Unity Pro
Diagnostics
Block Library

04/2015

33002523.14

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This document describes the functions and function blocks of the Diagnostics library.

Validity Note

This document is valid for Unity Pro 10.0 or later.

Related Documents

Title of Documentation	Reference Number
Unity Pro Program Languages and structure, Reference Manual	35006144 (English), 35006145 (French), 35006146 (German), 35013361 (Italian), 35006147 (Spanish), 35013362 (Chinese)
Unity Pro System Bits and Words, Reference Manual	EIO0000002135 (English), EIO0000002136 (French), EIO0000002317 (German), EIO0000002138 (Italian), EIO0000002139 (Spanish), EIO0000002140 (Chinese)

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

Part I
General

At a glance

This section contains general information about the Diagnostics library.

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

What Is in This Part?

This part contains the following chapters:

Chapter 1
Block Types and their Applications

Overview

This chapter describes the different block types and their applications.

What Is in This Chapter?

This chapter contains the following topics:

Block Types

Block Types

Block Types

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

There are the following types of block:

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

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

Elementary Function

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

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

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

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

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

Elementary Function Block

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

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

Derived Function Block

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

Procedure

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

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

Procedures do not return a value.

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

There is no visual difference between procedures and elementary functions.

⚠CAUTION

UNEXPECTED BEHAVIOR OF EQUIPMENT

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

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

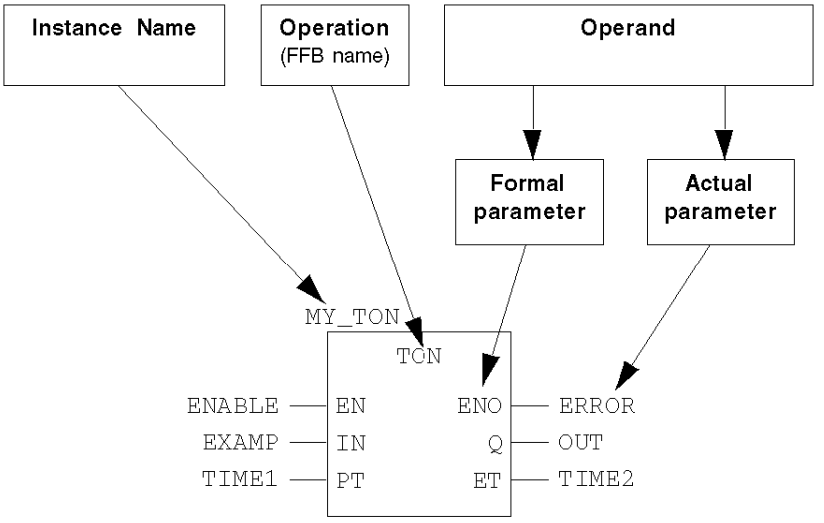
NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity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

FFB Structure

Structure

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

Call of a function block in the FBD programming language:



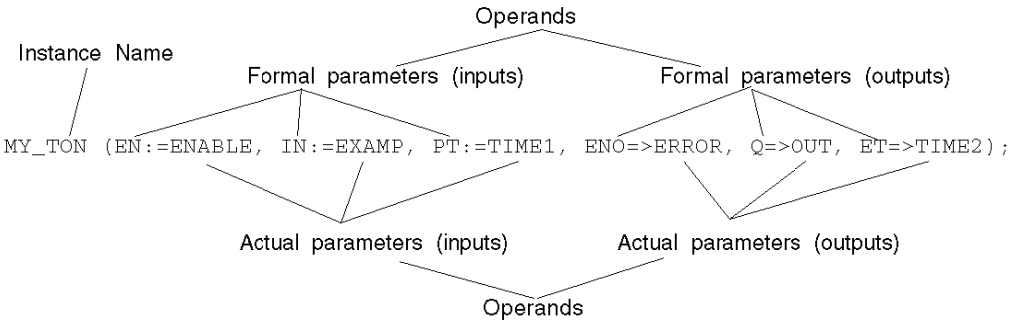
⚠CAUTION

UNEXPECTED APPLICATION BEHAVIOR

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

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

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



Operation

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

Operand

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

Formal/actual parameters

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

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

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

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

FFB Call in IL/ST

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

Example of a formal function call:

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

Example of an informal function call:

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

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

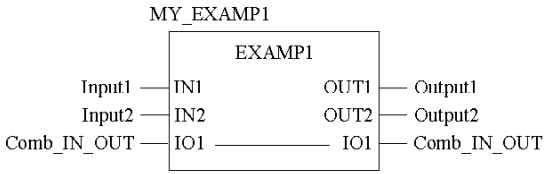
VAR_IN_OUT variable

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

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

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

Function block with VAR_IN_OUT variable in FBD:



Function block with VAR_IN_OUT variable in ST:

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

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

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

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

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

EN and ENO

Description

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

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

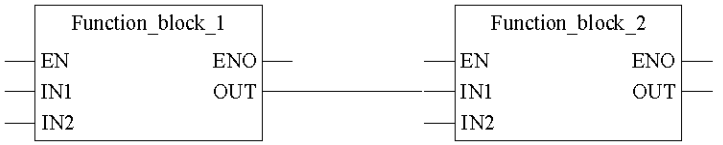
If the value of EN is equal to "1" when the FFB is invoked, the algorithms defined by the FFB will be executed. After the algorithms have been executed successfully, the value of ENO is set to "1". If certain error conditions are detected when executing these algorithms, ENO is set to "0".

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

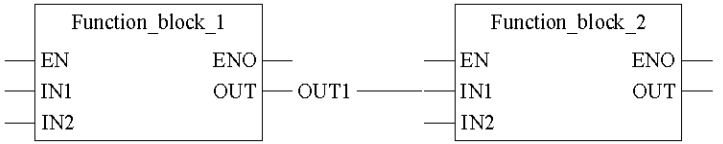
If the algorithms are executed successfully, then value of ENO is set to "1", else ENO is set to "0".

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

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



    If EN from FunctionBlock_1 is set to "0", the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle.
  - EN/ENO handling with function blocks that have one variable and one link as output parameters:



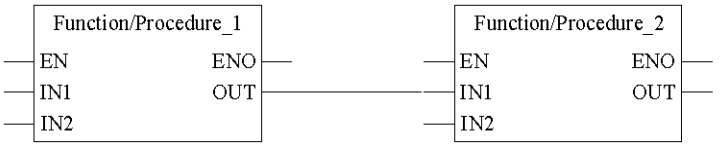
    If EN from FunctionBlock_1 is set to "0", the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle. The variable OUT1 on the same pin, either retains its previous status or can be changed externally without influencing the connection. The variable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

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

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

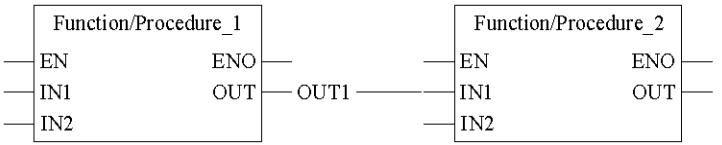
As defined in IEC61131-3, the outputs from deactivated functions (`EN`-input set to "0") is undefined. (The same applies to procedures.)
Here is an explanation of the output status in this case:

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



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

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



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

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

Conditional/Unconditional FFB Call

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

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

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

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

Note for IL and ST

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

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

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

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

Chapter 2
Block Availability on the Various Hardware Platforms

Block Availability on the Various Hardware Platforms

Introduction

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

NOTE: The functions and function blocks in this library are not defined in IEC 61131-3.

Diagnostics

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
ALRM_DIA	DFB	+	+	+	+	+
ASI_DIA	DFB	+	+	+	+	+
D_ACT	EFB	+	+	+	+	+
D_DYN	EFB	+	+	+	+	+
D_GRP	EFB	+	+	+	+	+
D_LOCK	EFB	+	+	+	+	+
D_PRE	EFB	+	+	+	+	+
D-REA	EFB	+	+	+	+	+
DEREG	EF	+	+	+	+	+
EV_DIA	DFB	+	+	+	+	+
MV_DIA	DFB	+	+	+	+	+
NEPO_DIA	DFB	+	+	+	+	+
ONLEVT	Procedure	+	+	+	+	+
REGDFB	Procedure	+	+	+	+	+
REGEXT	Procedure	+	+	+	+	+
REGIO	Procedure	+	+	+	+	+
SAFETY_MONITOR	DFB	+	+	+	+	+
SAFETY_MONITOR_V2	DFB	+	+	+	+	+
TEPO_DIA	DFB	+	+	+	+	+
UREGDFB	Procedure	+	+	+	+	+
USER_DIAG_ST_MODEL	DFB	+	+	+	+	+
Legend:						
+	Yes					
-	No					

Chapter 3
Diagnostics

Introduction

Two subjects are summarized under the subject diagnostics:

- 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.
- 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 Is in This Chapter?

This chapter contains the following topics:

System diagnostic

At a Glance

The system diagnostic is performed automatically. When the PLC detects a system error (for example, a watchdog being exceeded, an input/output error, division by zero, etc.), information is sent to the diagnostic viewer. The diagnostic viewer will display a system error message if you have checked the System diagnostics *(see Unity Pro, Operating Modes)* checkbox.

NOTE: As with the project diagnostic, the information viewed on the Viewer comes from the PLC's diag buffer *(see Unity Pro, Operating Modes)*, consequently the events are dated at the source and give the precise status of the process monitored.

