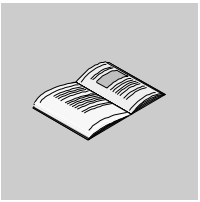
Concept
IEC Block Library
Part: DIAGNO

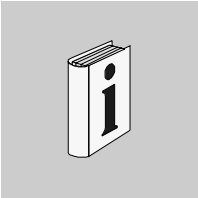
840 USE 504 00 eng Version 2.6

33002212.00

Table of Contents



About the Book



At a Glance

Document Scope

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

Validity Note

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

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

Related Documents

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

User Comments

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

General information about the DIAGNO function block library

I

Overview

At a Glance

This section contains general information about the DIAGNO function block library.

What's in this Part?

This part contains the following chapters:

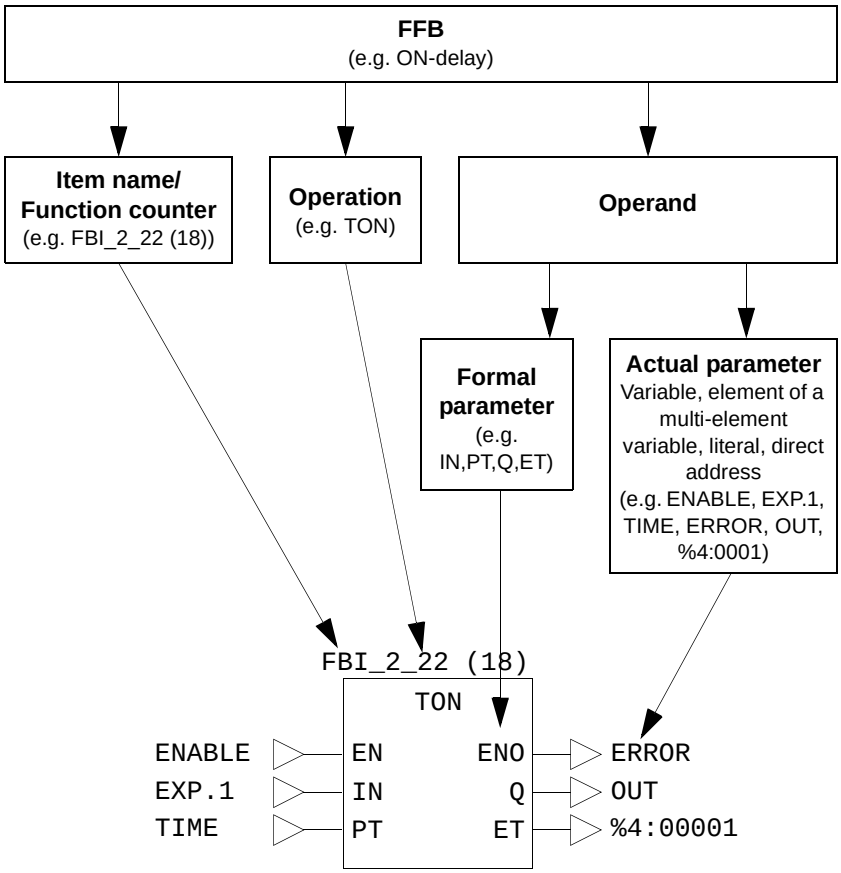
Parameterizing functions and function blocks

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.

Diagnostics

2

Overview

At a Glance

Two subjects are summarized under the subject diagnostics:
- System Diagnostics
- Process Diagnostics

What do we mean by system diagnostics?

System diagnostics deals with the analysis of the PLC status. It is part of the delivered system and always works without any programming.

What do we mean by process diagnostics?

The process diagnostics observes the external PLC environment and recognizes whether or not the process devices are functioning in the specified mode.

What's in this Chapter?

This chapter contains the following topics:

System diagnostics

System diagnostics capabilities

Concept offers the following self test capabilities:

- Gathering of error conditions from bus modules
- I/O
- Communication
- Comparing of programmed configuration with current configuration of bus modules
- Parameter check of the function blocks

Mode of operation for system diagnostics

These system diagnoses are part of the delivered system and always work without any programming. Error situations that arise outside the system, specifically in elementary function blocks, are automatically saved and will be displayed upon request. A message will automatically appear if there is at least one error message. Error messages have a number that identifies their type.

Message display

Error messages from the programming unit are displayed as text in the "Event viewer". Double-clicks on the list text open the image of the section, where the respective EFB is located. Status information is also displayed automatically. Exceeding the time limits in the Steps of the SFC utilizes the same options, except that the step name is displayed instead of the EFB item name and that the open section is the respective SFC section.

Process diagnostics

Process diagnostics capabilities

The process diagnostics observe the external PLC environment and report whether or not the process devices are functioning in default mode. The default behavior is set in the programming phase of the system and is reflected in the diagnostic functions of the runtime system. These diagnostic functions operate with a small subset of the input/output signals of the total process. In the actual process, this subset represents physically existing devices such as cylinders or motors together with their assigned limit switches.

Mode of operation for process diagnostics

The process diagnostics is implemented with the use of EFBs. One special EFB is provided for each diagnostics type. The USR (user runtime system) downloads each EFB with the current parameters, which could also be the result of a link, only once. They can be executed several times and have separate data areas for each Instance.

Diagnostic base EFBs

The diagnostic base EFBs are in the group "Diagnostics".
The following diagnostic base EFBs are available:

- ACT_DIA (See *ACT_DIA: Action diagnostics, p. 19*)
  Action diagnostics with optional motor-like behavior or pulse behavior
- DYN_DIA (See *DYN_DIA: Dynamic diagnostics, p. 27*)
  Dynamic diagnostics
- GRP_DIA (See *GRP_DIA: Signal group monitoring, p. 41*)
  Signal group monitoring
- LOCK_DIA (See *LOCK_DIA: Locking diagnostics, p. 45*)
  Locking diagnostics without reaction input
- PRE_DIA (See *PRE_DIA: Monitoring of process requirements, p. 51*)
  Monitoring of process requirements
- REA_DIA (See *REA_DIA: Reaction diagnostics, p. 55*)
  Reaction diagnostics

Extended diagnostics EFBs

The extended diagnostics EFBs are in the group "Extended".
These function blocks can be used for visualization of the diagnostics in conjunction with one of the following programs:

- Diagnostics Viewer in Concept (**Online** → **Online-Diagnostics...**)
- various diagnostics software

Note: This additional diagnostic information can only be utilized when using function blocks in the FBD (Function Block Dialog) programming language.

The following extended diagnostics EFBs are available:

- XACT (See *XACT: Extended locking/action diagnostics, p. 59*)
  Extended combination of locking and action diagnostics
- XACT_DIA (See *XACT_DIA: Extended action diagnostics, p. 71*)
  Extended action diagnostics with optional motor-like behavior or pulse behavior
- XDYN_DIA (See *XDYN_DIA: Extended dynamic diagnostics, p. 79*)
  Extended dynamic diagnostics
- XGRP_DIA (See *XGRP_DIA: Extended signal group monitoring, p. 85*)
  Extended signal group monitoring
- XLOCK (See *XLOCK: Extended locking diagnostics, p. 65*)
  Extended locking diagnostics with reaction input
- XLOCK_DIA (See *XLOCK_DIA: Extended locking diagnostics, p. 89*)
  Extended locking diagnostics without reaction input
- XPRE_DIA (See *XPRE_DIA: Extended process requirement monitoring, p. 95*)
  Extended monitoring of process requirements
- XREA_DIA (See *XREA_DIA: Extended reaction diagnostics, p. 99*)
  Extended reaction diagnostics

EFB descriptions

II

Overview

At a Glance

These EFB descriptions are documented in alphabetical order.

Note: The number of certain EFB inputs can be increased up to a maximum of 32 by vertically modifying the size of the FFB symbol. Refer to the description of the individual EFBs to determine which ones are concerned.

What's in this Part?

This part contains the following chapters:

ACT_DIA: Action diagnostics

3

Overview

At a Glance

This chapter describes the ACT_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block ACT_DIA is used for action diagnostics.
Action diagnostics is initiated when the defined action becomes active. This action initiates an operation in the process. This operation has to trigger a set reaction. This reaction mostly occurs with a set delay. However, if the reaction does not occur within the tolerance time DTIME, an error situation arises and the error output ERR becomes active. Contrary to the Locking diagnostics, in which the trigger of the diagnostics must remain active at all times, the behavior of the trigger (action) in action diagnostics can vary.
There are 3 different types of behavior:

- M behavior
- I behavior
- MI behavior

These alternatives vary in the behavior of the diagnostics if the action signal becomes "0" before an authorized value is placed at the reaction input.
The monitoring is performed cyclically. The activation of the diagnostics and at the same time the distribution of the cycle load can be achieved through the enable signal "ED".

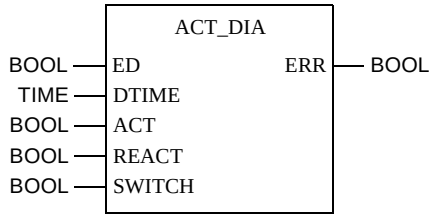
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
ACT	BOOL	Action signal
REACT	BOOL	Reaction signal
SWITCH	BOOL	M/I switch; 0: M behavior, 1: I behavior, 0/1: MI behavior
ERR	BOOL	Error message; 0: no error; 1: Error

Detailed description

Parameterization

In order to control the different modes of behavior (M, I, MI), the appropriate value must be set at SWITCH.

Behavior	SWITCH
M behavior	0
I behavior	1
MI behavior	0 -> 1 (value modification within selected time)

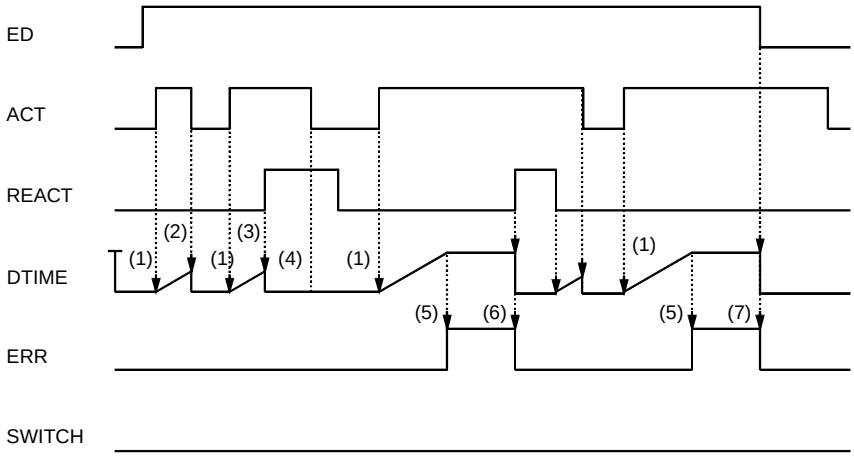
ACT_DIA: M behavior

Motor-like behavior

If the ACT input becomes "1" and REACT does not, the internal counter will be started.
If the action becomes inactive during processing, the monitoring time is stopped/reset or in the event of an error, the error processing is stopped.
When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until ACTION becomes "0", REACT becomes "1" or the diagnostics is deactivated.
If the tolerance time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of an action diagnostics with M behavior is given in the timing diagram.

