Concept
IEC Block Library
Part: SYSTEM

840 USE 504 00 eng Version 2.6

Table of Contents

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

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

User Comments

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

General Information on the SYSTEM Function Block Library

I

Overview

Introduction

General Information on the SYSTEM Function Block Library

What's in this Part?

This part contains the following chapters:

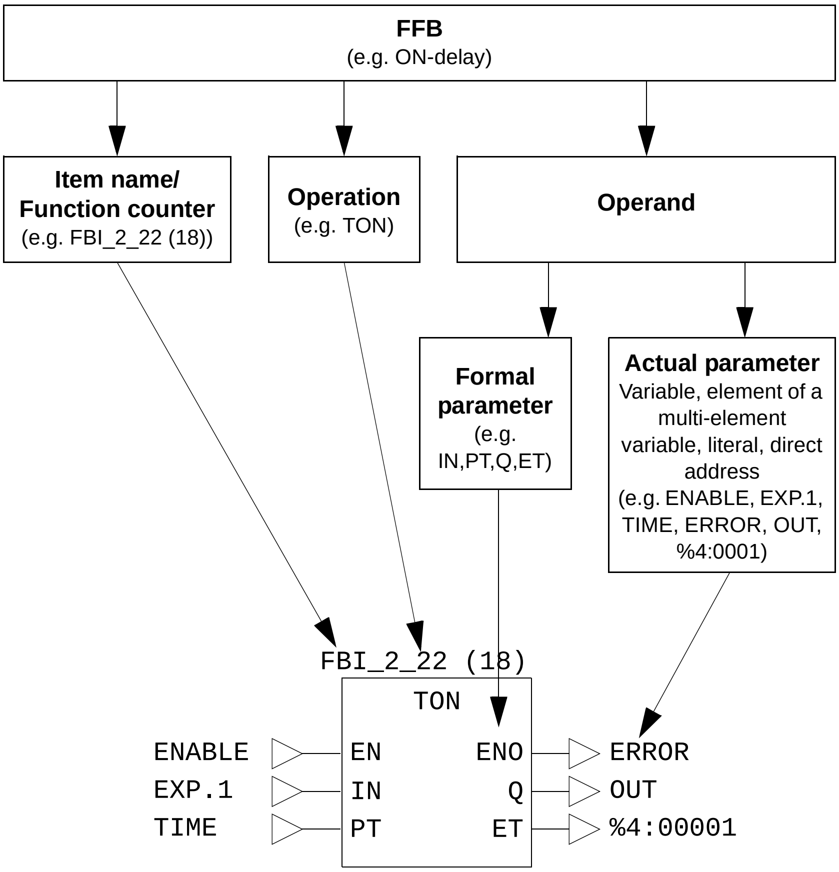
Parameterizing functions and function blocks

1

Parameterizing functions and function blocks

General

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



Operation

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

Operand

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

Formal/actual parameters

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

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

Conditional/ unconditional calls

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

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

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

Calling functions and function blocks in IL and ST

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

EFB Descriptions

II

Overview

Introduction

These EFB descriptions are 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

Overview

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.

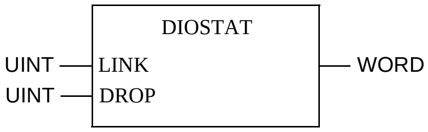
Note: If a module of the I/O drop is configured and works correctly, the corresponding bit is set to "1".

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

Overview

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

Overview

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.

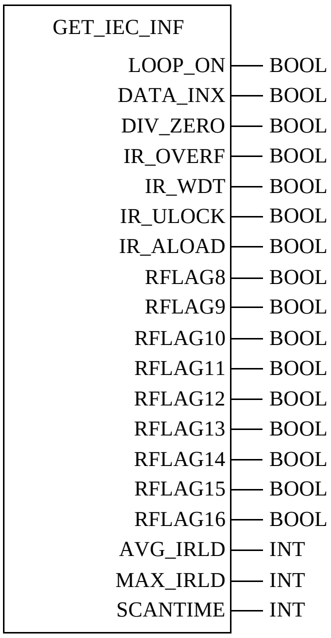
Runtime error

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 ends, logic is partially not executed.
DATA_INX	BOOL	With "1": Range exceeded, invalid access of structured data.
DIV_ZERO	BOOL	With "1": Division by zero 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.
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

Overview

Introduction

This section describes function block GET_TOD.

What's in this Chapter?

This chapter contains the following topics:

Brief description

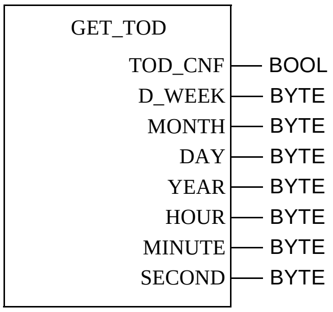
Function description

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

Overview

Introduction

This section describes function block HSBY_RD.

What's in this Chapter?

This chapter contains the following topics:

Brief description

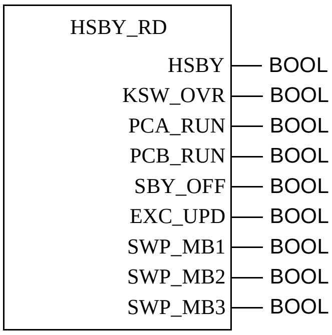
Function description

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.