Implementation

The following table describes the procedure to follow to use the system diagnostic on the Premium, Atrium and Quantum PLCs in Unity Pro.

Step	Action
1	Select **Tools** →**Project Settings** Result: the project settings configuration window is displayed.
2	In the **Diagnostics** area on the **Build** tab, check the **System diagnostics** *(see Unity Pro, Operating Modes)* checkbox.
3	Confirm with **OK**.
4	Build your project.
5	Transfer your project to the PLC.
6	Open the Diagnostic Viewer by selecting: **Tools** →**Diagnostic Viewer**. **Result**: the system diagnostic is operational and each system alarm now appears in the Viewer.

List of system alarms monitored

The following table summarizes the system information monitored automatically by the system diagnostic service.

System object	Succinct description of the alarm
%S10	Input/output error
%S11	Watchdog overflow!
%S15	Character string fault
%S18	Overflow or arithmetic error
%S19	Task period overrun
%S20	Index overflow
%S39	Saturation in event processing

System object	Succinct description of the alarm
%S51	Time loss in real time clock
%S65	Extract card command
%S66	Backup application to the memory card
%S67	State of the PCMCIA application memory card battery
%S68	State of processor battery
%S76	Diagnostic buffer configured
%S77	Diagnostic buffer full
%S96	Previously backup program
%S118	General Fipio I/O fault
%S119	General in-rack I/O fault
%SW0	Master task scanning period
%SW1	Fast task scanning period
%SW2	Period of auxiliary task scanning 0.
%SW3	Period of auxiliary task scanning 1.
%SW4	Period of auxiliary task scanning 2.
%SW4	Period of auxiliary task scanning 3.
%SW11	Watchdog duration
%SW17	Error status for floating operation
%SW76	Diagnostic function: save
%SW77	Diagnostic function: de-registration
%SW78	Diagnostic function: number of errors
%SW96	Save/restore %MW in flash memory.
%SW97	Storage card error code
%SW125	Type of blocking error
%SW146	Fipio bus arbiter function display
%SW153	List of Fipio channel manager faults
%SW154	List of Fipio channel manager faults

NOTE: The Fipio diagnostic is integrated in Unity Pro versions higher than 1.0.

Project diagnostic

At a Glance

The project diagnostic uses the diagnostic EFBs and DFBs and the diagnostic integrated in the SFC to generate alarms on the Diagnostic Viewer.

Each diagnostic EFB and each diagnostic DFB has its own specific operation described in the diagnostic library *(see page 31)*. If you cannot find the appropriate EFB or DFB among these elements, you can create your own diagnostic DFB *(see Unity Pro, Program Languages and Structure, Reference Manual)*.

NOTE: It is strongly recommended to only program a diagnostic DFB instance once within the application.

NOTE: As with the system diagnostic, the information viewed on the Viewer comes from the PLC's diag buffer *(see Unity Pro, Operating Modes)*, consequently the events are dated at the source and give the precise status of the process monitored.

Implementation of diagnostic EFBs or DFBs

The following table describes the procedure to follow to use the project with a diagnostic EFB or DFB on the Premium, Atrium and Quantum PLCs in Unity Pro.

Step	Action
1	Select **Tools →Project Settings** Result: the project settings configuration window is displayed.
2	In the **Diagnostics** area on the **Build** tab, check the **Application diagnostics** *(see Unity Pro, Operating Modes)* checkbox.
3	Choose the language of the messages in the Viewer.
4	Choose the **Application level** *(see Unity Pro, Operating Modes)* (alarm cause search level).
5	Confirm with **OK**.
6	Integrate the diagnostic EFBs or DFBs *(see page 31)* in your application. **Note**: The messages displayed in the Viewer will be the comments you will have associated with the instances of your diagnostic EFBs or DFBs.
7	Build your project.
8	Transfer your project to the PLC.
9	Open the Diagnostic Viewer by selecting: **Tools →Diagnostic Viewer**. **Result**: the system diagnostic is operational and each alarm generated by your EFBs or DFBs now appears in the Viewer.

Implementation of the SFC diagnostic

The following table describes the procedure to follow to use the SFC diagnostic on the Premium, Atrium and Quantum PLCs in Unity Pro.

Step	Action
1	Select **Tools →Project Settings** Result: the project settings configuration window is displayed.
2	In the **Diagnostics** area on the **Build** tab, check the **Application diagnostics** *(see Unity Pro, Operating Modes)* checkbox.
3	Confirm with **OK**.
4	Build your project.
5	Transfer your project to the PLC.
6	Open the Diagnostic Viewer by selecting: **Tools →Diagnostic Viewer**. **Result**: the SFC diagnostic is operational and each alarm linked to the SFC now appears in the Viewer.

Error message display

The number of messages it is possible to display is limited only by the size of the memory buffer.When there is not enough memory, a message warns the user and any messages of errors that have disappeared or have been acknowledged (if necessary) are then deleted.

It is possible to modify the color of the messages and the blinking associated with an acknowledged message.

In the viewer, it is possible to show only those messages which come from one or more specific zones.

The list of messages can be sorted according to each field. To do this, simply click on the column header containing the data on the basis of which the sort is to be carried out.

A second click carries out the sort in opposite order.

By default, the error messages are inserted into the list in the chronological order in which they appear.

Part II
Diagnostics

At a glance

This section describes the elementary functions and elementary function blocks of the `Diagnostics` family.

What Is in This Part?

This part contains the following chapters:

Chapter 4
ALRM_DIA: Interface with the diagnostics buffer

Subject of this Chapter

This chapter describes the ALRM_DIA DFB.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function Description

This DFB can be used to save any errors in a diagnostics buffer.

The switching of input COND1 to 0 or the switching of input COND0 to 1 causes an error to be saved in the diagnostics buffer.

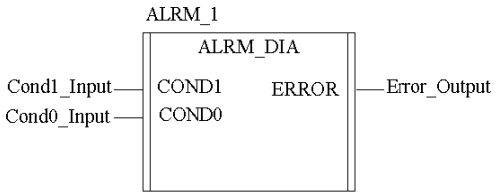
If both the COND1 and COND0 inputs are incorrect, one single error is saved. The error disappears when both the COND1 and COND0 inputs return to a correct value.

The additional parameters EN and ENO can be configured.

NOTE: This diagnostic DFB cannot be used in a user DFB.

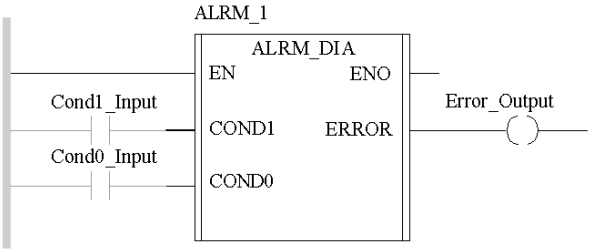
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL ALRM_1 (COND1:= Cond1_Input, COND0:= Cond0_Input,
ERROR => Error_Output)
```

Representation in ST

Representation:

```
ALRM_1 (COND1:= Cond1_Input, COND0: = Cond0_Input,
ERROR => Error_Output);
```

Description of the Parameters

The following table describes the input parameters:

Parameter	Type	Description
COND1	EBOOL	Input bit to be monitored in status 1. If the DFB is executed and this bit switches to 0, the DFB displays an error. If the input COND0 switches to 1, there is no new error. The default value is 1.
COND0	EBOOL	Input bit to be monitored in status 0. If the DFB is executed and this bit switches to 1, the DFB displays an error. If the input COND1 switches to 0, there is no new error. The default value is 0.

The following table describes the output parameters:

Parameter	Type	Description
ERROR	EBOOL	Error bit. This bit is set to 1 when an error appears. This bit is set to 0 if there is no longer an error.

Description of Variables

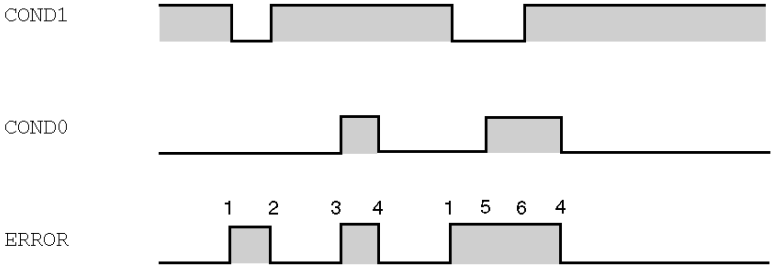
The following table describes the public variables:

Name	Type	Description
AREA_NR	INT	Area of automation system to be monitored. This word is used to specify which area of the automation system is to be monitored by the diagnostics DFB. Examples: ● Machining: No. 1 ● Milling: No. 2 ● Tapering: No. 3 AREA_NR must have the value 1, 2 or 3 in order for the user to identify the part of the automation system which is faulty. You are advised to make the above breakdown correspond to the functional module breakdown. AREA_NR can have a value between 0 and 15. The default value is 0.
OP_CTRL	EBOOL	Acknowledgement request. This bit indicates whether or not the operator needs to acknowledge the DFB instance: ● OP_CTRL = 0: the user does not have to provide acknowledgement, ● OP_CTRL = 1: the user has to provide acknowledgement. The default value is 0.

Detailed description of the operation of the ALRM_DIA function block

Timing diagram

The following timing diagram shows how the ALRM_DIA function block works.



Function

The following table describes the different phases illustrated by the above timing diagram:

Phase	Description
1	An error is detected when the input COND1 is set to 0.
2	The error is reset when the input COND1 is set to 1.
3	An error is detected when the input COND0 is set to 1.
4	The error is reset when the input COND0 is set to 0.
5	An error is not detected when the input COND0 is set to 1, as an error already exists.
6	The error is not reset when the input COND1 is set to 1, as the input COND0 remains at 1.

Chapter 5
D_ACT: Extended locking/action diagnostics

Introduction

This chapter describes the D_ACT block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The D_ACT 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 enables the action ACT. In this instance the trigger signal TRIGR must be active throughout the entire time. If the lock enable UNLOCK does not appear within the time period, an error situation occurs (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 message is terminated when the trigger signal TRIGR is inactive or the lock enable UNLOCK becomes active.

The REACT input enables the ACT output to be switched off or prevents its activation without a locking error being reported.

NOTE: Please ensure 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** are 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 set 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.

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 ED enable signal refers only to the activation of the diagnostics and has no effect on the ACT output of the locking diagnostics.

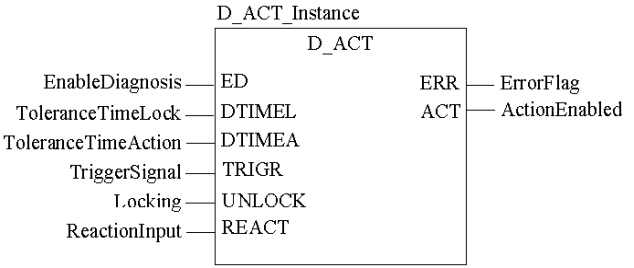
A positive edge of the ED enable signal (regardless at what time), or the locking signal UNLOCK becoming inactive while the TRIGR signal is active (in the action diagnostics phase), resets the function block and starts it in the locking diagnostics state.

An error will then always be reported when one of the internal timers reaches the values set for DTIMEL or DTIMEA.

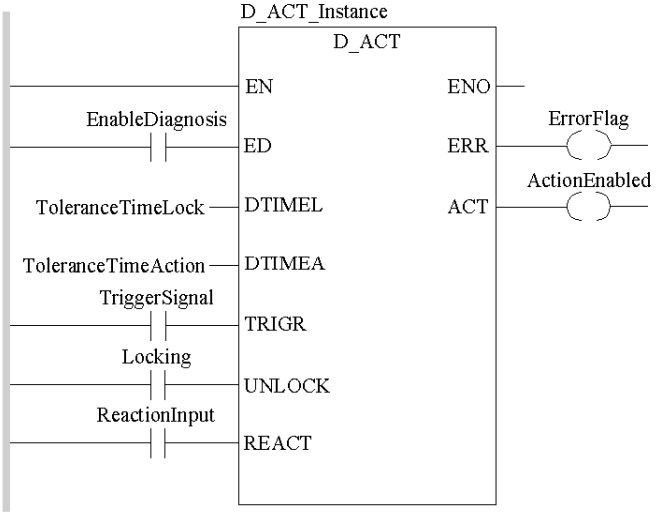
NOTE: Do not use diagnostic EFBs in DFBs.

EN and ENO can be configured as additional parameters.

Representation in FBD



Representation in LD



Representation in IL

```
CAL D_ACT_Instance (ED:=EnableDiagnosis,
    DTIMEL:=ToleranceTimeLock,
    DTIMEA:=ToleranceTimeAction, TRIGR:=TriggerSignal,
    UNLOCK:=Locking, REACT:=ReactionInput,
    ERR=>ErrorFlag, ACT=>ActionEnabled)
```

Representation in ST

```
D_ACT_Instance (ED:=EnableDiagnosis,
    DTIMEL:=ToleranceTimeLock,
    DTIMEA:=ToleranceTimeAction, TRIGR:=TriggerSignal,
    UNLOCK:=Locking, REACT:=ReactionInput,
    ERR=>ErrorFlag, ACT=>ActionEnabled) ;
```

Description of parameters

Description of the input parameters:

Parameter	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIMEL	TIME	Tolerance time for locking diagnostics
DTIMEA	TIME	Tolerance time for action diagnostics
TRIGR	BOOL	Trigger signal
UNLOCK	BOOL	Lock
REACT	BOOL	Reaction input

Description of the output parameters:

Parameter	Data type	Meaning
ERR	BOOL	Error message; 0: no error; 1: Error
ACT	BOOL	Action output

Public variables

Description of the public variables:

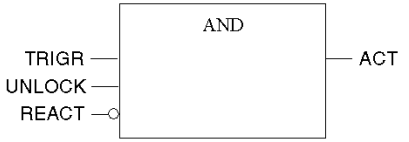
Parameter	Data type	Meaning
AREA_NR	BYTE	Automation area to be monitored. This byte specifies which area will be monitored by the diagnostic EFB. It is advisable to assign the numbers according to the functional modules. Values: 0 ... 15. The standard value is 0. Example: ● Cutting: No. 1 ● Milling: No. 2 ● Thread cutting: No. 3 In the example AREA_NR must have the value 1, 2 or 3, so that they can recognize the error affected area.
OP_CTRL	BOOL	This bit specifies whether a diagnostic event will request a user acknowledgement. ● 0: no user acknowledgement required ● 1: user acknowledgement required The standard value is 0.

Detailed description

Locking diagnostics parametering

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 timer will be started.

When the preset time at the DTIMEL input has expired, the ERR output will display an error that remains active until TRIGR becomes "0", ACT becomes "1", or diagnostics is 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 parametering

If the ACT output becomes "1" and REACT does not, the internal timer 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.

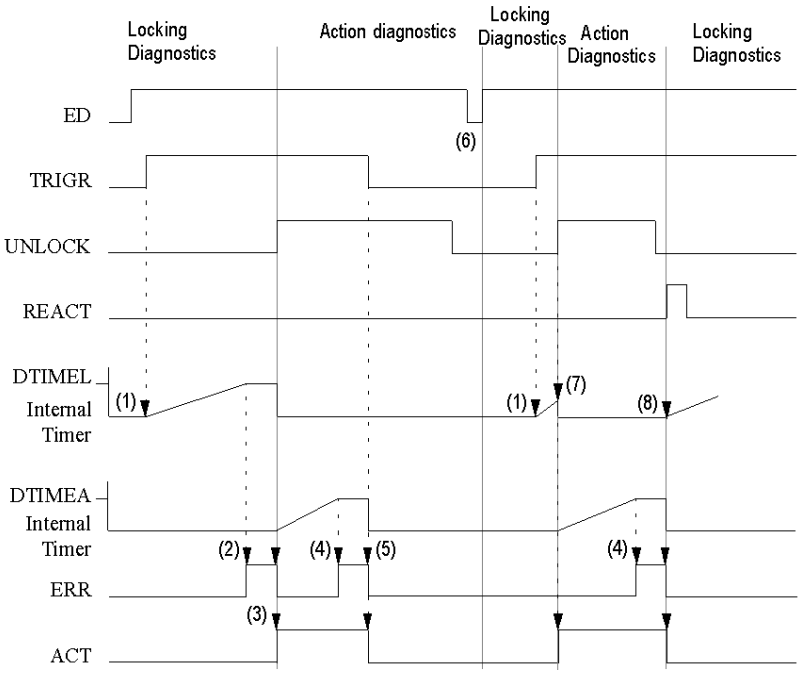
When the preset time at the DTIMEA input has expired, the ERR output will display an error that remains active until ACT becomes "0", REACT becomes "1", or diagnostics is deactivated.

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 timer starts when `TRIGR` is "1" and `UNLOCK` is "0".
(2) If the internal timer reaches the `DTIMEL` value, an error is reported.
(3) If `UNLOCK` becomes "1", the error will be cancelled, the internal timer is stopped and reset, and `ACT` becomes "1". Activating the action causes a switch to the action diagnostics. As the reaction has not yet occurred, the internal timer is started.
(4) If the internal timer reaches the `DTIMEA` value, an error is reported.
(5) If `TRIGR` becomes "0", the error will be cancelled, the internal timer is stopped and reset, and `ACT` becomes "0".
(6) When `ED` is "0", the function block is reset and is restarted in the locking diagnostics when `ED` is "1".
(7) When `UNLOCK` is "1", the internal timer is stopped and reset, and `ACT` becomes "1". Activating the action causes a switch to the action diagnostics. As the reaction has not yet occurred, the internal timer is started.
(8) If `REACT` becomes "1", the error will be cancelled, the internal timer is stopped and reset, and `ACT` becomes "0". It is switched again from the action diagnostics to the locking diagnostics.

Chapter 6
DEREG: Alarm de-registration

Description

Description of the function

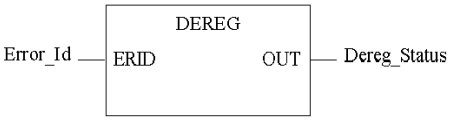
The DEREG function de-registers an alarm. This is entered into the code of a user diagnostics DFB and records the date when the error disappears in the diagnostics buffer.

NOTE: the alarm remains saved in the diagnostics buffer until the error is acknowledged (for errors that require acknowledgement) and read by all Viewers.

Additional parameters EN and ENO can be configured.

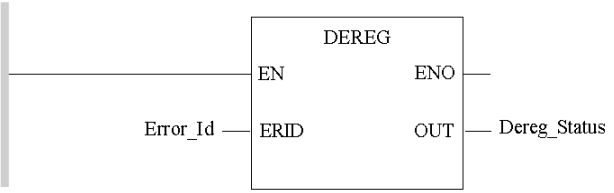
Representation in FBD

Representation applied to an integer:



Representation in LD

Representation applied to an integer:



Representation in IL

Representation applied to an integer:

```
LD Error_Id
DEREG
ST Dereg_Status
```

Representation in ST

Representation applied to an integer:

```
Dereg_Status: = DEREG(Error_Id);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Error_Id`	`INT`	Identifier of the recorded error.