Timing diagram

M behavior timing diagram



1. The internal time will start when ACT is "1" and REACT is "0".
2. The internal time is stopped/reset when ACT is "0".
3. The internal time is stopped/reset when REACT becomes "1".
4. Once the reaction has been detected, it is insignificant whether or not ACT is active.
5. If the internal time reaches the DTIME value, an error is reported.
6. The error is cancelled and the internal time stopped/reset when REACT becomes "1".
7. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".

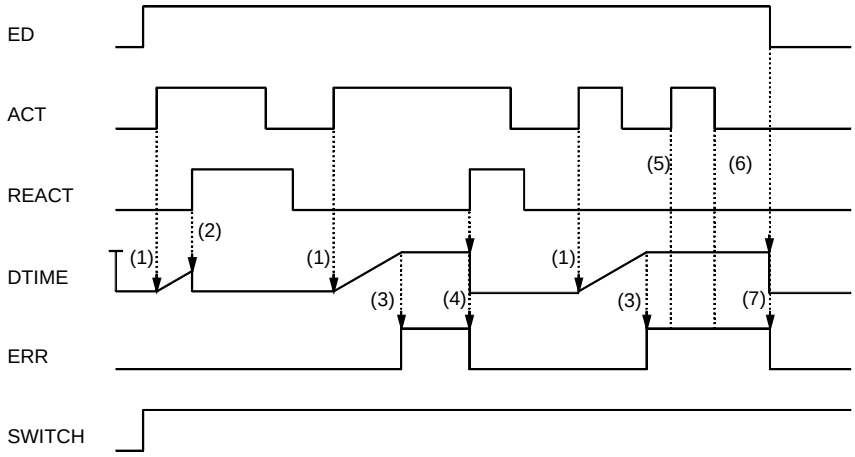
ACT_DIA: I behavior

Pulse behavior

After a transition of the action signal has been detected, diagnostics is activated and the monitoring time will start. The valency of the action signal is no longer significant. When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until REACT becomes "1" or diagnostics is deactivated. The diagnostics will only be terminated (different from the M behavior) with the incoming defined reaction. In order to allow the diagnostics to be terminated in case of error, the ED enable signal has to be projected.
If the tolerance time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of an action diagnostics with I behavior is given in the timing diagram.

Timing diagram

I behavior timing diagram



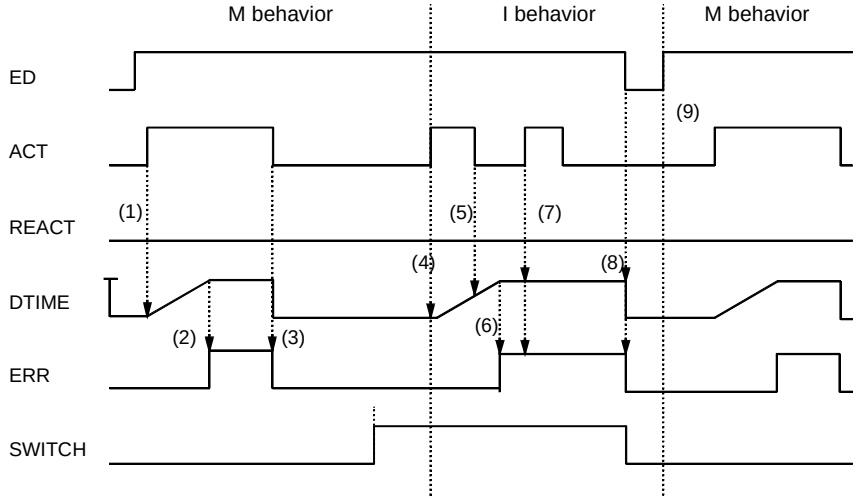
1. The internal time will start when ACT is "1" and REACT is "0".
2. The internal time is stopped/reset when REACT becomes "1".
3. If the internal time reaches the DTIME value, an error is reported.
4. The error is cancelled and the internal time stopped/reset when REACT becomes "1".
5. If the action diagnostics are still in progress (e.g., error handling), a positive transition of the action has no significance.
6. If the action diagnostics are still in progress (e.g., error handling), a negative transition of the action has no significance.
7. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".

ACT_DIA: MI behavior

MI behavior

In the MI behavior, the monitoring begins with M behavior. If the SWITCH signal becomes active during monitoring (transition), the diagnostics switches to I behavior. This is a one-time switch and it is not possible to change back to M behavior during this monitoring cycle.
Since action diagnostics with I behavior can only be terminated via the defined reaction or the ED enable signal, the enable signal has to be projected in the MI behavior as well.
If the tolerance time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of an action diagnostics with MI behavior is given in the timing diagram.

Timing diagram MI behavior timing diagram



1. The internal time will start when ACT is "1" and REACT is "0".
2. If the internal time reaches the DTIME value, an error is reported.
3. With M behavior, the error will be cancelled, and the internal time stopped/reset when ACT becomes "0".
4. If SWITCH is "1" and ACT becomes "1", the diagnostics will switch from M behavior to I behavior. The internal time will also start when ACT is "1" and REACT is "0".
5. If the action diagnostics is still in progress (e.g. internal time started) during I behavior, a negative transition of the action has no significance.
6. If the internal time reaches the DTIME value, an error is reported.
7. If the action diagnostics is still in progress (e.g. internal time started) during I behavior, a positive transition of the action has no significance.
8. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".
9. If the enable signal ED returns to "1" or REACT becomes "1", a switch from I behavior to M behavior occurs.

DYN_DIA: Dynamic diagnostics

4

Overview

At a Glance

This chapter describes the DYN_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block DYN_DIA is used for dynamic diagnostics.
Some processes require that LOCK_DIA (Locking diagnostics), ACT_DIA (Action diagnostics) and REA_DIA (reaction diagnostics) are combined into one unit that monitors the momentary condition of the diagnostics. This is only possible using a special function block, which internally manages the current diagnostics status.
To prevent this function block becoming too complex, only one ED enable signal and one ERR error output were defined.
The monitoring is performed cyclically. Activation of the diagnostics and, at the same time, distribution of the cycle load can be achieved through the enable signal ED.

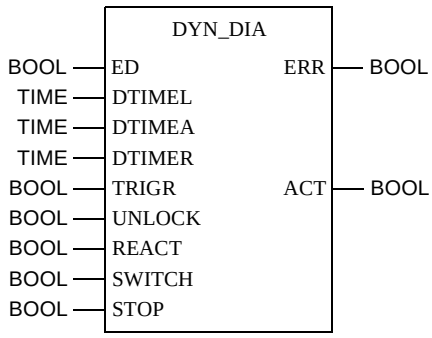
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

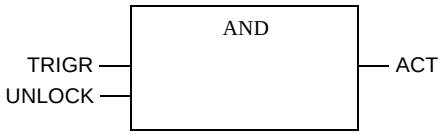
Parameter	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIMEL	TIME	Tolerance time LOCK_DIA (locking diagnostics)
DTIMEA	TIME	Tolerance time ACT_DIA (action diagnostics)
DTIMER	TIME	Tolerance time REA_DIA (reaction diagnostics)
TRIGR	BOOL	Trigger
UNLOCK	BOOL	Locking
REACT	BOOL	Reaction signal
SWITCH	BOOL	M/I switch; 0: M behavior, 1: I behavior, 0/1: MI behavior
STOP	BOOL	Stop signal
ERR	BOOL	Error message; 0: No error; 1: Error
ACT	BOOL	Action enabling

Detailed description

Parameterization

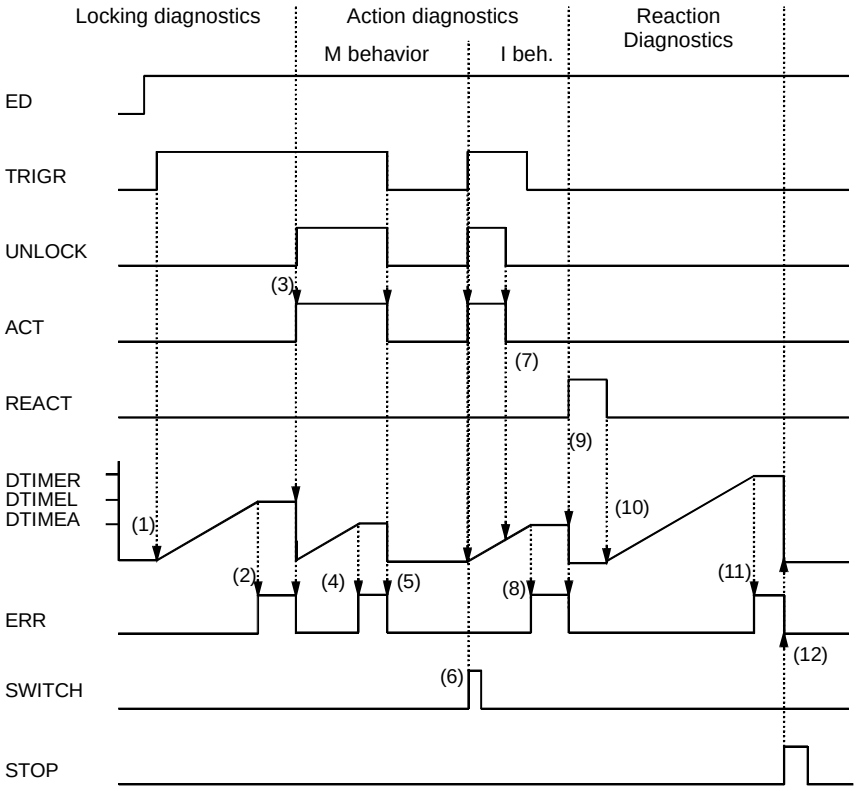
Note: The ACT output is created from TRIGR and UNLOCK with a logical AND. Other inputs (e.g. ED) have no effect on this.