Description

Symbol

Function block description:



Parameter description

Function block parameter description:

Parameters	Data type	Meaning
HSBY	BOOL	"1" = Hot Standby configuration found
KSW_OVR	BOOL	Keyswitch override "1" = Switch for Hot Standby module (CHSxxx) deactivated via software.
PCA_RUN	BOOL	PLC A running "1" = The PLC with the Hot Standby module with the switch position A in 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.
PCB_RUN	BOOL	PLC B running "1" = The PLC with the Hot Standby module with the switch position B in 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" = The standby PLC switches to the offline mode, if each PLC receives a different program.
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

Overview

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.

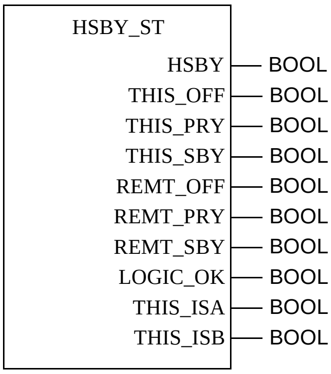
This function block is used to read the IEC Hot Standby status registers. If there is no Hot Standby configuration or if the existing Hot Standby configuration does not have a "non transfer" area containing the status register, the HSBY output is set to "0".

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

Overview

Introduction

This section describes function block HSBY_WR.

What's in this Chapter?

This chapter contains the following topics:

Brief description

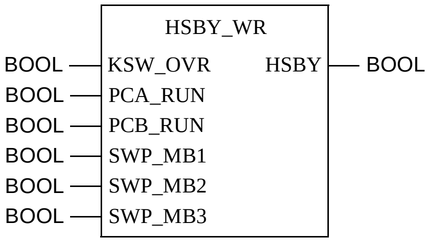
Function description

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).
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

Overview

Introduction

This chapter describes the ISECT_ON block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

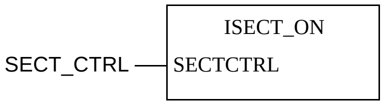
Function description

This function block can unlock a specific time event section, after it has previously been locked using the ISECT_OFF (See *ISECT_OFF: Locking a specific interrupt section, p. 43*) 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

Overview

Introduction

This chapter describes the ISECT_OFF block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

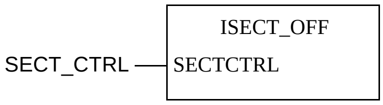
Function description

This function block can lock a specific time event section or I/O event section, and can be unlocked using the ISECT_ON (See *ISECT_ON: Unlocking a specific interrupt section, p. 41*) 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

Overview

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.

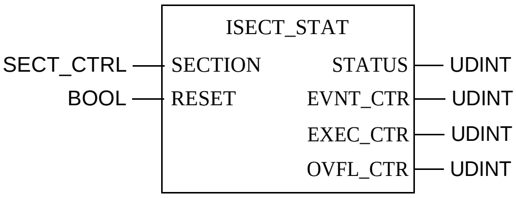
Note: You can also see the status table information using the menu command **Online → Event sections**.

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:

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17

Bit 32 Bit 17

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

Bit 16 Bit 1

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

Overview

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, p. 65* 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.

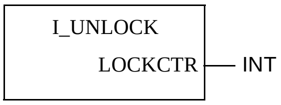
Function description

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 *I_LOCK: Locking all interrupt sections, p. 53*) 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

Overview

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, p. 65* 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.

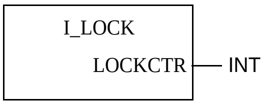
Function description

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 *I_UNLOCK: Unlocking all interrupt sections, p. 49*) 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

Overview

Introduction

This section describes function block LOOPBACK.

What's in this Chapter?

This chapter contains the following topics:

Brief description

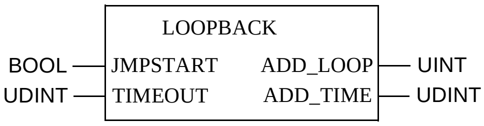
Function description

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

Overview

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.

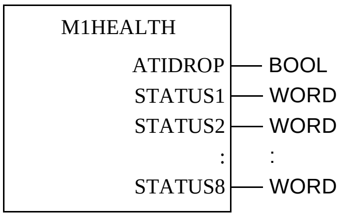
Note: If a module has been configured and works correctly, the corresponding bit is set to "1".

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

Overview

Introduction

This chapter describes the I_MOVE block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

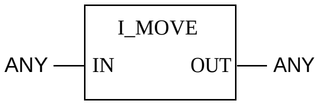
Function description

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

Overview

Introduction

This section describes function block ONLEVT.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

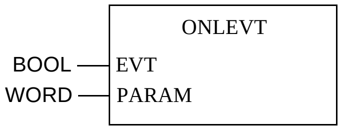
Function description

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

Overview

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 *DIOSTAT: Module health status (DIO), p. 15*).

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.

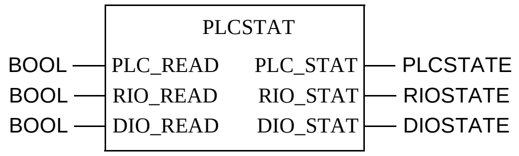
Evaluation for Atrium

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 *DIOSTAT: Module health status (DIO), p. 15*).