The following table describes the output parameters:

Parameter	Type	Comment

`Dereg_Status`	`INT`	Error registration report.
		● if the de-registration is successful, `Dereg_Status` = 0
		● if the de-registration fails:
		  ● `Dereg_Status` = 1: diagnostics buffer not configured
		  ● `Dereg_Status` = 21: error identifier incorrect
		  ● `Dereg_Status` = 22: no error is registered with this identifier
		The system word `%SW77` is reserved for reception of the diagnostics DFB de-registration result (use not compulsory but recommended)

Chapter 7
D_DYN: Extended dynamic diagnostics

Introduction

This chapter describes the D_DYN block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The D_DYN function block is used for dynamic Diagnostics.

For certain processes it is necessary to combine D_LOCK (Extended locking diagnostics), D_ACT (Extended action diagnostics) and D_REA (Extended reaction diagnostics) in one unit, which monitors the current state of the diagnostics. This is only possible using a special function block, which internally manages the current diagnostics status.

To prevent this function block from becoming too complex, only one ED enable signal and one ERR error output have been defined.

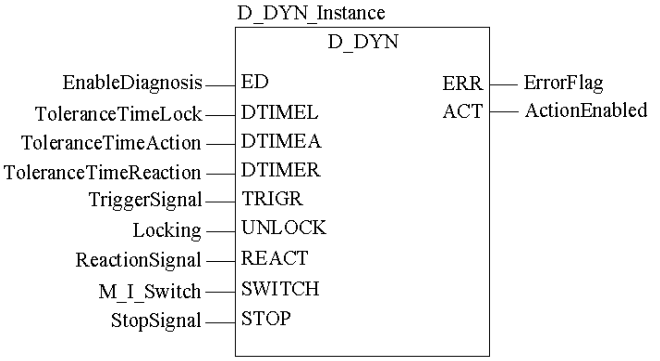
Monitoring is carried out cyclically. The activation of the diagnostics, and thereby the distribution of the cycle load, can be achieved through the enable signal ED.

NOTE: Do not use diagnostic EFBs in DFBs.

EN and ENO can be configured as additional parameters.

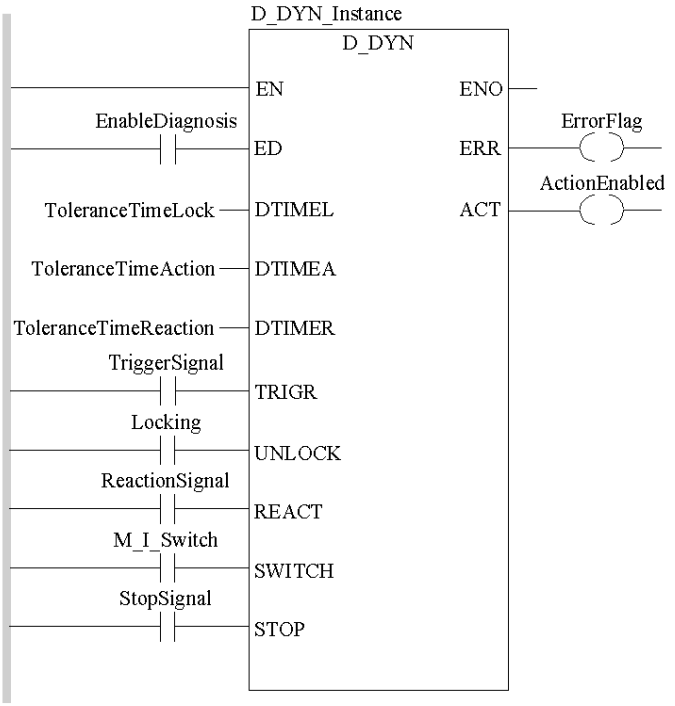
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL D_DYN_Instance (ED:=EnableDiagnosis,
    DTIMEL:=ToleranceTimeLock,
    DTIMEA:=ToleranceTimeAction,
    DTIMER:=ToleranceTimeReaction,
    TRIGR:=TriggerSignal, UNLOCK:=Locking,
    REACT:=ReactionInput, SWITCH:=M_I_Switch,
    STOP:=StopSignal, ERR=>ErrorFlag,
    ACT=>ActionEnabled)
```

Representation in ST

Representation:

```
D_DYN_Instance (ED:=EnableDiagnosis,
    DTIMEL:=ToleranceTimeLock,
    DTIMEA:=ToleranceTimeAction,
    DTIMER:=ToleranceTimeReaction,
    TRIGR:=TriggerSignal, UNLOCK:=Locking,
    REACT:=ReactionInput, SWITCH:=M_I_Switch,
    STOP:=StopSignal, ERR=>ErrorFlag,
    ACT=>ActionEnabled) ;
```

Description of parameters

Description of the input parameters:

Parameter	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIMEL	TIME	Tolerance time D_LOCK (locking diagnostics)
DTIMEA	TIME	Tolerance time D_ACT (action diagnostics)
DTIMER	TIME	Tolerance time D_REA (reaction diagnostics)
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

Description of the output parameters:

Parameter	Data type	Meaning
ERR	BOOL	Error message; 0: No error; 1: Error
ACT	BOOL	Action enabling

Public variables

Description of the public variables:

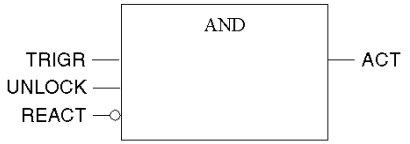
Parameter	Data type	Meaning
AREA_NR	BYTE	Automation area to be monitored. This byte specifies which area will be monitored by the diagnostic EFB. It is advisable to assign the numbers according to the functional modules. Values: 0 ... 15. The standard value is 0. Example: ● Cutting: No. 1 ● Milling: No. 2 ● Thread cutting: No. 3 In the example AREA_NR must have the value 1, 2 or 3, so that they can recognize the error affected area.
OP_CTRL	BOOL	This bit specifies whether a diagnostic event will request a user acknowledgement. ● 0: no user acknowledgement required ● 1: user acknowledgement required The standard value is 0.

Detailed description

Parametering

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:



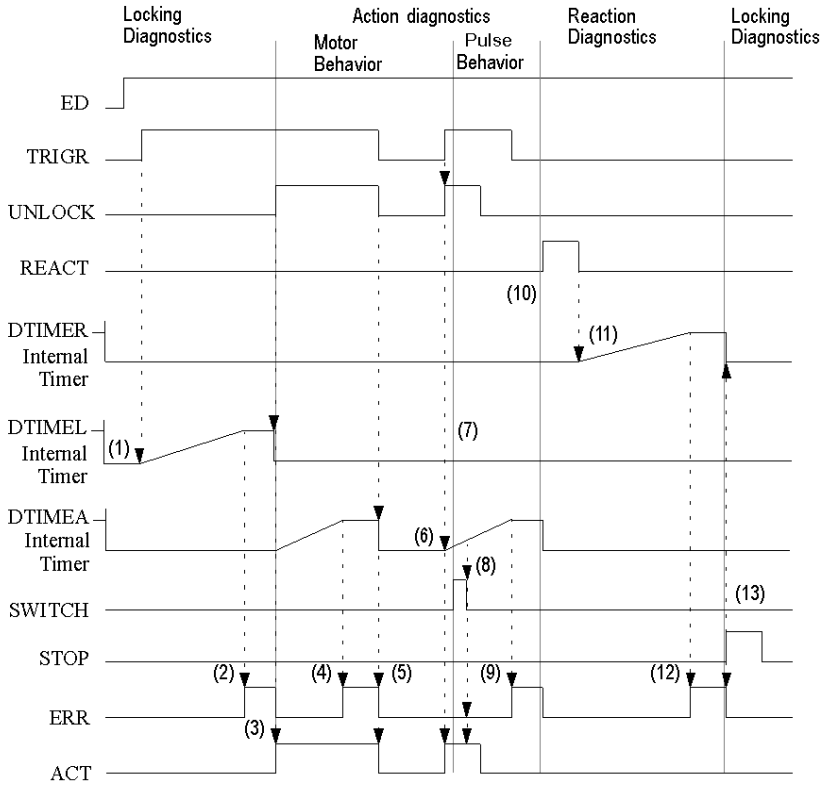
Parameterization for each diagnostics type can be found in the description for D_LOCK, D_ACT and D_REA.

An individual tolerance time (DTIMEL, DTIMEA, DTIMER) can be configured 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 timer starts when `TRIGR` is "1" and `UNLOCK` is "0".
(2) If the internal timer reaches the `DTIMEL` value, an error is reported.
(3) If `UNLOCK` becomes "1", the error will be cancelled, the internal timer is stopped and reset, and `ACT` becomes "1". Activating the action switches to the action diagnostics. As the reaction has not yet occurred, the internal timer is started.
(4) If the internal timer reaches the `DTIMEA` value, an error is reported.
(5) With M behavior, when `UNLOCK` becomes "0", `ACT` becomes "0", the error will be cancelled, and the internal timer is stopped and reset.
(6) When `TRIGR` and `UNLOCK` are "1" and `REACT` is "0", then `ACT` becomes "1" and the internal timer is started.
(7) If `SWITCH` becomes "1" and `ACT` is "1", a switch from M behavior to I behavior occurs.
(8) 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.
(9) If the internal timer reaches the `DTIMEA` value, an error is reported.

(10) If REACT becomes "1", the internal timer is stopped and reset. Activating the reaction switches to the reaction diagnostics.
(11) If REACT becomes "0", the internal timer is started.
(12) If the internal timer reaches the DTIMER value, an error is reported.
(13) If STOP becomes "1", the error will be cancelled and the internal timer is stopped and reset. An activation of the stop signal causes a switch back to locking diagnostics.

Chapter 8
D_GRP: Extended signal groups monitoring

Introduction

This chapter describes the D_GRP block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The D_GRP is used for signal group monitoring.

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.

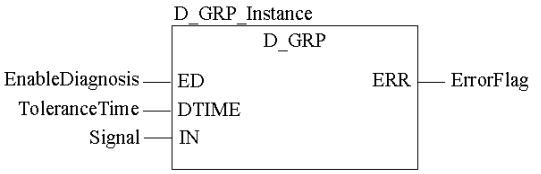
An error is reported when the status "1" is present at the signal input IN for longer than the tolerance time DTIME.

NOTE: Do not use diagnostic EFBs in DFBs.

EN and ENO can be configured as additional parameters.

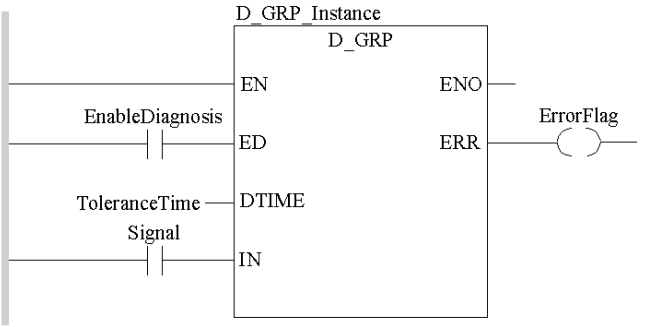
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL D_GRP_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, IN:=Signal, ERR=>ErrorFlag)
```