Representation of the relevant inputs for the ACT output



The parameterization of the individual diagnostics types can be found in the descriptions for LOCK_DIA, ACT_DIA and REA_DIA.
An individual tolerance time (DTIMEL, DTIMEA, DTIMER) can be parametered for every diagnostics type.
An example for the process of dynamic diagnostics is given in the timing diagram.

Timing diagram Timing diagram for dynamic diagnostics



1. The internal time starts when TRIGR is "1" and UNLOCK is "0".
2. If the internal time reaches the DTIMEL value, an error will be reported.
3. If UNLOCK becomes "1", the error will be cancelled, the internal time is stopped/ reset, and ACT becomes "1". Activating the action switches to the action diagnostics. As the reaction has still not occurred, the internal time is started.
4. If the internal time reaches the DTIMEA value, an error will be reported.
5. With M behavior, the error will be cancelled, and the internal time stopped/reset when ACT becomes "0".
6. If SWITCH becomes "1" and ACT is "1", a switch from M behavior to I behavior occurs. The internal time will also start when ACT is "1" and REACT is "0".
7. If the action diagnostics is still in progress (e.g. internal time started), a negative transition of the action has no significance.
8. If the internal time reaches the DTIMEA value, an error will be reported.
9. If REACT becomes "1", the internal time is stopped/reset. By activating the reaction, the reaction diagnostics is switched.
10. The internal time will start when REACT becomes "0".

11. If the internal time reaches the DTIMER value, an error will be reported.

12. The error will be cancelled and the internal time is stopped/reset when STOP becomes "1". By activating the stop signal, the locking diagnostics is switched again.

ERR2HMI: Error to HMI

5

Overview

At a Glance

This chapter describes the ERR2HMI block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block ERR2HMI is used to make error data from the internal PLC error buffer, which is detected by the diagnostic EFBs and the sequential function chart, available for diagnostic display in visualization.
The PLC error buffer is part of the runtime system and provides no interface for direct external access. EFBs, that read data from the error buffer and forward it to the corresponding receiver, are used to make the error data available for visualization.
The diagnostics communicates with the block via both the PCV_IN and PCV_OUT data structures. Jobs are assigned to the block in PCV_IN. The result is written to PCV_OUT by the block. PCV_OUT is read by the diagnostic display.
Only one ERR2HMI block may be projected per diagnostic display.

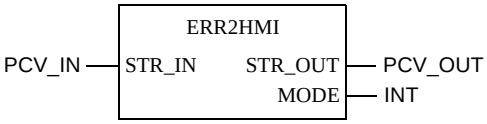
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
STR_IN	PCV_IN	Input data structure
STR_OUT	PCV_OUT	Output data structure
MODE	INT	0 = active, 1 = communication interrupted

Description of the PCV_IN elements:

Element	Data type	Meaning
Job	INT	Processing job for block
Parameter	INT	Parameters for specific jobs, e.g., error selection

Description of the PCV_OUT elements:

Element	Data type	Meaning
response	INT	Processing status: 0 = o.K.
counter	INT	Write counter of error buffer
st_feld [1] ... st_feld [64]	INT	Status fields for the 64 possible error entries
length	INT	Length of error entry
class	INT	Error class
type	INT	Error type
drop	INT	Drop number
q_status	INT	Acknowledgement status
m_status	INT	Message status
t_comes	DINT	Time stamp when error event occurs
t_comes_ms	INT	Time stamp when error event occurs
blob_id	INT	Internal ID character
t_goes	DINT	Time stamp when error event ends
t_goes_ms	INT	Time stamp when error event ends
scan_index	INT	Reference to diagnostic block
blob_adr	DINT	internal address
blob_gen_time	DINT	Time of generation
trans_ID	INT	References to transition
nmb_signals	INT	Number of error signals
error_list [1] ... fehler_liste [20]	INT	Reference to faulty signals

Note: Both data structures, PCV_IN and PCV_OUT, are only used for communication with the diagnostic display and may **not be manipulated** by the user.

ERRMSG: Message for error buffer overflow

6

Overview

At a Glance

This chapter describes the ERRMSG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block ERRMSG is used to display error buffer status information.

A display shows whether there has been a buffer overflow, and a counter measures the number of lost entries.

These values are standardized during the initialization of the error buffer (load program, reset error buffer).

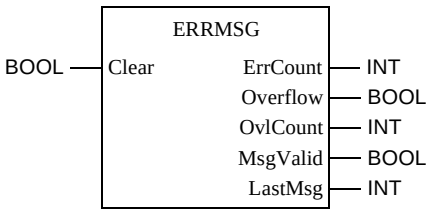
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Representation of the block:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
Clear	BOOL	A 0 -> 1 edge standardizes the inputs to: ● ErrCount = current value ● Overflow = 0 ● OvlCount = 0 ● MsgValid = 0 ● LastMsg = 0
ErrCount	INT	Shows the number of current entries in the error buffer
Overflow	BOOL	1 = An error should be entered in the buffer, but there is no space available for this entry. The output is reset when an error entry is deleted, thus making room for a new entry.
OvlCount	INT	Shows how often the overflow output has been set, i.e. how many entries into the error buffer were lost.
MsgValid	BOOL	1 = The error displayed in LastMsg is current. 0 = The error displayed in Last Msg is no longer valid.
LastMsg	INT	Display of the last Status (See *Error buffer status, p. 40*) encountered in the error buffer. The display remains until a standardization (input clear) is performed. The display's validity is shown in MsgValid.

Detailed description

Function description

Error messages and error withdrawal can appear several times in one cycle, and the queries to the error buffer entries are added. This leads to an update of the output couple LastMsg and MsgValid respectively. The EFB reads the displayed values from the buffer at the point of execution, and thus displays a momentary record which cannot make every state in the buffer available. As the number of overflows (data loss) is summed up at the OvlCount output, this value can still be read later.

Error buffer status

The following messages have been defined:

Value	Meaning	Message in the event viewer
0	no error	-
-4601	Buffer full	Insufficient memory for error buffer
-4602	Diagnostics not installed	Error buffer is not present
-4603	Memory error	Memory management error
-4604	Error index invalid	Incorrect error ID
-4605	MMI not logged in	Incorrect MMI ID
-4606	MMI log-on not possible (too many MMI)	MMI calculation terminated
-4607	Diagnostics data (BLOB) not found	BLOB not loaded
-4608	Block instance not found	No error data present for this element

GRP_DIA: Signal group monitoring

7

Overview

At a Glance

This chapter describes the block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The GRP_DIA function block is used for signal group monitoring.
The monitoring is performed cyclically. The activation of the Diagnostics and thereby the distribution of the cycle load can be achieved through the enable signal "ED".

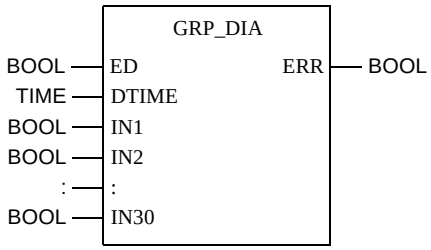
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Description of block parameter:

Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
IN1	BOOL	1. Signal
IN2	BOOL	2. Signal
:	:	:
IN30	BOOL	30. Signal
ERR	BOOL	Error message; 0: no error; 1:error

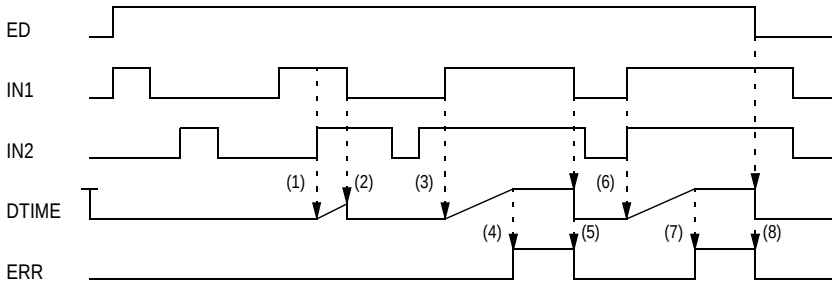
Detailed description

Parameterization

The inputs IN1 and IN2 are monitored whether more than one input is "1". Deactivating the diagnostics or the attached correct values at the inputs will reset the internal counter to "0".
When the default time at the DTIME input has expired, the ERR output displays an error; it remains active until less than two inputs are "1" or the diagnostics is deactivated.
If a tolerance time (DTIME) of "0" is entered, an error message appears immediately if more than one input becomes "1".
An example for the process of signal group monitoring is given in the timing diagram.

Timing diagram

Signal group monitoring timing diagram



1. The internal time is started when IN1 and IN2 simultaneously become"1".
2. The internal time is stopped/reset when IN1 becomes "0".
3. The internal time is started when IN1 and IN2 simultaneously become"1".
4. If the internal time reaches the DTIME value, an error will be reported.
5. The error is cancelled and the internal time is stopped/reset when IN1 becomes "0".
6. The internal time is started when IN1 and IN2 simultaneously become"1".
7. If the internal time reaches the DTIME value, an error will be reported.
8. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".

LOCK_DIA: Locking diagnostics

8

Overview

At a Glance

This chapter describes the LOCK_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The LOCK_DIA function block is used for locking diagnostics and to enable the action.
Locking Diagnostics is activated when the input with the TRIGR signals becomes active.
In control networks the trigger signal TRIGR (e.g. step counter, manual key) does not necessarily initiate the execution of an action directly, but is generally combined with locks from the process. It is therefore possible, that the action ACT only becomes active after a time delay or not at all. It is the task of lock diagnostics, to check whether UNLOCK is enabled within a tolerance time DTIME when the trigger signal is active. In this case the lock diagnostics enables the action ACT. In this instance the trigger signal TRIGR must be active throughout the entire time. An error situation exists, if the lock enabler UNLOCK does not appear within the time period (lock not free). In this instance the action output ACT does not become active, and the error output ERR is set. This error message is terminated when the trigger signal TRIGR is inactive or the lock enabler UNLOCK becomes active.
The lock diagnostics is terminated with an active action output ACT.
The monitoring is performed cyclically. Activation of the diagnostics and thereby distributing the cycle load can be achieved through the enable signal ED. The ED enable signal refers only to the activation of the diagnostics and has no effect on the ACT output.

Note: Contrary to the LOCK-DIA function block, the XLOCK function block has a REACT input that allows the ACT output to be switched off or prevents its activation, respectively, without a locking error being reported.