Element description PLCSTATE

Element description PLCSTATE:

Element	Data type	Meaning
word1	WORD	CPU status
word2	WORD	Hot-Standby status (Quantum only)
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
...	...	...
word160	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 RIOSTATE element for Quantum

Description of RIOSTATE elements for Quantum:

Element	Data type	Meaning
word1	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
word2	WORD	Cable A error
word3	WORD	Cable A error
word4	WORD	Cable A error
word5	WORD	Cable B error
word6	WORD	Cable B error
word7	WORD	Cable B error
word8	WORD	Global communication errors
word9	WORD	Global cumulative error counter for cable A
word10	WORD	Global cumulative error counter for cable B
word11	WORD	I/O drop 1 health status and repetition counter (first word)
word12	WORD	I/O drop 1 health status and repetition counter (second word)
word13	WORD	I/O drop 1 health status and repetition counter (third word)
word14	WORD	I/O drop 2 health status and repetition counter (first word)
...	...	...
word104	WORD	I/O drop 32 health status and repetition counter (first word)
word105	WORD	I/O drop 32 health status and repetition counter (second word)
word106	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:

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

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:

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

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
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:

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

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:

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

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:

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

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:

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

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:

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

Bit	Allocation
1-16	EOL pointer address

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

Bit allocation:

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

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

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

Bit allocation:

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

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:

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

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

General information

Note: Information corresponds to status words 12 to 171 of the PLC status dialogue.

The words show the I/O module health status.
Five words are reserved for each of the maximum 32 I/O drops. Each word corresponds to one of maximal 2 (Quantum) or 5 (B800) possible racks in each I/O drop.

Health display for Quantum hardware

Each of the racks for Quantum hardware can contain up to 15 I/O modules (except for the first rack which contains a maximum 14 I/O modules). Bit 1... 16 in each word show the corresponding I/O module health display in the racks.

Health display for B800 hardware

Each rack for B800 hardware can contain up to 11 I/O modules. Bit 1... 11 in each word show the corresponding I/O module health display in the racks.

I/O module health status

Bit allocation:

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

Bit	Allocation
1	Slot 1
2	Slot 2
...	...
16	Slot 16

Conditions for a correct health display

Four conditions must be fulfilled for an I/O module to give a correct health display:

- The data traffic of the slot has to be controlled.
- The slot must be valid for the equipped assembly.
- Valid communication must be established between the module and the RIO interface at RIO stations.
- Valid communication must be established between the I/O processor in the PLC and the RIO interface at the RIO station.

Status words for the MMI user controllers

The status of the 32 element button controllers and panelmate units in a RIO network can also be controlled with an I/O health status word. The button controllers are located on slot 4 in a I/O rack and can be controlled at bit 4 of the corresponding status word. A PanelMate on RIO is located on slot 1 in module rack 1 of the I/O drop and can be controlled at bit 1 of the first status word for the I/O drop.

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

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:

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

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:

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

Bit	Allocation
1	Local module operative

RIO status (RIOSTATE: word2 -9)

Bit allocation:

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

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

DIO status (DIO_STAT) for Quantum

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

Code	Error	Meaning (location of error)
23	BADINBC	Number of input bytes
25	BADRF1MAP	First reference number
26	BADRF2MAP	Second reference number
27	NOBYTES	No input or output bytes
28	BADDISMAP	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:

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

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

Bit allocation for word3:

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

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

Bit allocation for word4:

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

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

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

Bit allocation for word5:

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

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

Bit allocation for word6:

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

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

Bit allocation for word7:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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:

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

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.

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:

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

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:

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

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

Overview

Introduction

This chapter describes the PRJ_VERS block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

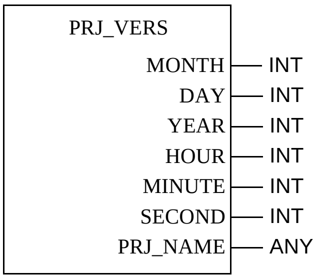
Function description

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

Overview

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

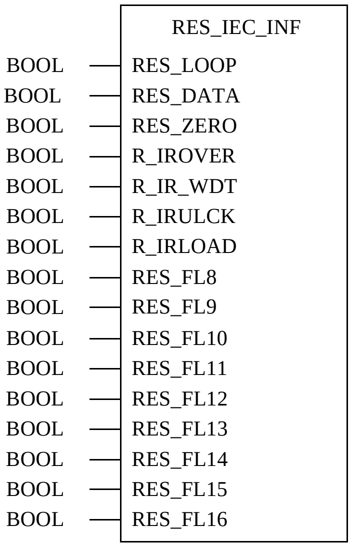
Function description

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, p. 21*).
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 *Parameter description, p. 24*) is reset.
RES_DATA	BOOL	With "1": Flag DATA_INX (See *Parameter description, p. 24*) is reset.
RES_ZERO	BOOL	With "1": Flag DIV_ZERO (See *Parameter description, p. 24*) is reset.
R_IROVER	BOOL	With "1": Flag IR_OVERF (See *Parameter description, p. 24*) is reset.
R_IR_WDT	BOOL	With "1": Flag IR_WDT (See *Parameter description, p. 24*) is reset.
R_IRULCK	BOOL	With "1": Flag IR_ULOCK (See *Parameter description, p. 24*) is reset.
R_IRALOAD	BOOL	With "1": Flag IR_ALOAD (See *Parameter description, p. 24*) is reset.
RES_F8	BOOL	With "1": Flag RFLAG8 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F9	BOOL	With "1": Flag RFLAG9 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F10	BOOL	With "1": Flag RFLAG10 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F11	BOOL	With "1": Flag RFLAG11 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F12	BOOL	With "1": Flag RFLAG12 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F13	BOOL	With "1": Flag RFLAG13 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F14	BOOL	With "1": Flag RFLAG14 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F15	BOOL	With "1": Flag RFLAG15 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)
RES_F16	BOOL	With "1": Flag RFLAG16 (See *Parameter description, p. 24*) is reset. (Reserved flag for later use.)

REV_XFER: Writing and reading the two reverse transfer register

21

Overview

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.