Representation in ST

Representation:

```
D_GRP_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, IN:=Signal, ERR=>ErrorFlag);
```

Description of parameters

Description of the input parameters:

Parameter	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
IN	BOOL	Signal

Description of the output parameters:

Parameter	Data type	Meaning
ERR	BOOL	Error message; 0: no error; 1:error

Public variables

Description of the public variables:

Parameter	Data type	Meaning
AREA_NR	BYTE	Automation area to be monitored. This byte specifies which area will be monitored by the diagnostic EFB. It is advisable to assign the numbers according to the functional modules. Values: 0 ... 15. The standard value is 0. Example: ● Cutting: No. 1 ● Milling: No. 2 ● Thread cutting: No. 3 In the example AREA_NR must have the value 1, 2 or 3, so that they can recognize the error affected area.
OP_CTRL	BOOL	This bit specifies whether a diagnostic event will request a user acknowledgement. ● 0: no user acknowledgement required ● 1: user acknowledgement required The standard value is 0.

Detailed description

Parametering

Deactivating the diagnostics or the setting the correct values at the inputs will reset the internal counter to "0".

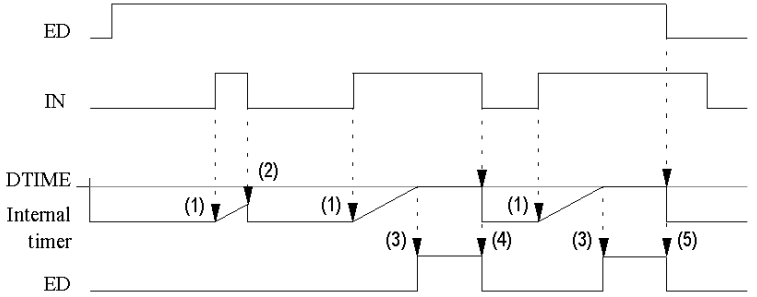
When IN is "1", the ERR output will display an error that remains active until IN becomes "0", or diagnostics is deactivated, after the time specified at the DTIME input has expired.

If a tolerance time DTIME of "0" is entered, an error message appears immediately if the IN input becomes "1".

An example for the process of signal group monitoring is given in the timing diagram

Timing diagram

Timing diagram signal group monitoring



(1) If IN is "1", the internal timer is started.
(2) If IN becomes "0", the internal timer is stopped and reset.
(3) If the internal timer reaches the DTIME value, an error is reported (ERR="1").
(4) If IN becomes "0", the error bit (ERR) is set to "0" and the internal timer is stopped and reset.
(5) When the enable signal ED is "0", (ERR) is set to "0" and the internal timer is stopped and reset.

Chapter 9
D_LOCK: Extended locking diagnostics

Introduction

This chapter describes the `D_LOCK` block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The D_LOCK function block is used for locking Diagnostics and to enable the action.

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 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. If the lock enable UNLOCK does not appear within the time period, an error situation occurs (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 message is terminated when the trigger signal TRIGR is inactive or the lock enable UNLOCK becomes active.

The D_LOCK function block contains a REACT input that enables the ACT output to be switched off or prevents its activation without a locking error being reported.

NOTE: Please ensure 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.

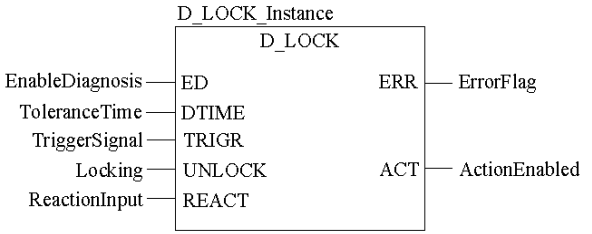
Monitoring is carried out cyclically. The activation of the diagnostics and thereby the distribution of the cycle load can be achieved through the enable signal ED. The ED enable signal refers only to the activation of the diagnostics and has no effect on the ACT output.

NOTE: Do not use diagnostic EFBs in DFBs.

EN and ENO can be configured as additional parameters.

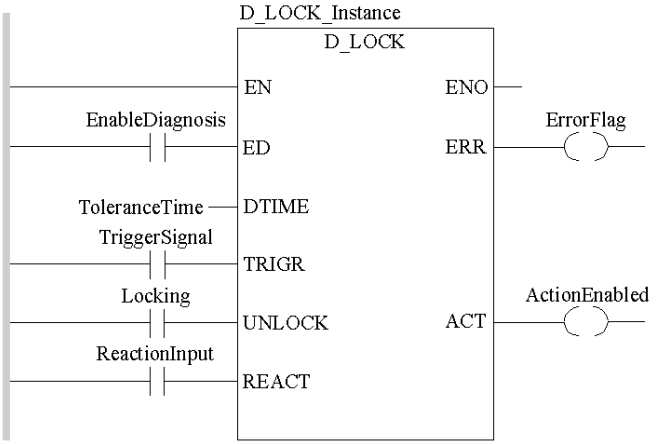
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL D_LOCK_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, TRIGR:=TriggerSignal,
    UNLOCK:=Locking, REACT:=ReactionInput,
    ERR=>ErrorFlag, ACT=>ActionEnabled)
```

Representation in ST

Representation:

```
D_LOCK_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, TRIGR:=TriggerSignal,
    UNLOCK:=Locking, REACT:=ReactionInput,
    ERR=>ErrorFlag, ACT=>ActionEnabled) ;
```

Description of parameters

Description of the input parameters:

Parameter	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
TRIGR	BOOL	Trigger signal
UNLOCK	BOOL	Lock
REACT	BOOL	Reaction input

Description of the output parameters:

Parameter	Data type	Meaning
ERR	BOOL	Error message; 0: no error; 1: Error
ACT	BOOL	Action output

Public variables

Description of the public variables:

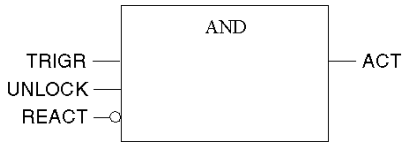
Parameter	Data type	Meaning
AREA_NR	BYTE	Automation area to be monitored. This byte specifies which area will be monitored by the diagnostic EFB. It is advisable to assign the numbers according to the functional modules. Values: 0 ... 15. The standard value is 0. Example: ● Cutting: No. 1 ● Milling: No. 2 ● Thread cutting: No. 3 In the example AREA_NR must have the value 1, 2 or 3, so that they can recognize the error affected area.
OP_CTRL	BOOL	This bit specifies whether a diagnostic event will request a user acknowledgement. ● 0: no user acknowledgement required ● 1: user acknowledgement required The standard value is 0.

Detailed description

Parametering

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 timer will be started.

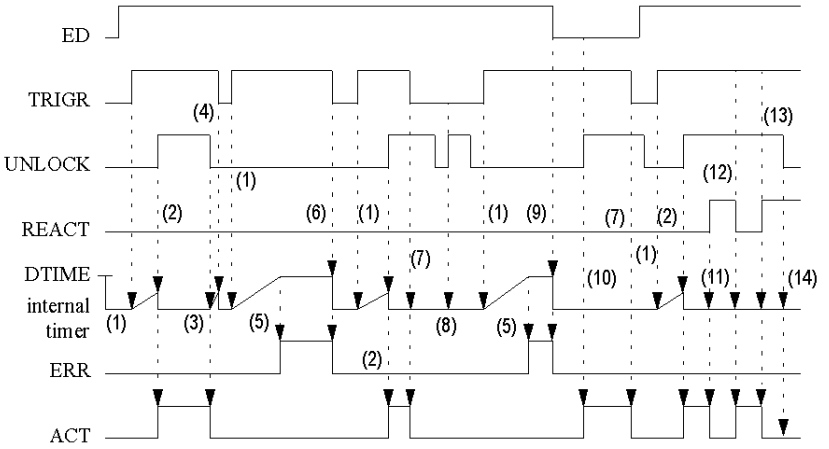
After the preset time at the DTIME input has expired, the ERR output displays an error. It remains active until either TRIGR becomes "0", or ACT becomes "1", or the diagnostic is deactivated.

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

Chapter 10
D_PRE: Extended process requirement monitoring

Introduction

This chapter describes the D_PRE block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The function block D_PRE is used for 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.

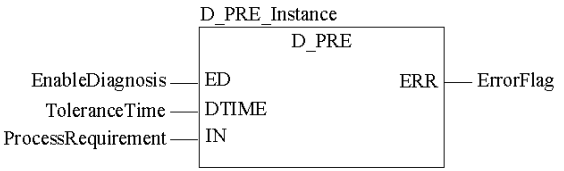
If more than one process requirement should be monitored simultaneously, the monitoring must be carried out with a pre-enabled AND block, whose output must be connected to the IN input of the D_PRE EFB.

NOTE: Do not use diagnostic EFBs in DFBs.

EN and ENO can be configured as additional parameters.

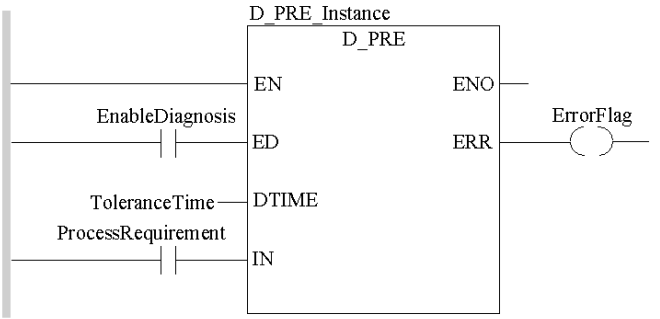
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL D_PRE_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, IN:=ProcessRequirement,
    ERR=>ErrorFlag)
```

Representation in ST

Representation:

```
D_PRE_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, IN:=ProcessRequirement,
    ERR=>ErrorFlag) ;
```

Description of parameters

Description of the input parameters:

Parameter	Data type	Meaning
ED	BOOL	Enable diagnostics
DTIME	TIME	Tolerance time
IN	BOOL	Process requirement

Description of the output parameters:

Parameter	Data type	Meaning
ERR	BOOL	Error message; 0: no error; 1: Error

Public variables

Description of the public variables:

Parameter	Data type	Meaning
AREA_NR	BYTE	Automation area to be monitored. This byte specifies which area will be monitored by the diagnostics EFB. It is advisable to assign the numbers according to the functional modules. Values: 0 ... 15. The standard value is 0. Example: ● Cutting: No. 1 ● Milling: No. 2 ● Thread cutting: No. 3 In the example AREA_NR must have the value 1, 2 or 3, so that they can recognize the error affected area.
OP_CTRL	BOOL	This bit specifies whether a diagnostic event will request a user acknowledgement. ● 0: no user acknowledgement required ● 1: user acknowledgement required The standard value is 0.

Detailed description

Parametering

If the signal connected to IN becomes "0" and the diagnostics is active, the internal counter will be started.

The deactivation of the diagnostics or of the attachment of the correct input value stops the timer (the requirements may contain errors during the tolerance time DTIME) and sets the timer back to "0".

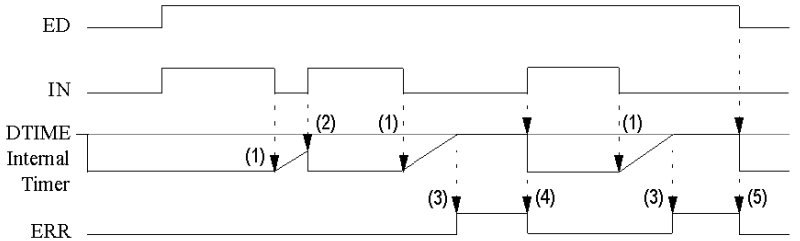
When the default time at the DTIME input has expired, the ERR output displays an error that remains active until the requirements are "1" or the diagnostic is deactivated.

If the tolerance time DTIME is entered as "0", an error message is immediately returned when the static conditional values (IN) become "0".

An example for process requirement monitoring is given in timing diagram.

Timing diagram

Monitoring of process requirements timing diagram



(1) If IN is "0", the internal timer is started.
(2) If IN is "1", the internal timer is stopped and reset.
(3) If the internal timer reaches the DTIME value, an error is reported (ERR becomes "1").
(4) If IN is "1", the error will be cancelled and the internal timer is stopped and reset.
(5) When the enable signal ED is "0", the error is cancelled (ERR becomes "0") and the internal timer is stopped and reset.

Chapter 11
D_REA: Extended reaction diagnostics

Introduction

This chapter describes the D_REA block.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The D_REA function block is used for reaction Diagnostics.

Once the expected reaction has occurred in the Action 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.

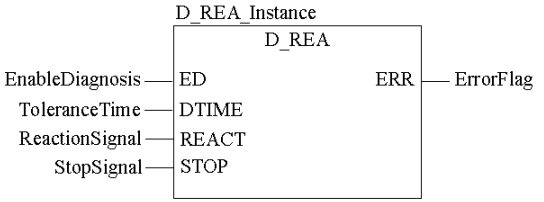
The stop condition terminates reaction diagnostics.

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.

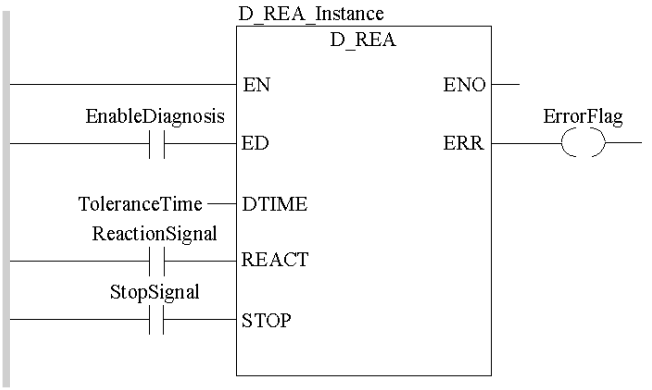
NOTE: Do not use diagnostic EFBs in DFBs.

EN and ENO can be configured as additional parameters.

Representation in FBD



Representation in LD



Representation in IL

```
CAL D_REA_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, REACT:=ReactionSignal,
    STOP:=StopSignal, ERR=>ErrorFlag)
```

Representation in ST

```
D_REA_Instance (ED:=EnableDiagnosis,
    DTIME:=ToleranceTime, REACT:=ReactionSignal,
    STOP:=StopSignal, ERR=>ErrorFlag) ;
```

Description of parameters

Description of the input parameters:

Parameter	Data type	Meaning
ED	BOOL	Enable diagnostic
DTIME	TIME	Tolerance time
REACT	BOOL	Reaction signal
STOP	BOOL	Stop signal

Description of the output parameters:

Parameter	Data type	Meaning
ERR	BOOL	Error message; 0: no error; 1: Error

Public variables

Description of the public variables:

Parameter	Data type	Meaning
AREA_NR	BYTE	Automation area to be monitored. This byte specifies which area will be monitored by the diagnostic EFB. It is advisable to assign the numbers according to the functional modules. Values: 0 ... 15. The standard value is 0. Example: ● Cutting: No. 1 ● Milling: No. 2 ● Thread cutting: No. 3 In the example AREA_NR must have the value 1, 2 or 3, so that they can recognize the error affected area.
OP_CTRL	BOOL	This bit specifies whether a diagnostic event will request a user acknowledgement. ● 0: no user acknowledgement required ● 1: user acknowledgement required The standard value is 0.

Detailed description

Parametering

If the REACT input is "0", the internal counter will be started.

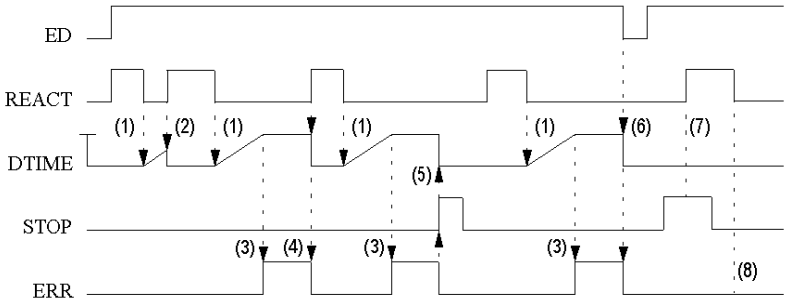
When the preset time at the DTIME input has expired, the ERR output will display an error that remains active until REACT becomes "1", STOP becomes "1", or diagnostics is deactivated.

