" signal (FALSE) at the DoNotSkp input will trigger the skip.

Additional parameters EN and ENO can be defined.

Representation

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Parameters	Meaning
DoNotSkp	BOOL	0 = Jump is executed
OUT	BOOL	0 = Jump was executed 1 = Jump was not executed

SYSCLOCK: System clock

27

Introduction

This section describes function block SYSCLOCK.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

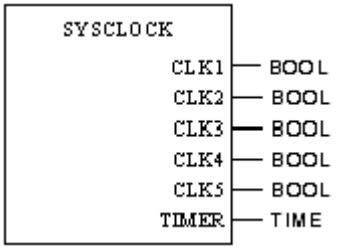
This function block generates pulses in the frequencies 0.3125 Hz, 0.6250 Hz, 1.2500 Hz, 2.5000 Hz and 5.0000 Hz. Additionally, the accumulated time since system start up is displayed.

Additional parameters EN and ENO can be defined.

Representation

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
CLK1	BOOL	Pulse frequency 0.3125 Hz (Pulse 3.2 s)
CLK2	BOOL	Pulse frequency 0.6250 Hz (Pulse 1.6 s)
CLK3	BOOL	Pulse frequency 1.2500 Hz (Pulse 800 ms)
CLK4	BOOL	Pulse frequency 2.5000 Hz (Pulse 400 ms)
CLK5	BOOL	Pulse frequency 5.0000 Hz (Pulse 200 ms)
TIMER	TIME	Accumulated time since system start up (in ms)

SYSSTATE: System state

28

Introduction

This section describes function block SYSSTATE.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function description

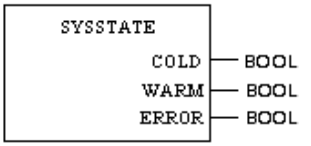
This function block displays the output system state information.

Additional parameters EN and ENO can be defined.

Representation

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
COLD	BOOL	"1": System is in cold start cycle (Cold start means the first start after the project is loaded completely (**Online** →**Load**).
WARM	BOOL	"1": System is in warm start cycle (Warm start means any other start; for example after switching on the power supply, for example, or when starting the PLC after a stop.)
ERROR	BOOL	"1": Fault messages in the error buffer have not been read yet.

NOTE: In cold start cycle the outputs COLD and WARM are set to "1".

XSFCCNTRL: Extended SFC controller

29

Introduction

This section describes function block XSFCCNTRL.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is used to control sequence strings.

This function block provides 2 more services than function block SFCCNTRL.

- It provides the option (ALLTRANS input) to edit all the transition sections of the respective SFC section for the function block (even when the respective step is not active).
- It provides the option of an expanded transition diagnostics. To evaluate this transition diagnostics you will need a special transition diagnostics software.

This function block is used to control the processing of a SFC section. You can skip steps, for example, or turn on/off the editing function of the transition conditions, or reset the string to the initial state.

The function block provides the use of all the control options that are provided by the commands of the online menu and the animation panel. Additionally the function block provides the option to disable the operating mode changes from the online menu/animation panel.

⚠ WARNING

Danger of unsafe, dangerous and destructive tool and process operations c

RESETSFC, DISTRANS, DISACT, STEPUN and STEPDEP should not be used for error detection on controllers for machine tool, process or material handling systems when they are running. This can lead to unsafe, dangerous and destructive tool and process operations c

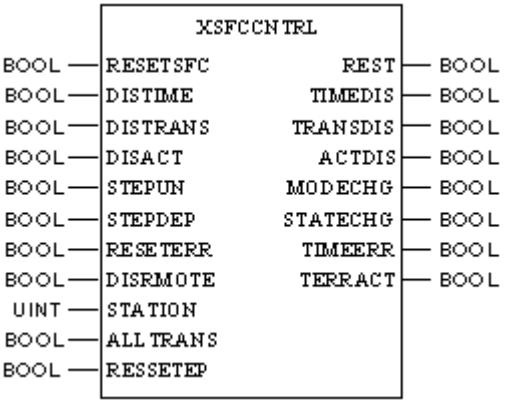
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Additional parameters EN and ENO can be defined.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
RESETSFC	BOOL	0 -> 1: Reset string; 1 -> 0: Standardised start of string (set initial step)
DISTIME	BOOL	1: Turn off time monitoring (This will not influence animation or output TERRACT.)
DISTRANS	BOOL	1: Turn off evaluation of transitions
DISACT	BOOL	1: Turn off editing of actions and reset all actions of the string
STEPUN	BOOL	0 -> 1: Activate next step, regardless of transition
STEPDEP	BOOL	0 -> 1: Activate next step, when transition condition is fulfilled
RESETERR	BOOL	0 -> 1: Turns off display of all minimum time monitoring errors during animation of the SFC section. Time monitoring errors already displayed will be updated. If there are no existing time monitoring errors, output TERRACT will be reset.
DISRMOTE	BOOL	1: Prevent controlling of SFC with the help of editing parameters of the online animation controller