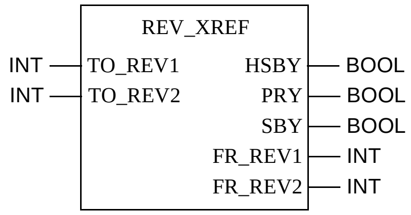
The function block REV_XFER provides the option of transmitting two 16 bit words (4x register) from the standby PLC to the primary PLC. This is possible only with an existing hot standby configuration including the non-transfer area. The two registers transmitted through this function block are the first 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 primary 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. Output only if HSBY is "1".
FR_REV2	INT	Content of second reverse transfer register. Output only if HSBY is "1".

RIOSTAT: Module health status (RIO)

22

Overview

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.

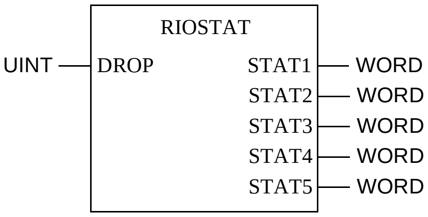
Note: If a module of the rack has been configured and works correctly, the corresponding bit is set to "1".

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

Overview

Introduction

This section describes function block SAMPLETM.

What's in this Chapter?

This chapter contains the following topics:

Brief description

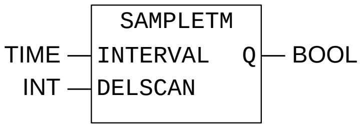
Function description

This function block is used to enable function blocks under time control.
Such controlling requires that output Q of the function block SAMPLETM is connected with input EN of the function block that is to be controlled.
After the length of time specified at input INTERVAL, output Q becomes active for one program cycle at a time.
The input DELSCAN prevents the simultaneous start of several scan-time-dependent FFBs that are selected by different SAMPLETM function blocks. The number of cycles that are recommened to delay an activation of Q after a cold start is specified at this input. This makes it possible to enable the scan-time-dependent function blocks step-by-step, which in turn will reduce the demand on the CPU during a start cycle.
Additional parameters EN and ENO may be configured.

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

Overview

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:

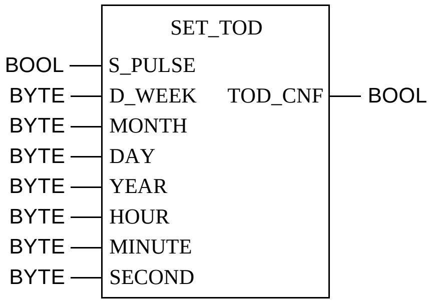
- If the value exceeds the specified maximum value, the maximum is used.
- If the value falls below the specified minimum value, the minimum is used.

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

Overview

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.

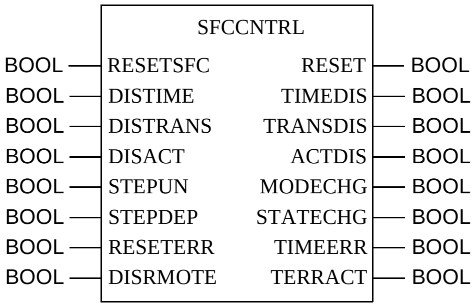
Failure to follow this precaution will result in death, serious injury, or equipment damage.

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
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 *ACTDIS (execution mode ACTions DISabled), p. 116*)
- 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 this precaution 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

Overview

Introduction

This chapter describes function block SKP_RST_SCT_FALSE.

What's in this Chapter?

This chapter contains the following topics:

Brief description

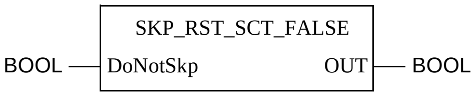
Function description

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

Overview

Introduction

This section describes function block SYSCLOCK.

What's in this Chapter?

This chapter contains the following topics:

Brief description

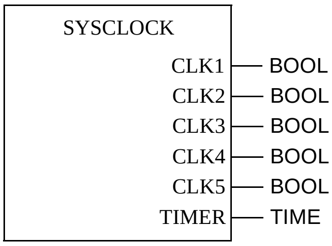
Function description

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

Overview

Introduction

This section describes function block SYSSTATE.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

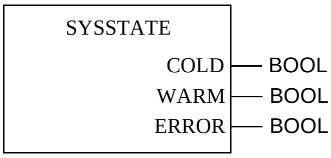
Function description

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

Overview

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

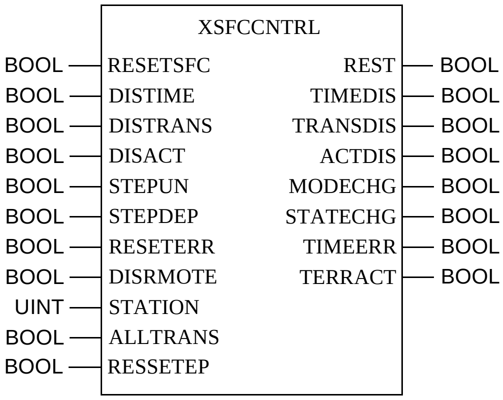
Failure to follow this precaution can result in death, serious injury, or equipment damage.

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
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.

Parameter	Data type	Meaning
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 this precaution 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 one given time. When a window is active, the heading changes color, in order to distinguish it from other windows. Unselected windows are inactive.

Actual parameter
Currently connected Input/Output parameters.

Addresses
(Direct) addresses are memory areas on the PLC. These are found in the State RAM and can be assigned input/output modules.
The display/input of direct addresses is possible in the following formats:

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

ANL_IN
ANL_IN stands for the data type "Analog Input" and is used for processing analog values. The 3x References of the configured analog input module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.