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 processing of reaction diagnostics.

Timing diagram

Reaction diagnostics timing diagram



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

Chapter 12
EV_DIA : Event monitoring DFB

Object of this chapter

This chapter describes the EV_DIADFB.

What Is in This Chapter?

This chapter contains the following topics:

Description

Description of the Function

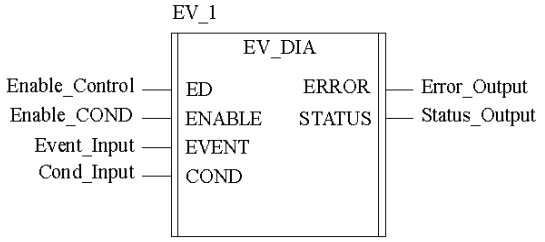
The DFB EV_DIA allows you to monitor the 2-bit state without taking account of any time correlation.

Additional parameters EN and ENO can be configured.

NOTE: This diagnostic DFB cannot be used in a user DFB.

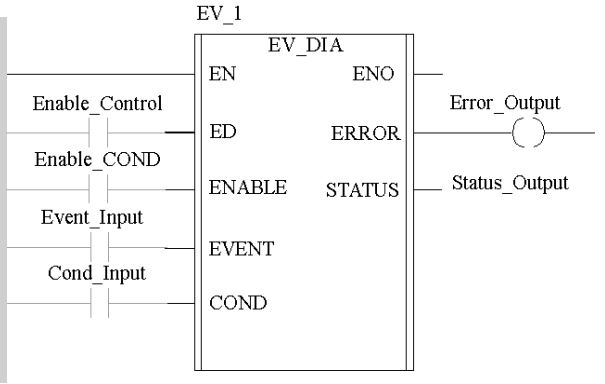
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL EV_1 (ED := Enable_Control, ENABLE := Enable_COND, EVENT :=
Event_Input, COND := Cond_Input, ERROR => Error_Output, STATUS =>
Status_Output)
```

Representation in ST

Representation:

```
EV_1 (ED := Enable_Control, ENABLE := Enable_COND, EVENT := Event_Input,
COND := Cond_Input, ERROR => Error_Output, STATUS => Status_Output);
```

Description of Parameters

The following table describes the input parameters:

Parameter	Type	Description
ED	EBOOL	DFB activation bit. If ED = 0, the EVENT and COND input are not monitored. The default value is 0.
ENABLE	EBOOL	Monitor enable bit If ENABLE = 0, only COND input is monitored. If ENABLE = 1, the COND and EVENT inputs are monitored. The default value is 0.
EVENT	EBOOL	Input bit to be monitored. If the DFB is executed and ENABLE = 1, the DFB verifies that the EVENT input: ● possesses the value specified by the VALUE public variable, ● is stable (no switching between 1, 0, 1 state). If this is not the case, the DFB will signal that there is a fault. If ENABLE = 0, theEVENT input is not monitored. The default value is 0.
COND	EBOOL	Input bit to be monitored. The input bit to be monitored is set at 1, whatever the status of the ENABLE input. If the DFB is executed and this bit changes to 0, the DFB will signal that there is a fault. The default value is 1.

The following table describes the output parameters:

Parameter	Type	Description
ERROR	EBOOL	Fault bit This bit is set to 1 as soon as a fault appears. This bit is set to 0 if the ED input returns to 0 or if there is no longer an error.
STATUS	INT	Fault type. The following bits indicate the type of fault detected : ● bit 0 =1 : EVENT is different from the VALUE specified ● bit 1 =1: COND does not possess the expected value of 1 ● bit 8 =1: EVENT is unstable This word is set at 0 if there is no fault. This word is set at 0 if the ED input returns to 0 or if there is no longer an error.

Description of variables

The following table describes the public variables:

Name	Type	Description
VALUE	EBOOL	Comparison value. Value (0 or 1) to which the EVENT input is compared. This variable can be modified by the program, its default value is 1.
AREA_NR	INT	Automatic operation zone to be monitored. This word is used to specify which automatic operations zone is being monitored by the diagnostics DFB. Examples: ● Manufacture : n° 1 ● Reaming : n° 2 ● Threading: n° 3 AREA_ NR must have a value of 1, 2 or 3 for the user to identify which section of the automatic operation has a fault. It is advisable to make the above division correspond with the division in the function module. AREA_ NR can take on a value between 0 and 15. The default value is 0.
OP_CTRL	EBOOL	Acknowledge request This bit signals whether or not a DFB instance must be acknowledged by the operator : ● OP_CTRL = 0 : not acknowledged by the operator, ● OP_CTRL = 1 : acknowledged by the operator, The default value is 0.

Detailed description of the operation of the EV_DIA function block

Introduction

As soon as one of the inputs monitored is no longer parameterized within the DFB, the latter signals a fault while updating these outputs:

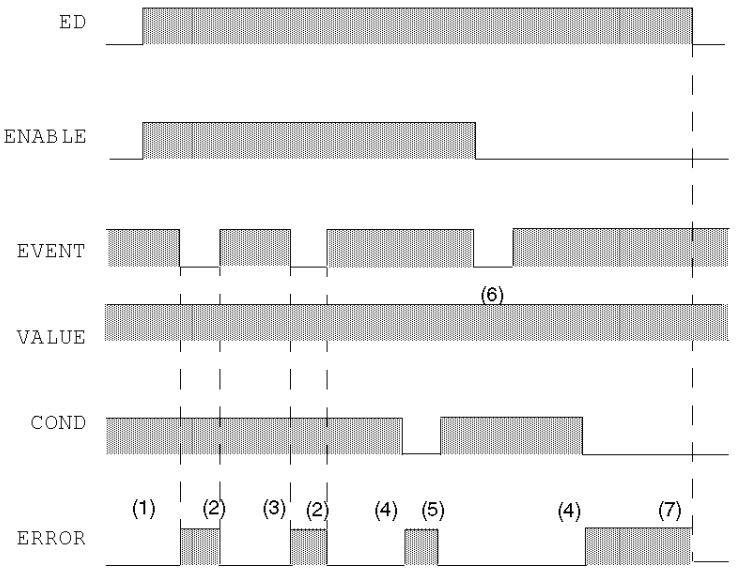
- setting the ERROR bit to 1,
- setting to 1 of the STATUS word bit that corresponds to the fault.

Any faults detected during a single monitoring cycle are picked up as they appear (STATUS word bit is set to 1, corresponding with output update).

At the end of a monitoring cycle (falling Edge ED input), the ERROR and STATUS outputs are reinitialized to 0.

Trend diagram

The following diagram shows how the EV_DIA function block works .



Operation

The following table describes the different phases illustrated by the diagram below:

Phase	Description
1	When the EVENT input is different from the VALUE public variable (ENABLE = 1), a fault is detected.
2	The ERROR output changes to 0 when the EVENT input takes on the value VALUE of the public variable.
3	A fault is detected when the EVENT input becomes unstable. This type of fault appears after the status of EVENT input changes twice in the same monitoring cycle. The EVENT input unstable fault (bit 8 of word STATUS goes to 1) , becomes a EVENT different from VALUE fault (bit 1 of word STATUS goes to 1) if there are more than 1000 PLC cycles before a new fault is detected. The EVENT input unstable fault disappears if there are more than 1000 PLC cycles, and also if the EVENT input is always equal to the value specified by VALUE.
4	A fault is detected when COND input is anything other than 1.
5	ERROR output changes to 0 when COND input takes on the value of 1.
6	EVENT input is different from the VALUE public variable: there is no fault as ENABLE input = 0.
7	ERROR output changes to 0 when ED input takes on the value of 0.

DFB operation during power outage

During a cold restart, the DFB initializes the parameters and public variables:

- COND input set at 1, and other inputs set at 0,
- outputs set at 0,
- VALUE set at 1.

Example of using and programming the EV_DIA function block

Application description

This example describes the monitoring of filling a hopper

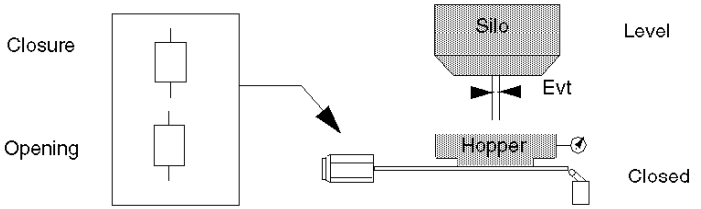
Cycle : pour 100kg of the product into the hopper.

Checks to be carried out

- check that the hopper is closed during filling,
- always check that the silo is not empty.

Illustration of the application

The drawing below shows the application and the checks that have been carried out



Program in ST language

The application is programmed in text in this example.

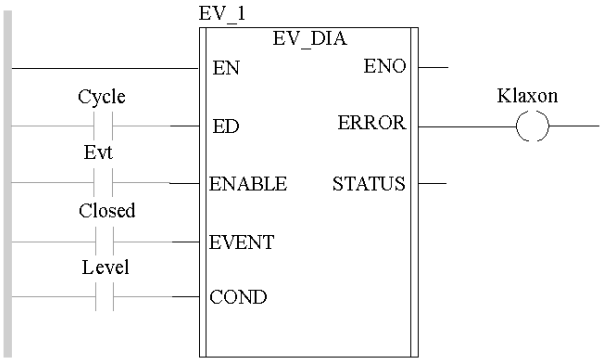
```
%L0:
EV_1  (ED := Cycle, ENABLE := Evt, EVENT := Closed, COND := Level, ERROR
=> Klaxon);
!IF (Cycle AND Closed)
THEN
 SET (Evt);
ELSE
 RESET (Evt);
END_IF;
(*Hopper trap door Command*)
!IF Weight >= 100
THEN
 RESET (Evt);
 RESET (Closure);
 SET (Opening);
END_IF;
!IF Weight =0
THEN
 RESET (Opening);
 SET (Closure);
END_IF;
```

The level in the silo is always checked, as long as the cycle is running.

When the hopper fills `Evt` on `ENABLE`, the hopper trap-door is monitored in its Closed state (`EVENT` input).

DFB graphic presentation

The illustration below shows a graphic representation of the DFB diagnostics as it is hardwired in this example.



Chapter 13
MV_DIA : Motion monitoring DFB

Object of this chapter

This chapter describes the `MV_DIA` DFB.

What Is in This Chapter?

This chapter contains the following topics:

Description

Description of the Function

The MV_DIA DFB can monitor:

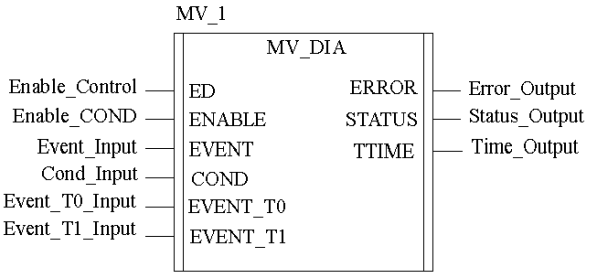
- the state of a bit without time constraints,
- a motion (change in bit state within a defined time interval).

Additional parameters EN and ENO can be configured.

NOTE: This diagnostic DFB cannot be used in a user DFB.

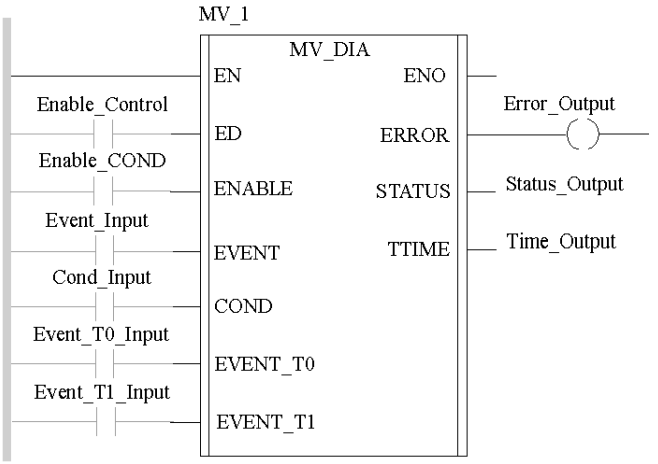
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL MV_1 (ED := Enable_Control, ENABLE := Enable_COND, EVENT :=
Event_Input, COND := Cond_Input, EVENT_T0 := Event_T0_Input, EVENT_T1 :=
Event_T1_Input, ERROR => Error_Output, STATUS => Status_Output, TTIME =>
Time_Output)
```