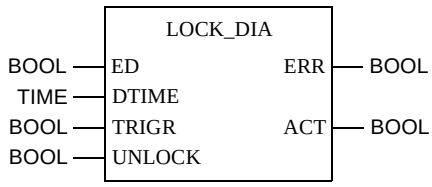
Note: NEVER use diagnostic EFBs in DFBs.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

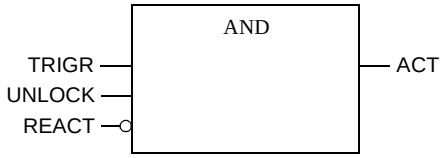
Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
TRIGR	BOOL	Trigger signal
UNLOCK	BOOL	Lock
ERR	BOOL	Error message; 0: no error; 1: Error
ACT	BOOL	Action output

Detailed description

Parameterization

Note: The output is created with a logical AND from TRIGR and UNLOCK. Other inputs (e.g. ED) have no effect on this.

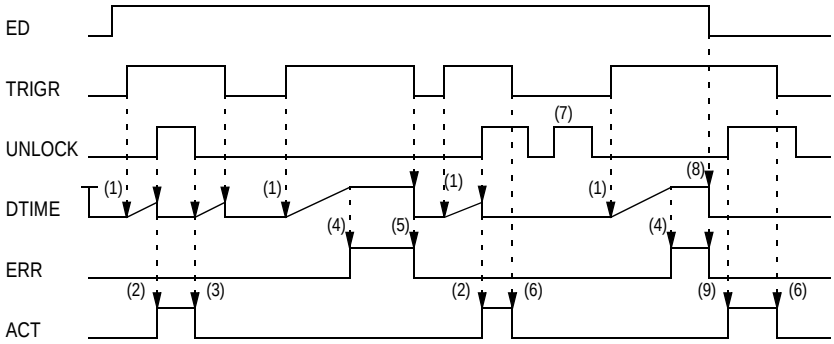
Representation of the relevant inputs for the ACT output



If the TRIGR input (trigger signal) becomes "1" and UNLOCK doesn't, the internal counter will be started.
When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until TRIGR becomes "0", ACT becomes "1" or diagnostics is deactivated.
If the trigger time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of a lock diagnostic is given in the timing diagram.

Timing diagram

Locking diagnostics timing diagram



1. The internal time starts when TRIGR is "1" and UNLOCK is "0".
2. The internal time is stopped/reset and ACT becomes "1" when UNLOCK becomes "1".
3. ACT becomes "0" when UNLOCK becomes "0".
4. If the internal time reaches the DTIME value, an error will be reported.
5. The error is cancelled and the internal time stopped/reset when TRIGR becomes "0".
6. ACT becomes "0" when TRIGR becomes "0".
7. If UNLOCK is "1" and TRIGR is "0", the internal time does not start.
8. The error is cancelled and the internal time stopped/reset, if the enable signal ED becomes "0".
9. If TRIGR and UNLOCK are "1" and ED is "0", action becomes "1". ED has no effect on the ACT signal.

PRE_DIA: Monitoring of process requirements

9

Overview

At a Glance

This chapter describes the PRE_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block PRE_DIA is used for the monitoring process requirements. Process requirements are process characteristics that are absolutely necessary for the operation of the machine or system (e.g. coolant, emergency stop). General requirements are for example requirements for machine operating modes or basic settings
The absence of such requirements is monitored. The monitoring is carried out cyclically. The activation of the diagnostics and thereby the distribution of the cycle load can be achieved through the enable signal ED.
The number of inputs IN can be increased up to 30 by vertically modifying the size of the block.

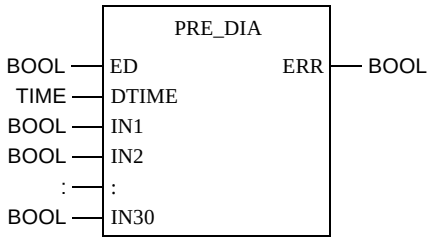
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
IN1	BOOL	1. Process requirement
:	:	:
IN30	BOOL	30. Process requirement
ERR	BOOL	Error message; 0: no error; 1: Error

Detailed description

Parameterization

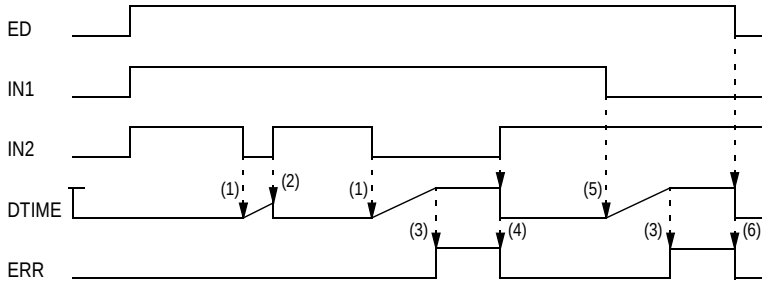
If at least one of the signals connected to INx becomes "0" and the diagnostics is active, the internal counter will be started.

Note: Please note that all visible and unlinked inputs are automatically assigned a "0", i.e. create only as many IN inputs as are actually needed.

The deactivation of the diagnostics or of the attachment of the correct input value stops the counter (the requirements may contain errors during the tolerance time) and sets the counter back to "0".
When the default time at the DTIME input has expired, the ERR output displays an error; it remains active until the requirements are "1" or the diagnostic is deactivated.
If the tolerance time entered is DTIME "0", there is an immediate error message when the static conditional values (INx) become "0".
An example showing the process of monitoring the process requirements can be found in the timing diagram.

Timing diagram

Timing diagram monitoring process requirements



1. The internal time will start when IN2 becomes "0".
2. The internal time is stopped/reset when IN2 becomes "1".
3. If the internal time reaches the DTIME value, an error will be reported.
4. The error is cancelled and the internal time is stopped/reset when IN2 becomes "1".
5. The internal time will start when IN1 becomes "0".
6. The error is cancelled and the internal time stopped/reset, if the enable signal ED becomes "0".

REA_DIA: Reaction diagnostics

10

Overview

At a Glance

This chapter describes the REA_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block REA_DIA is used for reaction Diagnostics.
Once the expected reaction has occurred in the Action diagnostic, the reaction diagnostics check whether the process contains the status.
The process reaction, defined as a term or a signal, is checked through the reaction diagnostics to determine whether the status is stable. During technical processes, it is possible that reactions change momentarily (e.g. hitting limit switches). In order for the reaction diagnostics not to activate the error message ERR directly in such a case, a tolerance time DTIME can be defined. An error signal occurs if this time is exceeded. The error signal becomes inactive when the reaction returns to the setpoint status or when the stop condition is met.
The stop condition terminates reaction diagnostics.
Monitoring is performed cyclically. Activation of the diagnostics and, at the same time, distribution of the cycle load can be achieved through the enable signal ED.

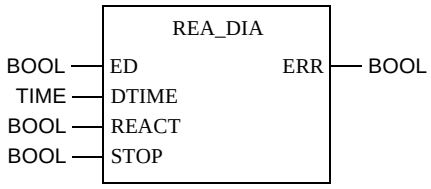
Note: NEVER use diagnostic EFBs in DFBs.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
REACT	BOOL	Reaction signal
STOP	BOOL	Stop signal
ERR	BOOL	Error message; 0: no error; 1: Error

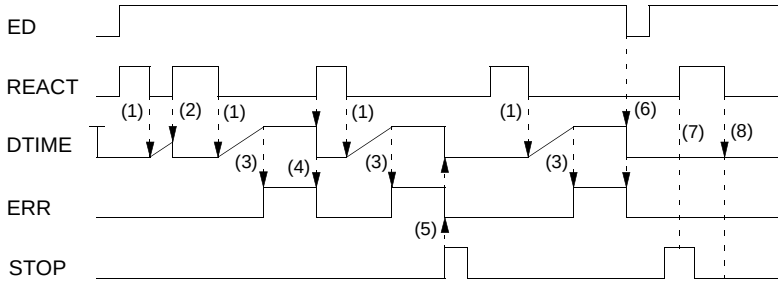
Detailed description

Parameterization

If the REACT input becomes "0", the internal counter will be started.
When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until REACT becomes "1", STOP becomes "1" or the diagnostic is deactivated.
If the tolerance time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of a reaction diagnostic is given in the timing diagram.

Timing diagram

Reaction diagnostics timing diagram



1. The internal time will start when REACT becomes "0".
2. The internal time is stopped/reset when REACT becomes "1".
3. If the internal time reaches the DTIME value, an error will be reported.
4. The error is cancelled and the internal time stopped/reset when REACT becomes "1".
5. The error will be cancelled and the internal time is stopped/reset when STOP becomes "1".
6. The error is cancelled and the internal time stopped/reset, if the enable signal ED becomes "0".
7. If REACT becomes "1", when STOP is "1", the reaction diagnostic is not started.
8. If subsequently REACT becomes "0", the internal time is not started, even if STOP is "0" again.

XACT: Extended locking/action diagnostics

11

Overview

At a Glance

This chapter describes the XACT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The XACT function block provides a combination of locking and action Diagnostics.
The **Locking diagnostics** are activated when the input with the TRIGR signals becomes active.
In control networks the trigger signal TRIGR (e.g. step counter, manual key) does not necessarily initiate the execution of an action directly, but is generally combined with locks from the process. It is therefore possible that the action ACT only becomes active after a time delay or not at all. It is the task of lock diagnostics to check whether UNLOCK is enabled within a tolerance time (DTIMEL) when the trigger signal is active. In this case the lock diagnostics enable the action ACT. In this instance the trigger signal TRIGR must be active throughout the entire time. An error situation exists if the lock enabler UNLOCK does not appear within the time period, (lock not free). In this instance the action output ACT does not become active and the error output ERR is set. In addition, the logic at the UNLOCK input is analyzed and the error is entered in the error buffer. This error information is then displayed in a diagnostic signal on an attached MMI. This error message is terminated when the trigger signal TRIGR becomes inactive or the lock enabler UNLOCK becomes active.
The REACT input provides for the ACT output to be switched off or, respectively, prevents its activation without a locking error being reported.

Note: Please be sure that the REACT input is not negated. To switch off the action, REACT must have the value "1".