ANL_OUT
ANL_OUT stands for the data type "Analog Output" and is used for processing analog values. The 4x-References of the configured analog output module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.

ANY
In the existing version "ANY" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD and therefore derived data types.

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

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

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

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

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

Application window The window, which contains the working area, the menu bar and the tool bar for the application. The name of the application appears in the heading. An application window can contain several document windows. In Concept the application window corresponds to a Project.

Argument Synonymous with Actual parameters.

ASCII mode American Standard Code for Information Interchange. The ASCII mode is used for communication with various host devices. ASCII works with 7 data bits.

Atrium The PC based controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module occupies a motherboard (requires SA85 driver) with two slots for PC104 daughter boards. From this, a PC104 daughter board is used as a CPU and the others for INTERBUS control.

B

Back up data file (Concept EFB) The back up file is a copy of the last Source files. The name of this back up file is "backup??.c" (it is accepted that there are no more than 100 copies of the source files. The first back up file is called "backup00.c". If changes have been made on the Definition file, which do not create any changes to the interface in the EFB, there is no need to create a back up file by editing the source files (**Objects** → **Source**). If a back up file can be assigned, the name of the source file can be given.

Base 16 literals

Base 16 literals function as the input of whole number values in the hexadecimal system. The base must be denoted by the prefix 16#. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

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

Base 8 literal

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

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

Basis 2 literals

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

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

Binary connections

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

Bit sequence

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

BOOL

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

Bridge

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

BYTE

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

C

Cache The cache is a temporary memory for cut or copied objects. These objects can be inserted into sections. The old content in the cache is overwritten for each new Cut or Copy.

Call up The operation, by which the execution of an operation is initiated.

Coil A coil is a LD element, which transfers (without alteration) the status of the horizontal link on the left side to the horizontal link on the right side. In this way, the status is saved in the associated Variable/ direct address.

Compact format (4:1) The first figure (the Reference) is separated from the following address with a colon (:), where the leading zero are not entered in the address.

Connection A check or flow of data connection between graphic objects (e.g. steps in the SFC editor, Function blocks in the FBD editor) within a section, is graphically shown as a line.

Constants Constants are Unlocated variables, which are assigned a value that cannot be altered from the program logic (write protected).

Contact A contact is a LD element, which transfers a horizontal connection status onto the right side. This status is from the Boolean AND- operation of the horizontal connection status on the left side with the status of the associated Variables/direct Address. A contact does not alter the value of the associated variables/direct address.

D

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

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

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

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

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

Decentral Network (DIO)
A remote programming in Modbus Plus network enables maximum data transfer performance and no specific requests on the links. The programming of a remote net is easy. To set up the net, no additional ladder diagram logic is needed. Via corresponding entries into the Peer Cop processor all data transfer requests are met.

Declaration
Mechanism for determining the definition of a Language element. A declaration normally covers the connection of an Identifier with a language element and the assignment of attributes such as Data types and algorithms.

Definition data file (Concept EFB)
The definition file contains general descriptive information about the selected FFB and its formal parameters.

Derived data type
Derived data types are types of data, which are derived from the Elementary data types and/or other derived data types. The definition of the derived data types appears in the data type editor in Concept.
Distinctions are made between global data types and local data types.

Derived Function Block (DFB)
A derived function block represents the Call up of a derived function block type. Details of the graphic form of call up can be found in the definition " Function block (Item)". Contrary to calling up EFB types, calling up DFB types is denoted by double vertical lines on the left and right side of the rectangular block symbol.
The body of a derived function block type is designed using FBD language, but only in the current version of the programming system. Other IEC languages cannot yet be used for defining DFB types, nor can derived functions be defined in the current version.
Distinctions are made between local and global DFBs.

DINT
DINT stands for the data type "double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The range of values for variables of this data type is from –2 exp (31) to 2 exp (31) –1.

Direct display
A method of displaying variables in the PLC program, from which the assignment of configured memory can be directly and indirectly derived from the physical memory.

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

Dummy
An empty data file, which consists of a text header with general file information, i.e. author, date of creation, EFB identifier etc. The user must complete this dummy file with additional entries.

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

E

Elementary functions/ function blocks (EFB) Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose bodies, for example, cannot be modified with the DFB Editor (Concept-DFB). EFB types are programmed in "C" and mounted via Libraries in precompiled form.

EN / ENO (Enable / Error display) If the value of EN is "0" when the FFB is called up, the algorithms defined by the FFB are not executed and all outputs contain the previous value. The value of ENO is automatically set to "0" in this case. If the value of EN is "1" when the FFB is called up, the algorithms defined by the FFB are executed. After the error free execution of the algorithms, the ENO value is automatically set to "1". If an error occurs during the execution of the algorithm, ENO is automatically set to "0". The output behavior of the FFB depends whether the FFBs are called up without EN/ENO or with EN=1. If the EN/ENO display is enabled, the EN input must be active. Otherwise, the FFB is not executed. The projection of EN and ENO is enabled/disabled in the block properties dialog box. The dialog box is called up via the menu commands **Objects** → **Properties...** or via a double click on the FFB.

Error When processing a FFB or a Step an error is detected (e.g. unauthorized input value or a time error), an error message appears, which can be viewed with the menu command **Online** → **Online events...** . With FFBs the ENO output is set to "0".

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

Expression Expressions consist of operators and operands.