Parameter	Data type	Meaning
STATION	UINT	Drop number (if no entry is made, drop number "0" will be used).
ALLTRANS	BOOL	1: All transition sections of the respective SFC section for the function block will be processed.
RESSTEPT	BOOL	1: Time registration is deactivated. All step times, time monitoring errors and output TERRACT will be reset, as long as the signal is displayed. 0: Time registration is active.
RESET	BOOL	1: String is reset.
TIMEDIS	BOOL	1: Time monitoring is disabled
TRANSDIS	BOOL	1: Evaluation of transitions is disabled
ACTDIS	BOOL	1: Editing of actions is disabled and all actions of the string are reset
MODECHG	BOOL	1: String operating mode has changed
STATECHG	BOOL	1: String status has changed
TIMEERR	BOOL	1: Error in time monitoring detected (will be displayed for one cycle only)
TERRACT	BOOL	1: Error in time monitoring of a transition detected (will be displayed until error is inactive)

Function description

Controlling of a sequence string

Each function block XSFCCNTRL corresponds to one SFC section.

There are 4 possibilities to control a string:
- with the menu commands in the online menu
- with the animation controller (in the online menu)
- with the function block SFCCNTRL
- with the function block XSFCCNTRL

If a sequence string is controlled with different control options at the same time, they are of equal status.

It is possible to lock the editing parameters for the SFC that run with commands of the on-line menu or the animation controller using the function block SFCCNTRL.

NOTE: To assign the function block to a corresponding SFC section the name of the SFC section should be given the instance name of the function block XSFCCNTRL.

Correct function block editing can only be done by placing the function block into a section, which will be processed earlier than the SFC section that needs to be controlled. This is done using the menu command **Project →Execution order...**

Organization of inputs/outputs

Inputs and outputs of the function block are divided into 5 groups:
- Settings of operating modes
  - RESETSFC
  - DISTIME
  - DISTRANS
  - DISACT
- Control commands
  - STEPUN
  - STEPDEP
  - RESETERR
  - STATION
  - ALLTRANS
  - RESSTPEPT
- Locking the SFC online commands
  - DISRMOTE
- Display of operating mode settings
  - RESET
  - TIMEDIS
  - TRANSDIS
  - ACTDIS

- General displays
  - MODECHG
  - STATECHG
  - TIMEERR
  - TERRACT

Parameter description

General information

⚠ WARNING

Risk of unsafe, dangerous and destructive tool and process operations.

RESETSFC, DISTRANS, DISACT, STEPUN and STEPDEP should not be used for error detection on controllers for machine tool, process or material handling systems when they are running. This can lead to unsafe, dangerous and destructive tool and process operations connected to the controller.

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

RESETSFC

This input enables you to reset the string and perform a standardised start.

- Reset the string
  A 0 -> 1 edge at the input will stop the string and reset all the actions. It is not possible to operate.
- Standardized start of the string
  A 1 -> 0 edge at the input will reset the string, i.e. the initial step will be active.

DISTIME (DISable TIME check)

Signal 1 at the input will disable the time monitoring of the steps. This will not influence animation or output TERRACT.

DISTRANS (DISableTRANSitions)

Signal 1 at the input will disable the evaluation of transition states. The string will remain in the current state, regardless of the signals at the transitions. The string can only be controlled with the commands (RESETSFC, STEPUN, STEPDEP).

DISACT (DISable ACTions)

Signal 1 at the input will disable the processing of the step actions.

STEPUN (STEP UNconditional)

A 0 -> 1 edge at the input activates the next step, regardless of the transition state, but only when the step delay time of the active step is completed.