An active ACT output terminates the locking diagnostics and starts the action diagnostics.
The **action diagnostics** will be initiated when the defined ACT action becomes active. This action initiates an operation in the process, e.g. an output is set for putting the motor into standby operation. This operation has to trigger a specific reaction. This reaction mostly occurs with a set delay. However, if the reaction does not occur within the tolerance time DTIMEA, an error situation arises and the error output ERR becomes active. In addition, the logic at the REACT input is analyzed and the error is entered in the error buffer. This error information is then displayed in a diagnostic signal on an attached MMI.
Monitoring is performed cyclically. Activation of the diagnostics and, at the same time, distribution of the cycle load can be achieved through the enable signal ED. The ED enable signal refers only to the activation of the diagnostics and has no effect on the ACT output of the locking diagnostics.
A positive edge of the ED enable signal (regardless at what time), or the locking signal UNLOCK becoming inactive while the TRIGR signal is active (in the action diagnostics phase), resets the function block and starts it in the locking diagnostics state.

Note: If in Concept in dialog **Project** → **Code generation options...** you select the option **Include diagnostics information**, the function block provides additional diagnostics codes which can be evaluated using diagnostics software. The function block, however, only makes the diagnostics codes available, if it is used in the FBD programming language.

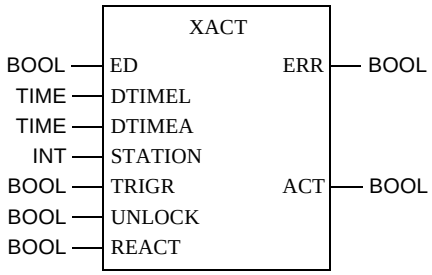
Note: NEVER use diagnostic EFBs in DFBs.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

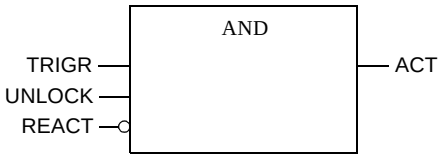
Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIMEL	TIME	Tolerance time for locking diagnostics
DTIMEA	TIME	Tolerance time for action diagnostics
STATION	INT	Drop number (if no entry is made, drop number "0" will be used).
TRIGR	BOOL	Trigger signal
UNLOCK	BOOL	Lock
REACT	BOOL	Reaction input
ERR	BOOL	Error message; 0: no error; 1: Error
ACT	BOOL	Action output

Detailed description

Locking diagnostics parameterization

Note: The ACT output is created with a logical AND from TRIGR and UNLOCK. REACT must not be active in this situation. Other inputs (e.g. ED) have no effect on this.

Representation of the relevant inputs for the ACT output

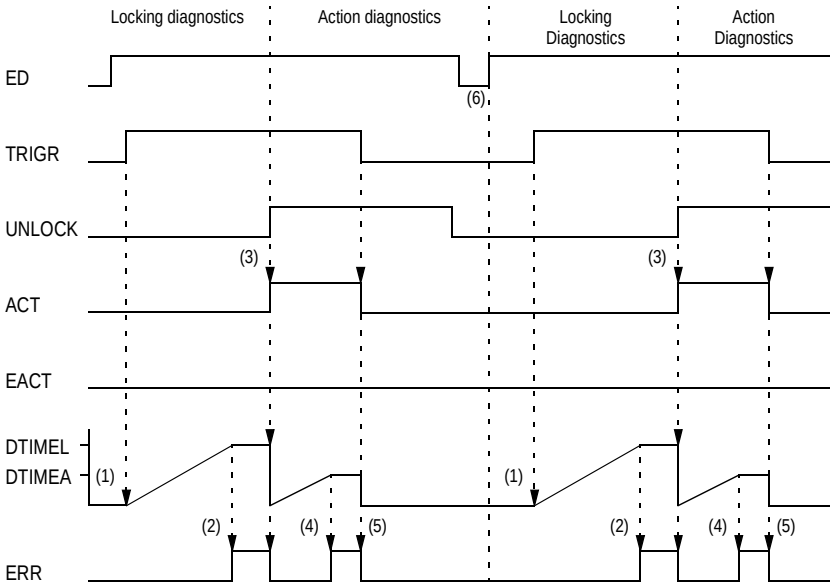


If the TRIGR input (trigger signal) becomes "1" and UNLOCK does not, the internal counter will be started.
When the default time at the DTIMEL input has expired, the ERR output will display an error; it remains active until TRIGR becomes "0", ACT becomes "1" or the diagnostics are deactivated.
If the trigger time (DTIMEL) is entered as "0", an error message is displayed as soon as an error situation occurs.
A detailed example showing the process of locking diagnostics can be found in the locking diagnostics timing diagram.
An active ACT output terminates the locking diagnostics and starts the action diagnostics.

Action diagnostics parameterization

If the ACT output becomes "1" and REACT does not, the internal counter will be started.
If the action becomes inactive during processing, the monitoring time is stopped/ reset or in the event of an error, the error processing is stopped.
When the default time at the DTIMEA input has expired, the ERR output will display an error; it remains active until ACT becomes "0", REACT becomes "1" or the diagnostics are deactivated.
If the tolerance time (DTIMEA) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of lock / action diagnostics is given in the timing diagram.

Timing diagram Locking / action diagnostics timing diagram



1. The internal time starts when TRIGR is "1" and UNLOCK is "0".
2. If the internal time reaches the DTIMEL value, an error will be reported.
3. If UNLOCK becomes "1", the error will be cancelled, the internal time is stopped/ reset, and ACT becomes "1". By activating the action, the action diagnostics are switched. As the reaction has not yet occurred, the internal time is started.
4. If the internal time reaches the DTIMEA value, an error will be reported.
5. The error is cancelled and the internal time stopped/reset when ACT becomes "0".
6. When ED becomes "0", the function block is reset and locking diagnostics will start.

XLOCK: Extended locking diagnostics

12

Overview

At a Glance

This chapter describes the XLOCK block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block XLOCK is used for locking diagnostics and for enabling the action.
Locking diagnostics are activated when the input with the TRIGR signals becomes active.
In control networks the trigger signal TRIGR (e.g. step counter, manual key) does not necessarily initiate the execution of an action directly, but is generally combined with locks from the process. It is therefore possible that the action ACT only becomes active after a time delay or not at all. It is the task of lock diagnostics to check whether UNLOCK is enabled within a tolerance time DTIME when the trigger signal is active. In this case the lock diagnostics enable the action ACT. In this instance the trigger signal TRIGR must be active throughout the entire time. An error situation exists if the lock enable UNLOCK does not appear within the time period, (lock not freed). In this instance the action output ACT does not become active and the error output ERR is set. In addition, the logic at the UNLOCK input is analyzed and the error is entered in the error buffer. This error information is then displayed in a diagnostic signal on an attached MMI. This error message is terminated when the trigger signal TRIGR is inactive or the lock enable UNLOCK becomes active.
Contrary to the LOCK_DIA and XLOCK_DIA function blocks, the XLOCK function block has a REACT input that allows the ACT output to be switched off or prevents its activation, respectively, without a locking error being reported.

Note: Please be sure that the REACT input is not negated. To switch off the action, REACT must have the value "1".

The lock diagnostics terminate with an active action output ACT.
Monitoring is performed cyclically. Activation of the diagnostics and, at the same time, distribution of the cycle load can be achieved through the enable signal ED. The ED enable signal refers only to the activation of the diagnostics and has no effect on the ACT output.

Note: If, in Concept, in the dialog **Project** → **Code generation options...** you select the option **Include diagnostics information**, the function block provides additional diagnostics codes which can be evaluated using diagnostics software. The function block, however, only makes the diagnostics codes available, if used in the FBD programming language.

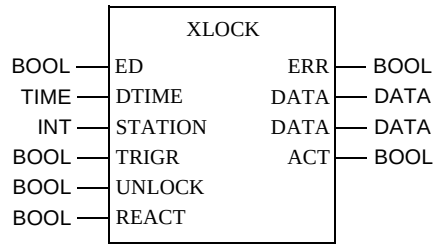
Note: NEVER use diagnostic EFBs in DFBs.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

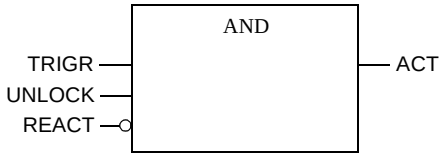
Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
STATION	INT	Drop number (if no entry is made, drop number "0" will be used).
TRIGR	BOOL	Trigger signal
UNLOCK	BOOL	Lock
REACT	BOOL	Reaction input
ERR	BOOL	Error message; 0: no error; 1: Error
ACT	BOOL	Action output

Detailed description

Parameterization

Note: The ACT output is created with a logical AND from TRIGR and UNLOCK. REACT must not be active at that stage. Other inputs (e.g. ED) have no effect on this.

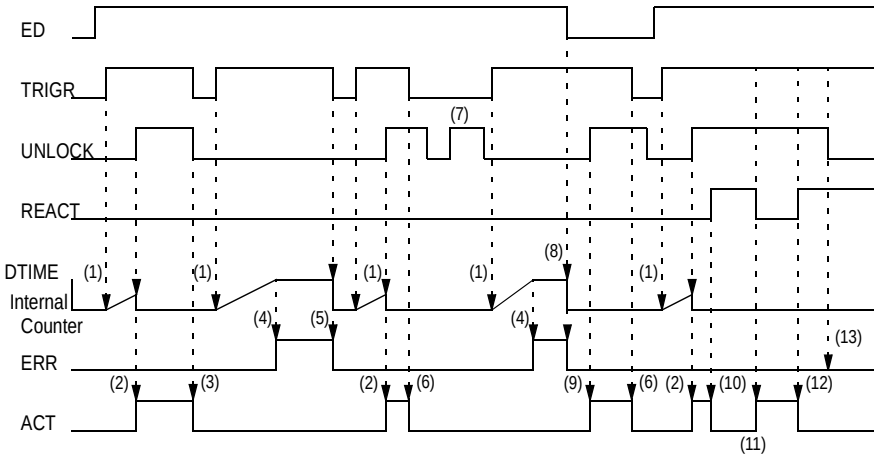
Representation of the relevant inputs for the ACT output



If the TRIGR input (trigger signal) becomes "1" and UNLOCK does not, the internal counter will be started.
When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until TRIGR becomes "0", ACT becomes "1" or the diagnostics are deactivated.
If the trigger time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of a lock diagnostic is given in the timing diagram.