F

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

Field variables Variables, one of which is assigned, with the assistance of the key word ARRAY (field), a defined Derived data type. A field is a collection of data elements of the same Data type.

FIR filter Finite Impulse Response Filter

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

Function (FUNC) A Program organization unit, which exactly supplies a data element when executing. A function has no internal status information. Multiple call ups of the same function with the same input parameter values always supply the same output values.
Details of the graphic form of function call up can be found in the definition " Function block (Item)". In contrast to the call up of function blocks, the function call ups only have one unnamed output, whose name is the name of the function itself. In FBD each call up is denoted by a unique number over the graphic block; this number is automatically generated and cannot be altered.

Function block (item) (FB) A function block is a Program organization unit, which correspondingly calculates the functionality values, defined in the function block type description, for the output and internal variables, when it is called up as a certain item. All output values and internal variables of a certain function block item remain as a call up of the function block until the next. Multiple call up of the same function block item with the same arguments (Input parameter values) supply generally supply the same output value(s).
Each function block item is displayed graphically by a rectangular block symbol. The name of the function block type is located on the top center within the rectangle. The name of the function block item is located also at the top, but on the outside of the rectangle. An instance is automatically generated when creating, which can however be altered manually, if required. Inputs are displayed on the left side and outputs on the right of the block. The names of the formal input/output parameters are displayed within the rectangle in the corresponding places.
The above description of the graphic presentation is principally applicable to Function call ups and to DFB call ups. Differences are described in the corresponding definitions.

Function block dialog (FBD) One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.

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

Function counter The function counter serves as a unique identifier for the function in a Program or DFB. The function counter cannot be edited and is automatically assigned. The function counter always has the structure: .n.m

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

G

Generic data type A Data type, which stands in for several other data types.

Generic literal If the Data type of a literal is not relevant, simply enter the value for the literal. In this case Concept automatically assigns the literal to a suitable data type.

Global derived data types Global Derived data types are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global DFBs Global DFBs are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global macros Global Macros are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Groups (EFBs) Some EFB libraries (e.g. the IEC library) are subdivided into groups. This facilitates the search for FFBs, especially in extensive libraries.

I

I/O component list The I/O and expert assemblies of the various CPUs are configured in the I/O component list.

IEC 61131-3 International norm: Programmable controllers – part 3: Programming languages.

IEC format (QW1)

In the place of the address stands an IEC identifier, followed by a five figure address:

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

IEC name conventions (identifier)

An identifier is a sequence of letters, figures, and underscores, which must start with a letter or underscores (e.g. name of a function block type, of an item or section). Letters from national sets of characters (e.g. ö,ü, é, õ) can be used, taken from project and DFB names.
Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as different identifiers. Several leading and multiple underscores are not authorized consecutively.
Identifiers are not permitted to contain space characters. Upper and/or lower case is not significant; e.g. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers are not permitted to be Key words.

IIR filter

Infinite Impulse Response Filter

Initial step (starting step)

The first step in a chain. In each chain, an initial step must be defined. The chain is started with the initial step when first called up.

Initial value

The allocated value of one of the variables when starting the program. The value assignment appears in the form of a Literal.

Input bits (1x references)

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

> **Note:** The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 100201 signifies an input bit in the address 201 of the State RAM.

Input parameters (Input)

When calling up a FFB the associated Argument is transferred.

Input words (3x references)

An input word contains information, which come from an external source and are represented by a 16 bit figure. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format.
Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the user data store, i.e. if the reference 300201 signifies a 16 bit input word in the address 201 of the State RAM.

Instantiation

The generation of an Item.

Instruction (IL) Instructions are "commands" of the IL programming language. Each operation begins on a new line and is succeeded by an operator (with modifier if needed) and, if necessary for each relevant operation, by one or more operands. If several operands are used, they are separated by commas. A tag can stand before the instruction, which is followed by a colon. The commentary must, if available, be the last element in the line.

Instruction (LL984) When programming electric controllers, the task of implementing operational coded instructions in the form of picture objects, which are divided into recognizable contact forms, must be executed. The designed program objects are, on the user level, converted to computer useable OP codes during the loading process. The OP codes are deciphered in the CPU and processed by the controller's firmware functions so that the desired controller is implemented.

Instruction list (IL) IL is a text language according to IEC 1131, in which operations, e.g. conditional/ unconditional call up of Function blocks and Functions, conditional/unconditional jumps etc. are displayed through instructions.

INT INT stands for the data type "whole number". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The range of values for variables of this data type is from –2 exp (15) to 2 exp (15) –1.

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

Example
-12, 0, 123_456, +986

INTERBUS (PCP) To use the INTERBUS PCP channel and the INTERBUS process data preprocessing (PDP), the new I/O station type INTERBUS (PCP) is led into the Concept configurator. This I/O station type is assigned fixed to the INTERBUS connection module 180-CRP-660-01.
The 180-CRP-660-01 differs from the 180-CRP-660-00 only by a clearly larger I/O area in the state RAM of the controller.

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

FBI = Function block item
n = Section number (number running)
m = Number of the FFB object in the section (number running)

J

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

K

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

L

Ladder Diagram (LD)
Ladder Diagram is a graphic programming language according to IEC1131, which optically orientates itself to the "rung" of a relay ladder diagram.