In simultaneous branching this command always activates every branching; in alternative branching it always activates the branching on the left.

The command STEPDEP is used to activate branching within the process.

STEPDEP (STEP transition DEPendent)

A 0 -> 1 edge at the input will activate the next step if the transition condition is fulfilled.

The use of this control command makes sense only with signal 1 at the input DISTRANS.

The control command freezing the transitions (DISTRANS = 1) enables the user to edit manually and step by step the string elements. The transition will be processed in accordance with the transition condition.

RESETERR (RESET ERRor display)

A 0 -> 1 edge at the input will turn off the display of all minimum time monitoring errors in the SFC section animation. Time monitoring errors already displayed will be updated. If there are no existing time monitoring errors, output TERRACT will be reset.

DISRMOTE (DISable ReMOTE)

Signal 1 at the input blocks controlling the SFC using editing parameters of the online animation controller (set/reset flag, time check lock, transition lock, action lock). The SFC can still be controlled with the function block SFCCNTRL.

STATION (STATION number

Station number for transition diagnostics. In case of no other entry, station number "0" will be used.

ALLTRANS (edit ALL TRANSitions)

Signal 1 at the input means that all the transition sections of the respective SFC section for the function block will be processed (even when the respective step is not active). Only the state of the transitions will be determined. That does not influence the sequence string.

By activating the check box **Animate all conditions of the transition sections** in the dialog **Options** → **Preferences** →**Graphical Editors...** you can activate the animation of those transitions and, this way, the determined state of the transitions will be displayed.

NOTE: The additional editing of transition sections with inactive steps may prolong the cycle time of the program by a significant amount.

RESSTEPT (RESet STEP Time)

Signal 1 disables the time registration. All step times (the accumulated time since the activation of a step), time control errors and output TERRACT will be reset, as long as signal 1 is displayed. All displays for faulty steps will be deactivated.

NOTE: For experts:

1. Signal 1 at the input will cause the SFC processor to cancel all fault messages in the diagnostics buffer.
2. The input does not influence the "Automatic acknowledgement".

RESET (mode of RESET)

The output is set to 1 if the string is stopped with the reset command, regardless of the reset being triggered via the function block (input RESETSFC) or via the SFC on-line commands. Therefore it can happen that the output has a different status from the input RESETSFC.

TIMEDIS (execution mode TIME supervision DISabled)

The output is set to 1 if the time error display is switched off, regardless of the display being switched off via the function block (input DISTIME) or via the SFC on-line commands. Therefore it can happen that the output has a different status from the input DISTIME.

TRANSDIS (execution mode TRANSitions DISabled)

The output is set to 1 if the transition evaluation is stopped, regardless of the display being switched off via the function block (input DISTRANS) or via the SFC online commands. Therefore it can happen that the output has a different status from the input DISTRANS.

ACTDIS (execution mode ACTions DISabled)

The output is set to 1 if the action output is stopped, regardless of the output being switched off via the function block (inputDISACT) or via the SFC online commands. Therefore it can happen that the output has a different status from the input DISACT.

MODECHG (execution MODECHanGe)

The output is set to 1 for a cycle if one or more operation modes of the string are modified, regardless of the modification being made via the function block (input RESETSFC, DISTIME, DISACT or DISTRANS) or via the SFC online commands.

STATECHG (sfc STATE CHanGe)

The output is set to 1 for a cycle if the state of the string is modified, regardless of the modification being caused by the sequence of the string, via the function block or via the SFC online commands.

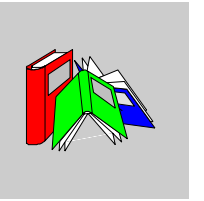
TIMEERR (supervision TIME ERROR)

The output is set to 1 for one cycle if one or more time controlling errors occurred.

TERRACT (supervision Time ERRor ACTive)

The output remains at 1 as long as one or more time monitoring errors occur.

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 on 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 to the data type 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 output module, which were specified in the I/O component list, are automatically assigned to the data type and should therefore only be occupied with Unlocated Variables.

ANY

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

ANY_ELEM

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

ANY_INT

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

ANY_NUM

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

ANY_REAL

In the present version, "ANY_REAL" covers the REAL data type.