Timing diagram

Locking diagnostics timing diagram



1. The internal time starts when TRIGR is "1" and UNLOCK is "0".
2. The internal time is stopped/reset and ACT becomes "1" when UNLOCK becomes "1".
3. ACT becomes "0" when UNLOCK becomes "0".
4. If the internal time reaches the DTIME value, an error will be reported.
5. The error is cancelled and the internal time stopped/reset when TRIGR becomes "0".
6. ACT becomes "0" when TRIGR becomes "0".
7. If UNLOCK is "1" and TRIGR is "0", the internal time does not start.
8. The error is cancelled and the internal time stopped/reset, if the enable signal ED becomes "0".
9. If TRIGR and UNLOCK are "1" and ED is "0", action becomes "1". ED has no effect on the ACT signal.
10. ACT becomes "0" when REACT becomes "1".
11. ACT becomes "1" when REACT becomes "0" and TRIGR and UNLOCK are "1".
12. ACT becomes "0" when REACT becomes "1" and TRIGR and UNLOCK are "1".
13. If UNLOCK is "0" and REACT is "1", ERR remains "0". (No error will be reported because there was a reaction to the action.)

XACT_DIA: Extended action diagnostics

13

Overview

At a Glance

This chapter describes the XACT_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block XACT_DIA is used for action diagnostics.
Action diagnostics are initiated when the defined action becomes active. This action initiates an operation in the process. This operation has to trigger a specific reaction. This reaction mostly occurs with a set delay. However, if the reaction does not occur within the tolerance time DTIME, an error situation arises and the error output ERR becomes active. In contrast to Locking diagnostics, in which the trigger of the diagnostics must remain active at all times, the behavior of the trigger (action) in action diagnostics can vary.
There are 3 different types of behavior:

- M behavior
- I behavior
- MI behavior

These possibilities vary in the behavior of the diagnostics if the action signal becomes "0" before an authorized value is placed at the reaction input.
The monitoring is performed cyclically. Activating the diagnostics, which results in a distribution of the cycle load, can be achieved through the enable signal "ED".

Note: If, in Concept, in the dialog **Project** → **Code generation options...** you activate the option **Include diagnostics information**, the function block provides additional diagnostics codes which can be evaluated using diagnostics software. However, the diagnostics codes are only made available from the function block if the function block is used in the FBD programming language.

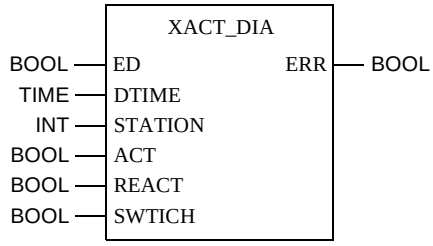
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
STATION	INT	Drop number (if no entry is made, drop number "0" will be used).
ACT	BOOL	Action signal
REACT	BOOL	Reaction signal
SWITCH	BOOL	M/I switch; 0: M behavior, 1: I behavior, 0/1: MI behavior
ERR	BOOL	Error message; 0: no error; 1: Error

Detailed description

Parameterization

In order to control the different types of behavior (M, I, MI behavior), the appropriate value must be set at SWITCH and ACT.

Behavior	SWITCH	ACT
M behavior	0	0 or 1
I behavior	1	0 or 1
MI behavior	0 -> 1 (value modification within selected time)	1

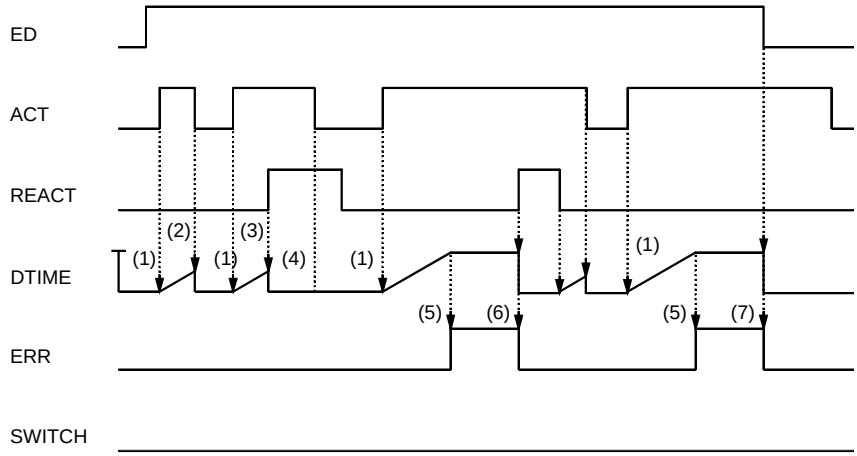
XACT_DIA: M behavior

Motor-like behavior

If the ACT input becomes "1" and REACT does not, the internal counter starts.
If the action becomes inactive during processing, the monitoring time is stopped/reset or, in the event of an error, the error processing is stopped.
When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until ACTION becomes "0", REACT becomes "1" or the diagnostics are deactivated.
If the tolerance time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of action diagnostics with M behavior is given in the timing diagram.

Timing diagram

M behavior timing diagram



1. The internal time is started when ACT is "1" and REACT is "0".
2. The internal time is stopped/reset when ACT is "0".
3. The internal time is stopped/reset when REACT is "1".
4. Once the reaction has been detected, it is insignificant whether or not ACT is active.
5. If the internal time reaches the DTIME value, an error will be reported.
6. The error is cancelled and the internal time stopped/reset when REACT becomes "1".
7. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".

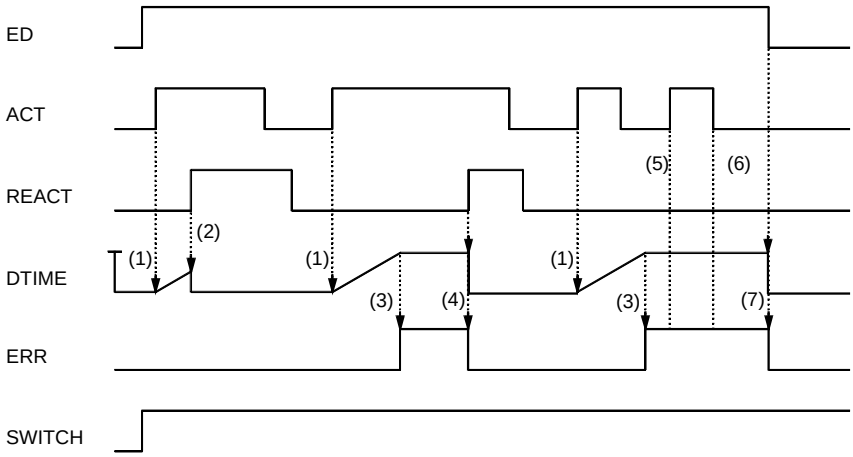
XACT_DIA: I behavior

Pulse behavior

After an edge of the action signal has been detected, diagnostics is activated and the monitoring time will start. The action signal's valency is no longer influential. When the default time at the DTIME input has expired, the ERR output will display an error; it remains active either until REACT becomes "1" or diagnostics is deactivated.
The diagnostics will only be terminated (in contrast to the case of M behavior) by the incoming defined reaction. In order to allow the diagnostics to be terminated despite any errors which may occur, the ED enable signal must be projected.
If the tolerance time (DTIME) is entered as "0", an error message is displayed as soon as an error situation arises.
An example for the process of action diagnostics with I behavior is given in the timing diagram.

Timing diagram

I behavior timing diagram



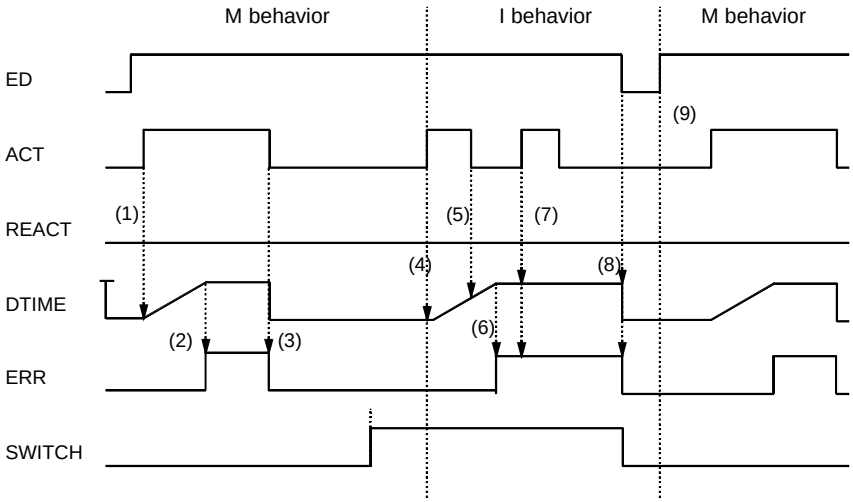
1. The internal time will start when ACT is "1" and REACT is "0".
2. The internal time is stopped/reset when REACT is "1".
3. If the internal time reaches the DTIME value, an error will be reported.
4. The error is cancelled and the internal time stopped/reset when REACT becomes "1".
5. If the action diagnostics are still being processed (e.g. error handling), an action positive edge is not significant.
6. If the action diagnostics are still being processed (e.g. error handling), an action negative edge is not significant.
7. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".

XACT_DIA: MI behavior

MI behavior

In the case of MI behavior, monitoring begins with M behavior. If during monitoring the ACT signal is "1" and the SWITCH signal becomes active (0 -> 1 edge), the diagnostics switch to I behavior.
This switch can only take place once and it is not possible to change back to M behavior during this monitoring cycle.
Since action diagnostics with I behavior can only be terminated via the defined reaction or the ED enable signal, the enable signal must also be projected in the case of MI behavior.
If the tolerance time (DTIME) is entered as "0", an error message is immediately displayed if an error situation occurs.
An example for the process of action diagnostics with MI behavior is given in the timing diagram.

Timing diagram MI behavior timing diagram



1. The internal time will start when ACT is "1" and REACT is "0".
2. If the internal time reaches the DTIME value, an error will be reported.
3. With M behavior, the error will be cancelled, and the internal time stopped/reset when ACT becomes "0".
4. If SWITCH is "1" and ACT becomes "1", the diagnostics will switch from M behavior to I behavior. The internal time will also start when ACT is "1" and REACT is "0".
5. If the action diagnostics are still being processed (e.g. internal time started) during I behavior, a negative edge of the action is not significant.
6. If the internal time reaches the DTIME value, an error will be reported.
7. If the action diagnostics are still being processed (e.g. internal time started) during I behavior, a positive edge of the action is not significant.
8. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".
9. If the enable signal ED returns to "1" or REACT becomes "1", a switch from I behavior to M behavior occurs.