Ladder Logic 984 (LL)
In the terms Ladder Logic and Ladder Diagram, the word Ladder refers to execution. In contrast to a diagram, a ladder logic is used by engineers to draw up a circuit (with assistance from electrical symbols),which should chart the cycle of events and not the existing wires, which connect the parts together. A usual user interface for controlling the action by automated devices permits ladder logic interfaces, so that when implementing a control system, engineers do not have to learn any new programming languages, with which they are not conversant.
The structure of the actual ladder logic enables electrical elements to be linked in a way that generates a control output, which is dependant upon a configured flow of power through the electrical objects used, which displays the previously demanded condition of a physical electric appliance.
In simple form, the user interface is one of the video displays used by the PLC programming application, which establishes a vertical and horizontal grid, in which the programming objects are arranged. The logic is powered from the left side of the grid, and by connecting activated objects the electricity flows from left to right.

Landscape format
Landscape format means that the page is wider than it is long when looking at the printed text.

Language element
Each basic element in one of the IEC programming languages, e.g. a Step in SFC, a Function block item in FBD or the Start value of a variable.

Library
Collection of software objects, which are provided for reuse when programming new projects, or even when building new libraries. Examples are the Elementary function block types libraries.
EFB libraries can be subdivided into Groups.

Literals
Literals serve to directly supply values to inputs of FFBs, transition conditions etc. These values cannot be overwritten by the program logic (write protected). In this way, generic and standardized literals are differentiated.
Furthermore literals serve to assign a Constant a value or a Variable an Initial value. The input appears as Base 2 literal, Base 8 literal, Base 16 literal, Integer literal, Real literal or Real literal with exponent.

Local derived data types
Local derived data types are only available in a single Concept project and its local DFBs and are contained in the DFB directory under the project directory.

Local DFBs

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

Local link

The local network link is the network, which links the local nodes with other nodes either directly or via a bus amplifier.

Local macros

Local Macros are only available in a single Concept project and are contained in the DFB directory under the project directory.

Local network nodes

The local node is the one, which is projected evenly.

Located variable

Located variables are assigned a state RAM address (reference addresses 0x,1x, 3x, 4x). The value of these variables is saved in the state RAM and can be altered online with the reference data editor. These variables can be addressed by symbolic names or the reference addresses.

Collective PLC inputs and outputs are connected to the state RAM. The program access to the peripheral signals, which are connected to the PLC, appears only via located variables. PLC access from external sides via Modbus or Modbus plus interfaces, i.e. from visualizing systems, are likewise possible via located variables.

M

Macro

Macros are created with help from the software Concept DFB.
Macros function to duplicate frequently used sections and networks (including the logic, variables, and variable declaration).
Distinctions are made between local and global macros.

Macros have the following properties:

- Macros can only be created in the programming languages FBD and LD.
- Macros only contain one single section.
- Macros can contain any complex section.
- From a program technical point of view, there is no differentiation between an instanced macro, i.e. a macro inserted into a section, and a conventionally created macro.
- Calling up DFBs in a macro
- Variable declaration
- Use of macro-own data structures
- Automatic acceptance of the variables declared in the macro
- Initial value for variables
- Multiple instancing of a macro in the whole program with different variables
- The section name, the variable name and the data structure name can contain up to 10 different exchange markings (@0 to @9).

MMI

Man Machine Interface

Multi element variables

Variables, one of which is assigned a Derived data type defined with STRUCT or ARRAY.
Distinctions are made between Field variables and structured variables.

N

Network

A network is the connection of devices to a common data path, which communicate with each other via a common protocol.

Network node

A node is a device with an address (164) on the Modbus Plus network.

Node address

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

O

Operand An operand is a Literal, a Variable, a Function call up or an Expression.

Operator An operator is a symbol for an arithmetic or Boolean operation to be executed.

Output parameters (Output) A parameter, with which the result(s) of the Evaluation of a FFB are returned.

Output/discretes (0x references) An output/marker bit can be used to control real output data via an output unit of the control system, or to define one or more outputs in the state RAM. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 000201 signifies an output or marker bit in the address 201 of the State RAM.

Output/marker words (4x references) An output/marker word can be used to save numerical data (binary or decimal) in the State RAM, or also to send data from the CPU to an output unit in the control system. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

P

Peer processor The peer processor processes the token run and the flow of data between the Modbus Plus network and the PLC application logic.

PLC Programmable controller

Program The uppermost Program organization unit. A program is closed and loaded onto a single PLC.

Program cycle A program cycle consists of reading in the inputs, processing the program logic and the output of the outputs.

Program organization unit A Function, a Function block, or a Program. This term can refer to either a Type or an Item.

Programming device Hardware and software, which supports programming, configuring, testing, implementing and error searching in PLC applications as well as in remote system applications, to enable source documentation and archiving. The programming device could also be used for process visualization.

Programming redundancy system (Hot Standby) A redundancy system consists of two identically configured PLC devices, which communicate with each other via redundancy processors. In the case of the primary PLC failing, the secondary PLC takes over the control checks. Under normal conditions the secondary PLC does not take over any controlling functions, but instead checks the status information, to detect mistakes.

Project General identification of the uppermost level of a software tree structure, which specifies the parent project name of a PLC application. After specifying the project name, the system configuration and control program can be saved under this name. All data, which results during the creation of the configuration and the program, belongs to this parent project for this special automation.
General identification for the complete set of programming and configuring information in the Project data bank, which displays the source code that describes the automation of a system.

Project data bank The data bank in the Programming device, which contains the projection information for a Project.

Prototype data file (Concept EFB) The prototype data file contains all prototypes of the assigned functions. Further, if available, a type definition of the internal

R

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

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

Real literal

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

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literal with exponent

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

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

Reference

Each direct address is a reference, which starts with an ID, specifying whether it concerns an input or an output and whether it concerns a bit or a word. References, which start with the code 6, display the register in the extended memory of the state RAM.
0x area = Discrete outputs
1x area = Input bits
3x area = Input words
4x area = Output bits/Marker words
6x area = Register in the extended memory

> **Note:** The x, which comes after the first figure of each reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

Register in the extended memory (6x reference)

6x references are marker words in the extended memory of the PLC. Only LL984 user programs and CPU 213 04 or CPU 424 02 can be used.