Application Window

The window contains 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 (requires 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 hexadecimal system. 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 dual system. 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 in 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.

Bit sequence

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 connection on its left side, unchanged, to the horizontal connection 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 call up 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 it 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.

Defragmenting

With defragmenting, unanticipated gaps (e.g. resulting from deleting unused variables) are removed from memory.

See also **PLC Selection** in the context help.

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 the 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 output side of a derived function block is created in FBD language, LD language, ST language, IL language, but only in the current version of the programming system. Derived functions can also not be defined in the current version.

A distinction is made between local and global DFBs.

DFB Code

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

DFB instance data

The DFB instance data is internal data from the derived function blocks 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, base 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type 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 executable code of all EFBs used. In addition the used EFBs count in DFBs.

Elementary functions/function blocks (EFB)

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

EN / ENO (Enable / Error signal)

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

Error

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

Evaluation

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

Expression

Expressions consist of operators and operands.

F

FFB (Functions/Function blocks)

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

Field variables

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 invocations can be found in the definition "Functional block (instance)". In contrast to the invocations 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 invocations 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 = Number of the section (consecutive numbers)

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 stored 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 locating the EFBs 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 program 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 a 16-bit input word 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 = Number of the section (consecutive numbers)

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 comment must, if present, be the last element of the line.

Instruction (LL984)

When programming electrical controls, the user must 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 "commands" of the ST programming language. Instructions must be completed 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, base 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 decimal system. 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 by which the execution of an operation is initiated.

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 dependent 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 a 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 with direct values. These values can not be overwritten by the program logic (write-protected). A distinction is made between generic and standardized literals.

Literals are also used to allocate, to a constant, a value or a variable, an initial value.

Entries are made as base 2 literal, base 8 literal, base 16 literal, integer literal, real literal or real literal with exponent.

Local derived data types

Local derived data types are only available in a single Concept project and 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 addressed 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 values 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.

Node Address

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

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 executed.

Output parameter (output):

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

Output/Marker bits (0x references)

An 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.

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.

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 permissible 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 characters, 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 = Output/Marker bits

1x range = Input bits

3x range = Input words

4x range = Output registers

6x range = Register in the extended memory

NOTE: The x, which follows each initial reference type number, represents a five-digit 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 with 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 for numbers or timing errors for 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 functioning mode 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, however, it is 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-digit 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-digit 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.

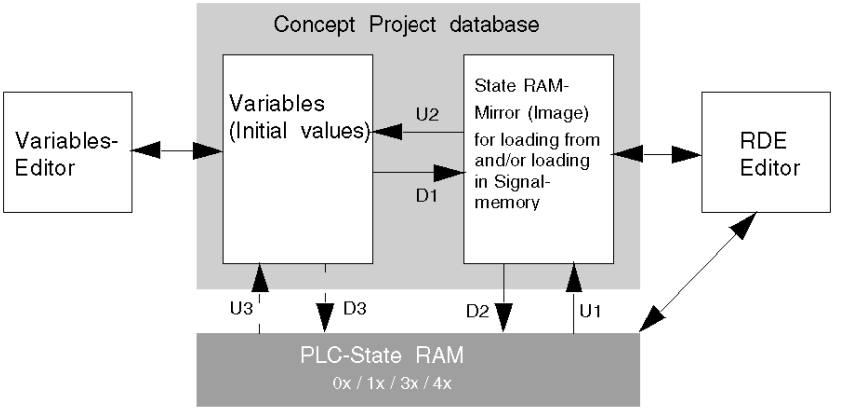
Entering a comma will automatically assign the data type REAL.

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 located in the state RAM.

State RAM overview for uploading and downloading

Overview:



Status Bits

For every device with global inputs or specific inputs/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, in reference to its inputs and outputs, follows 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 = Number of the section (consecutive numbers)

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 preparation of I/O-map SY/MAX-I/O modules for remote controlling 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 labeling 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 data type TIME is 1 ms.

Time literals

Permissible 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 node. The token passes round the nodes in a rotating (increasing) address sequence. All nodes 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, base 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, base 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

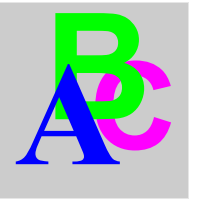
Warning

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



P

R

S

U

W

X