XDYN_DIA: Extended dynamic diagnostics

14

Overview

At a Glance

This chapter describes the XDYN_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block XDYN_DIA is used for dynamic diagnostics.
For certain processes it is necessary to combine XLOCK_DIA (Extended locking diagnostics), XACT_DIA (Extended action diagnostics) and XREA_DIA (Extended reaction diagnostics) in one unit, which monitors the current state of the diagnostics. This is only possible via a special function block, which internally manages the current diagnostics status.
To prevent this function block from becoming too complicated, only one ED enable signal and one ERR error output have been defined.
The monitoring is performed cyclically. Activation of the diagnostics causing distribution of the cycle load can be achieved through the enable signal ED.

Note: If, in Concept, in the dialog **Project** → **Code generation options...** you select the option **Include diagnostics information**, the function block provides additional diagnostics codes which can be evaluated using diagnostics software. However, the function block only makes the diagnostics codes available if the function block is used in the FBD programming language.

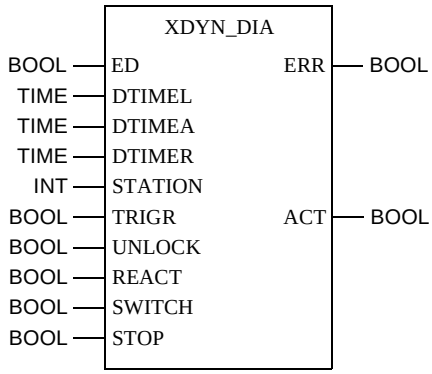
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

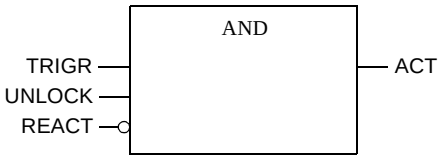
Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIMEL	TIME	Tolerance time LOCK_DIA (locking diagnostics)
DTIMEA	TIME	Tolerance time ACT_DIA (action diagnostics)
DTIMER	TIME	Tolerance time REA_DIA (reaction diagnostics)
STATION	INT	Station number (if no entry is made, station number "0" will be used).
TRIGR	BOOL	Trigger
UNLOCK	BOOL	Lock
REACT	BOOL	Reaction signal
SWITCH	BOOL	M/I switch; 0: M behavior, 1: I behavior, 0/1: MI behavior
STOP	BOOL	Stop signal
ERR	BOOL	Error message; 0: No error; 1: Error
ACT	BOOL	Action enabling

Detailed description

Parameterization

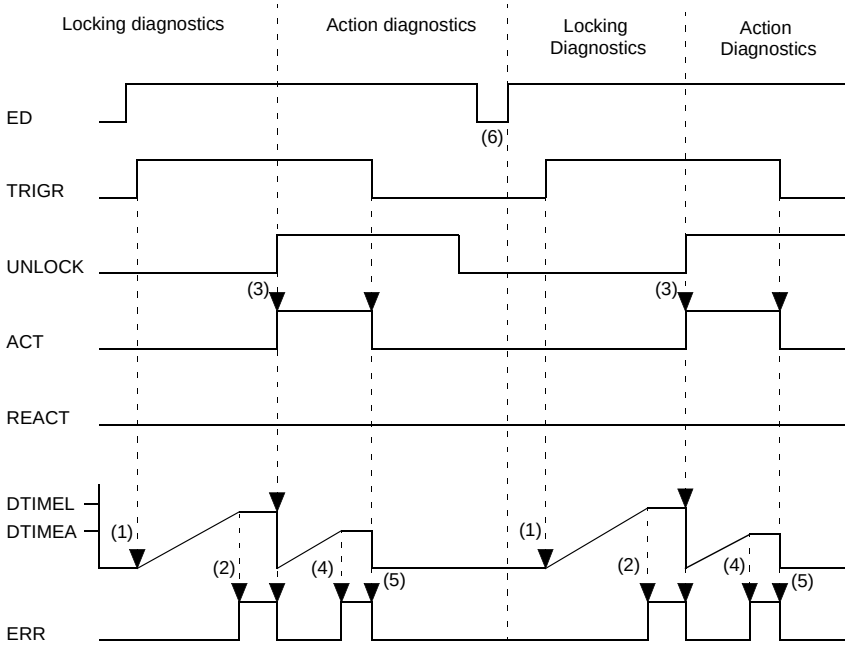
Note: The output is created with a logical AND from TRIGR and UNLOCK. Other inputs (e.g. ED) have no effect on this.

Representation of the relevant inputs for the ACT output



Parameterization for each diagnostics type can be found in the description for XLOCK_DIA, XACT_DIA and XREA_DIA.
An individual tolerance time (DTIMEL, DTIMEA, DTIMER) can be parametered for every diagnostics type.
An example for the process of a dynamic diagnostic is given in the timing diagram.

Timing diagram Timing diagram for dynamic diagnostics



1. The internal time starts when TRIGR is "1" and UNLOCK is "0".
2. If the internal time reaches the DTIMEL value, an error will be reported.
3. If UNLOCK becomes "1", the error will be cancelled, the internal time is stopped/ reset, and ACT becomes "1". The pass power of the action causes a switch to action diagnostics. As the reaction has still not occurred, the internal time is started.
4. If the internal time reaches the DTIMEA value, an error will be reported.
5. With M behavior, the error will be cancelled, and the internal time stopped/reset when ACT becomes "0".
6. If SWITCH becomes "1" and ACT is "1", a switch from M behavior to I behavior occurs. The internal time will also start when ACT is "1" and REACT is "0".
7. If the action diagnostics are still in progress (e.g. internal time started) during I behavior, a negative edge of the action is not significant.
8. If the internal time reaches the DTIMEA value, an error will be reported.
9. If REACT becomes "1", the internal time is stopped/reset. The pass power of the reaction causes a switch to reaction diagnostics.
10. The internal time will start when REACT becomes "0".
11. If the internal time reaches the DTIMER value, an error will be reported.
12. The error will be cancelled and the internal time is stopped/reset when STOP becomes "1". The pass power of the stop signal causes a switch back to locking diagnostics.

XGRP_DIA: Extended signal group monitoring

15

Overview

At a Glance

This chapter describes the XGRP_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The XGRP_DIA function block is used for signal group monitoring.
The monitoring is performed cyclically. Activating the diagnostics which causes the distribution of the cycle load, can be achieved through the enable signal "ED".

Note: If you activate in Concept the option **Include diagnostics information** (in the dialog **Project** → **Code generation options…**), the function block provides additional diagnostics codes which can be evaluated using diagnostics software. However, the function block only makes the diagnostics codes available if the function block is used in the FBD programming language.

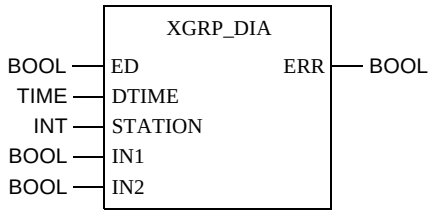
Note: NEVER use diagnostic EFBs in DFBs.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
STATION	INT	Station number (if no entry is made, station number "0" will be used).
IN1	BOOL	1. Signal
IN2	BOOL	2. Signal
ERR	BOOL	Error message; 0: no error; 1:error

Detailed description

Parameterization

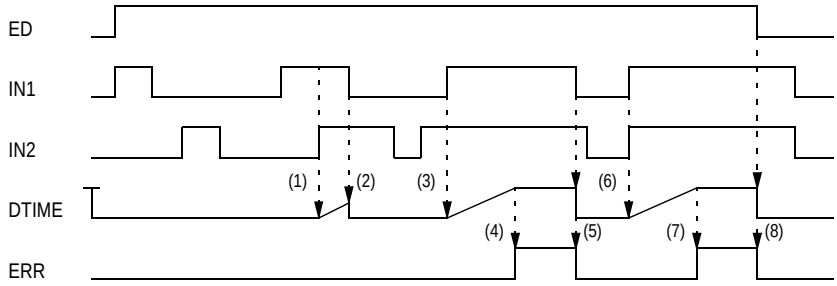
The inputs IN1 and IN2 are monitored to determined whether more than one input is "1".

Note: Unlike GRP_DIA, this function block only possesses two INx-inputs, as with the XGRP_DIA there is an additional analysis of the faulty signals and an entry in the error buffer. This analysis can only be made for 2 signals.

Deactivating the diagnostics or the attached correct values at the inputs will reset the internal counter to "0".
When the default time at the DTIME input has expired, the ERR output displays an error; it remains active until fewer than two inputs are "1" or until the diagnostics is deactivated.
If a tolerance time (DTIME) of "0" is entered, an error message comes up immediately if more than one input becomes "1".
An example for the process of signal group monitoring is given in the timing diagram

Timing diagram

Signal group monitoring timing diagram



1. The internal time is started when IN1 and IN2 simultaneously become "1".
2. The internal time is stopped/reset when IN1 becomes "0".
3. The internal time is started when IN1 and IN2 become "1" simultaneously.
4. If the internal time reaches the DTIME value, an error will be reported.
5. The error is cancelled and the internal time is stopped/reset when IN1 becomes "0".
6. The internal time is started when IN1 and IN2 become"1"simultaneously.
7. If the internal time reaches the DTIME value, an error will be reported.
8. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".

XLOCK_DIA: Extended locking diagnostics

16

Overview

At a Glance

This chapter describes the XLOCK_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block XLOCK_DIA is used for locking diagnostics and to enable the action.
Locking diagnostics are activated when the input with the TRIGR trigger signals becomes active.
In control networks the trigger signal TRIGR (e.g. step counter, manual key) does not necessarily initiate the execution of an action directly, but is generally combined with locks from the process. It is therefore possible that the action ACT only becomes active after a time delay or not at all. It is the task of lock diagnostics to check whether UNLOCK is enabled within a tolerance time DTIME when the trigger signal is active. In this case the lock diagnostics enables the action ACT. In this instance the trigger signal TRIGR must be active throughout the entire time An error situation exists if the lock enabler UNLOCK does not occur within the time period, (lock not free). In this instance the action output ACT does not become active and the error output ERR is set. This error message is terminated when the trigger signal TRIGR is inactive or the lock enabler UNLOCK becomes active.
The lock diagnostics is terminated with an active action output ACT.
The monitoring is carried out cyclically. The activation of the diagnostics which causes the distribution of the cycle load can be achieved through the enable signal ED. The ED enable signal refers only to the activation of the diagnostics and has no effect on the ACT output.

Note: Unlike the XLOCK_DIA function block, the XLOCK function block has a reaction input REACT, that enables the action output ACT to be switched off, i.e. hindering its activation, without a lock error being displayed.