RIO (Remote I/O)

Remote I/O provides a physical location of the I/O coordinate setting device in relation to the processor to be controlled. Remote inputs/outputs are connected to the consumer control via a wired communication cable.

RP (PROFIBUS) RP = Remote Peripheral

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

Rum-time error Error, which occurs during program processing on the PLC, with SFC objects (i.e. steps) or FFBs. These are, for example, over-runs of value ranges with figures, or time errors with steps.

S

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

Section A section can be used, for example, to describe the functioning method of a technological unit, such as a motor.
A Program or DFB consist of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages can be used within a section.
Each section has its own Document window in Concept. For reasons of clarity, it is recommended to subdivide a very large section into several small ones. The scroll bar serves to assist scrolling in a section.

Separator format (4:00001) The first figure (the Reference) is separated from the ensuing five figure address by a colon (:).

Sequence language (SFC) The SFC Language elements enable the subdivision of a PLC program organizational unit in a number of Steps and Transitions, which are connected horizontally by aligned Connections. A number of actions belong to each step, and a transition condition is linked to a transition.

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

Source code data file (Concept EFB) The source code data file is a usual C++ source file. After execution of the menu command **Library** → **Generate data files** this file contains an EFB code framework, in which a specific code must be entered for the selected EFB. To do this, click on the menu command **Objects** → **Source**.

Standard format (400001) The five figure address is located directly after the first figure (the reference).

Standardized literals

If the data type for the literal is to be automatically determined, use the following construction: 'Data type name'#'Literal value'.

Example
INT#15 (Data type: Integer, value: 15),
BYTE#00001111 (data type: Byte, value: 00001111)
REAL#23.0 (Data type: Real, value: 23.0)

For the assignment of REAL data types, there is also the possibility to enter the value in the following way: 23.0.
Entering a comma will automatically assign the data type REAL.

State RAM

The state RAM is the storage for all sizes, which are addressed in the user program via References (Direct display). For example, input bits, discretes, input words, and discrete words are located in the state RAM.

Statement (ST)

Instructions are "commands" of the ST programming language. Instructions must be terminated with semicolons. Several instructions (separated by semi-colons) can occupy the same line.

Status bits

There is a status bit for every node with a global input or specific input/output of Peer Cop data. If a defined group of data was successfully transferred within the set time out, the corresponding status bit is set to 1. Alternatively, this bit is set to 0 and all data belonging to this group (of 0) is deleted.

Step

SFC Language element: Situations, in which the Program behavior follows in relation to the inputs and outputs of the same operations, which are defined by the associated actions of the step.

Step name

The step name functions as the unique flag of a step in a Program organization unit. The step name is automatically generated, but can be edited. The step name must be unique throughout the whole program organization unit, otherwise an Error message appears.
The automatically generated step name always has the structure: S_n_m

S = Step
n = Section number (number running)
m = Number of steps in the section (number running)

Structured text (ST)

ST is a text language according to IEC 1131, in which operations, e.g. call up of Function blocks and Functions, conditional execution of instructions, repetition of instructions etc. are displayed through instructions.

Structured variables Variables, one of which is assigned a Derived data type defined with STRUCT (structure).
A structure is a collection of data elements with generally differing data types (Elementary data types and/or derived data types).

SY/MAX In Quantum control devices, Concept closes the mounting on the I/O population SY/MAX I/O modules for RIO control via the Quantum PLC with on. The SY/MAX remote subrack has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O system. The SY/MAX I/O modules are performed when highlighting and including in the I/O population of the Concept configuration.

Symbol (Icon) Graphic display of various objects in Windows, e.g. drives, user programs and Document windows.

T

Template data file (Concept EFB) The template data file is an ASCII data file with a layout information for the Concept FBD editor, and the parameters for code generation.

TIME TIME stands for the data type "Time span". The input appears as Time span literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.

Time span literals Permitted units for time spans (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or a combination thereof. The time span must be denoted by the prefix t#, T#, time# or TIME#. An "overrun" of the highest ranking unit is permitted, i.e. the input T#25H15M is permitted.

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

Token The network "Token" controls the temporary property of the transfer rights via a single node. The token runs through the node in a circulating (rising) address sequence. All nodes track the Token run through and can contain all possible data sent with it.

Traffic Cop The Traffic Cop is a component list, which is compiled from the user component list. The Traffic Cop is managed in the PLC and in addition contains the user component list e.g. Status information of the I/O stations and modules.

Transition The condition with which the control of one or more Previous steps transfers to one or more ensuing steps along a directional Link.

U

UDEFB User defined elementary functions/function blocks
Functions or Function blocks, which were created in the programming language C, and are available in Concept Libraries.

UDINT UDINT stands for the data type "unsigned double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1.

UINT UINT stands for the data type "unsigned integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The value range for variables of this type stretches from 0 to (2exp16)-1.

Unlocated variable Unlocated variables are not assigned any state RAM addresses. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the system and can be altered with the reference data editor. These variables are only addressed by symbolic names.

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

V

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

Vertical format Vertical format means that the page is higher than it is wide when looking at the printed text.

W

Warning When processing a FFB or a Step a critical status is detected (e.g. critical input value or a time out), a warning appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output remains at "1".

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

Index