Note: If, in Concept, in the dialog **Project** → **Code generation options...** you select the option **Include diagnostics information**, the function block provides additional diagnostics codes which can be evaluated using diagnostics software. However, the function block only makes diagnostics codes available if the function block is used in the FBD programming language.

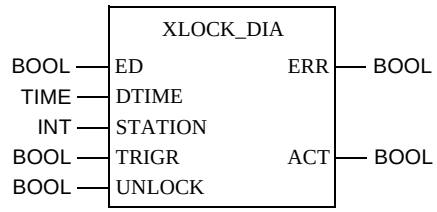
Note: NEVER use diagnostic EFBs in DFBs.

As additional parameters EN and ENO can be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

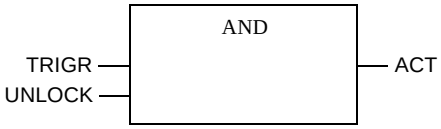
Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
STATION	INT	Station number (if no entry is made, station number "0" will be used).
TRIGR	BOOL	Trigger signal
UNLOCK	BOOL	Lock
ERR	BOOL	Error message; 0: no error; 1: Error
ACT	BOOL	Action output

Detailed description

Parameterization

Note: The output is created with a logical AND from TRIGR and UNLOCK. Other inputs (e.g. ED) have no effect on this.

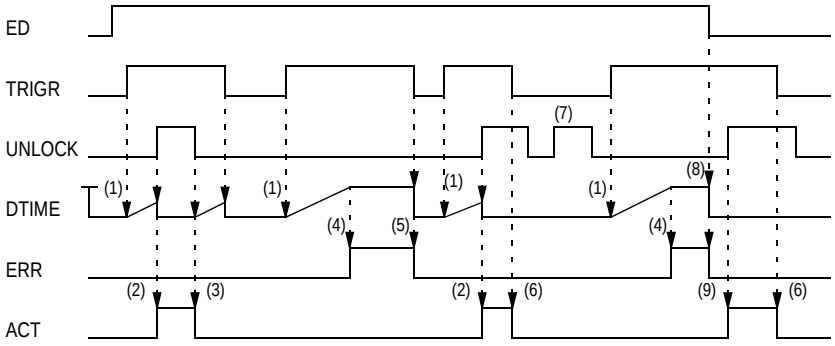
Representation of the relevant inputs for the ACT output



If the TRIGR input (trigger signal) becomes "1" and UNLOCK doesn't, the internal counter will be started.
When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until TRIGR becomes "0", ACT becomes "1" or diagnostics is deactivated.
If the trigger time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
An example for the process of a lock diagnostic is given in the timing diagram.

Timing diagram

Locking diagnostics timing diagram



1. The internal time starts when TRIGR is "1" and UNLOCK is "0".
2. The internal time is stopped/reset and ACT becomes "1" when UNLOCK becomes "1".
3. ACT becomes "0" when UNLOCK becomes "0".
4. If the internal time reaches the DTIME value, an error will be reported.
5. The error is cancelled and the internal time stopped/reset when TRIGR becomes "0".
6. ACT becomes "0" when TRIGR becomes "0".
7. If UNLOCK is "1" and TRIGR is "0", the internal time does not start.
8. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".
9. If TRIGR and UNLOCK are "1" and ED is "0", action becomes "1". ED has no effect on the ACT signal.

XPRE_DIA: Extended process requirement monitoring

17

Overview

At a Glance

This chapter describes the XPRE_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block XPRE_DIA is used for the monitoring of process requirements. Process requirements or preconditions are process characteristics that are absolutely necessary for the operation of the machine or system (e.g. coolant, emergency stop). General requirements are for example requirements for machine operating modes or basic settings.
The absence of such requirements is monitored. The monitoring is carried out cyclically. The activation of the diagnostics which causes the distribution of the cycle load can be achieved through the enable signal ED.
The number of inputs IN can be increased up to a maximum of 30 by vertically modifying the size of the block.

Note: If, in Concept, in the dialog **Project** → **Code generation options...** you select the option **Include diagnostics information**, the function block provides additional diagnostics codes which can be evaluated using diagnostics software. However, the function block only makes the diagnostics codes available if the function block is used in the FBD programming language.

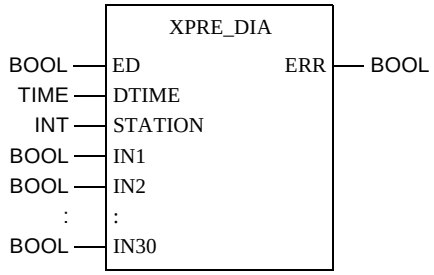
Note: NEVER use diagnostic EFBs in DFBs.

As additional parameters EN and ENO can be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
STATION	INT	Station number (if no entry is made, station number "0" will be used).
IN1	BOOL	1. Process requirement
:	:	:
IN30	BOOL	30. Process requirement
ERR	BOOL	Error message; 0: no error; 1: Error

Detailed description

Parameterization

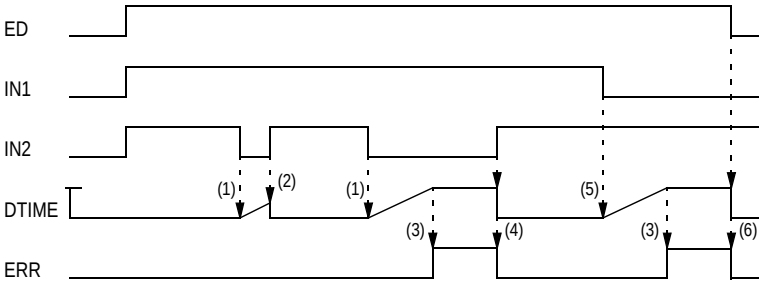
If at least one of the signals connected to INx becomes "0" and the diagnostics are active, the internal counter will be started.

Note: Please note that all visible and unlinked inputs are automatically assigned a "0", i.e. create only as many IN inputs as are actually needed.

The deactivation of the diagnostics or of the attachment of the correct input value stops the counter (the requirements may be contain errors during the tolerance time) and sets the counter back to "0".
When the default time at the DTIME input has expired, the ERR output displays an error; it remains active until the requirements are "1" or the diagnostics are deactivated.
If the tolerance time entered is DTIME "0", there is an immediate error message when the static conditional values (INx) become "0".
An example for process requirement monitoring is given in timing diagram.

Timing diagram

Monitoring of process requirements timing diagram



1. The internal time will start when IN2 becomes "0".
2. The internal time is stopped/reset when IN2 becomes "1".
3. If the internal time reaches the DTIME value, an error will be reported.
4. The error is cancelled and the internal time is stopped/reset when IN2 becomes "1".
5. The internal time will start when IN1 becomes "0".
6. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".

XREA_DIA: Extended reaction diagnostics

18

Overview

At a Glance

This chapter describes the XREA_DIA block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block REA_DIA is used for reaction diagnostics.
Once the expected reaction has occurred in the actions diagnostics, the reaction diagnostics are checked to ascertain whether the process contains the status.
The process reaction, defined as a term or a signal, is checked through the reaction diagnostics to determine whether the status is stable During technical processes it can occur that the reactions change momentarily (e.g. hitting limit switches). In order for the reaction diagnostics not to activate the error message ERR directly in such a case, a tolerance time DTIME can be defined. An error signal occurs if this time is exceeded. The error signal becomes inactive when the reaction returns to the setpoint status or when the stop condition is met.
The stop condition terminates reaction diagnostics.
The monitoring is carried out cyclically. The activation of the diagnostics and thereby the distribution of the cycle load can be achieved through the enable signal ED.

Note: If, in Concept, in the dialog **Project** → **Code generation options...** you select the option **Include diagnostics information**, the function block provides additional diagnostics codes which can be evaluated using diagnostics software. However, the function block only makes the diagnostics codes available if the function block is used in the FBD programming language.

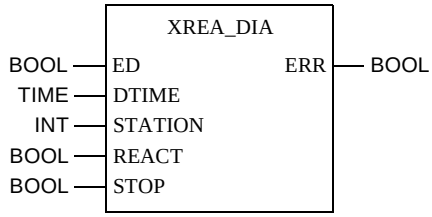
Note: NEVER use diagnostic EFBs in DFBs.

As additional parameters EN and ENO can be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
STATION	INT	Drop number (if no entry is made, drop number "0" will be used).
REACT	BOOL	Reaction signal
STOP	BOOL	Stop signal
ERR	BOOL	Error message; 0: no error; 1: Error

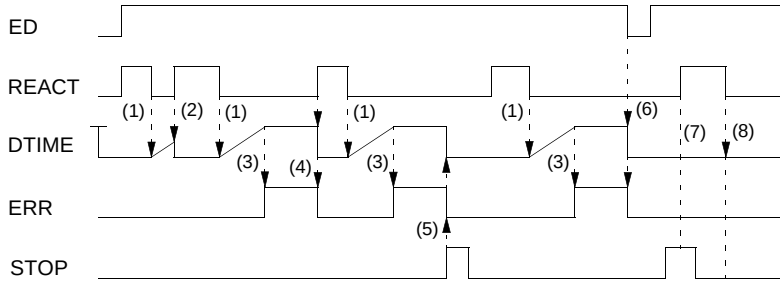
Detailed description

Parameterization

If the REACT input becomes "0", the internal counter will be started.
When the default time at the DTIME input has expired, the ERR output will display an error; it remains active until REACT becomes "1", STOP becomes "1" or the diagnostics are deactivated.
If the tolerance time (DTIME) is entered as "0", an error message is displayed as soon as an error situation occurs.
The timing diagram provides an example for the process of a reaction diagnostics

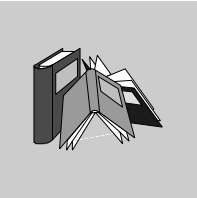
Timing diagram

Reaction diagnostics timing diagram



1. The internal time will start when REACT becomes "0".
2. The internal time is stopped/reset when REACT is "1".
3. If the internal time reaches the DTIME value, an error will be reported.
4. The error is cancelled and the internal time stopped/reset when REACT becomes "1".
5. The error will be cancelled and the internal time is stopped/reset when STOP becomes "1".
6. The error is cancelled and the internal time stopped/reset, if the enable signal ED is "0".
7. If REACT becomes "1", when STOP is "1", the reaction diagnostics are not started.
8. If REACT subsequently becomes "0", the internal time is not started, even if STOP is "0" again.

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** → **Online events...** . 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

F

G

L

M

P

R

S

X