Representation in ST

Representation:

```
MV_1 (ED := Enable_Control, ENABLE := Enable_COND, EVENT := Event_Input,
COND := Cond_Input, EVENT_T0 := Event_T0_Input, EVENT_T1 :=
Event_T1_Input, ERROR => Error_Output, STATUS => Status_Output, TTIME =>
Time_Output);
```

Description of Parameters

The following table describes the input parameters:

Parameter	Type	Description
ED	EBOOL	DFB activation bit. If ED = 0, the EVENT,EVENT_T0, EVENT_T1 and COND inputs are not monitored. The default value is 0.
ENABLE	EBOOL	Monitor enable bit If ENABLE = 0, only COND input is monitored. If ENABLE = 1, the CONDand EVENT_T0, EVENT_T1 are monitored. The default value is 0.
EVENT	EBOOL	Input bit to be monitored. If the DFB is executed and ENABLE = 1, the DFB verifies that the EVENT input: ● possesses the value specified by the VALUE public variable, ● is stable (no switching between 1, 0, 1 state). ● possesses the value specified by the VALUE public variable, a MMIN minimum time and a MMAX maximum time. If this is not the case, the DFB will signal that there is a fault. If ENABLE = 0, theEVENT input is not monitored. The default value is 0.
COND	EBOOL	Input bit to be monitored. The input bit to be monitored is set at 1, whatever the status of the ENABLE input. If the DFB is executed and this bit changes to 0, the DFB will signal that there is a fault. The default value is 1.
EVENT_T0	EBOOL	Exterior event associated with T0 time. This (optional) parameter is a bit which must change from 0 to 1 before T0 time or in ENABLE = 1 format. The default value is 1.
EVENT_T1	EBOOL	Exterior event associated with T1 time. This (optional) parameter is a bit which must change from 0 to 1 before T1 time or in ENABLE = 1 format. The default value is 1.

The following table describes the output parameters:

Parameter	Type	Description
ERROR	EBOOL	Fault bit This bit is set to 1 as soon as a fault appears. This bit is set to 0 if the ED input returns to 0 or if there is no longer an error.
STATUS	INT	Fault type. The following bits indicate the type of fault detected : ● bit 0 =1: EVENT is different from the VALUE specified, ● bit 1 =1: COND does not possess the expected value of 1, ● bit 2=1: EVENT does not possess the VALUE value during the MIN period requested, ● bit 3 =1: EVENT possesses the value VALUE beyond the MAX time requested, ● bit 4 =1: EVENT_T0 not seen at 1 before the T0 time requested, ● bit 5 =1: EVENT_T1 not seen at 1 before the time T1 requested, ● bit 6 =1: EVENT_T0 not seen at 1 during range ENABLE=1, ● bit 7 =1: EVENT_T1 not seen at 1 during range ENABLE=1, ● bit 8 =1: EVENT is unstable, ● bit 9 =1: EVENT_T0 falls back to 0 after T0 time, ● bit 10 =1: EVENT_T1 falls back to 0 after T1 time, ● bit 14 =1: fault through the overrunning of the internal clock. This word is set at 0 if there is no fault. This word is set at 0 if the ED input returns to 0 or if there is no longer an error.
TTIME	INT	Current time. Word indicating the current time with a time base expressed in multiples of N x 100 ms The N coefficient is defined by the BASE public variable. TTIME is initialized at the PPRESET value, and starts to change on the rising edge of the ENABLE input. It sets to 0, on the ENABLE falling edge. If a fault is detected (caused by EVENT input), TTIME does not remain frozen: ● if ENABLE = 0, TTIME = 0 ● if ENABLE = 1, TTIME = internal running time

Detailed description of public variables

General public variables

The following table describes the general public variables:

Name	Type	Description
VALUE	EBOOL	Comparison value. Value (0 or 1) to which the EVENT input is compared. This variable can be modified by the program, its default value is 1.
PPRESET	INT	Current time initialization value. This word is used to define either by program or by variable modification the current time initialization value (TTIME) on the ENABLE rising edge. This variable can be modified by the program, its default value is 0.
BASE	INT	Base time value. This word defines the N coefficient required for defining the time base. All times are expressed in multiples of N x 100 ms. The default value is 1.
AREA_NR	INT	Automatic operation zone to be monitored. This word is used to specify which automatic operations zone is being monitored by the diagnostics DFB. Examples: ● Manufacture: n° 1. ● Reaming: n° 2. ● Threading: n° 3. AREA_ NR must have a value of 1, 2 or 3 for the user to identify which section of the automatic operation has a fault. It is advisable to make the above division correspond with the division in the function module. AREA_ NR can take on a value between 0 and 15. The default value is 0.
OP_CTRL	EBOOL	Acknowledge request. This bit signals whether or not a DFB instance must be acknowledged by the operator: ● OP_CTRL = 0: not acknowledged by the operator, ● OP_CTRL = 1: acknowledged by the operator, The default value is 0.

Public variables associated with EVENT input

The following table describes the public variables associated with MV_DIA DFB EVENT input:

Name	Type	Description
MMIN	INT	Minimum time. This word defines the minimum amount of time during which EVENT input must be equal to the VALUE internal data. As soon as EVENT input does not match VALUE before MMIN time has passed, the DFB indicates that there is a fault. If this is the first fault on EVENT input since the last initialization (ENABLE 0> 1), the time at which the fault occurred (MMIN) is recorded by DEFTIME. This variable can be modified by the program, its default value is 0.
MMAX	INT	Maximum time. This word defines the maximum amount of time during which EVENT input must be equal to the VALUE internal data. If the EVENT input is equal to VALUE after the MMAX time, then the DFB indicates that there is a fault. If this is the first fault on EVENT input since the last initialization (ENABLE 0> 1), the time at which the fault occurred (MMAX) is recorded by DEFTIME. This variable can be modified by the program, its default value is 0.
DEFTIME	INT	Recording the time of the first diagnostic message. This word records the time at which the first diagnostic message on EVENT input occurs. DEFTIME starts at 0 on the falling/rising edge of EVENT input depending on the condition (0 or 1) of the VALUE variable. The DEFTIME variable **cannot** be modified by program, its default value is **0**.
MIN_VAL	INT	Recording the minimum time. This word records the minimum time during which EVENT input possessed the value specified by VALUE data. MIN_VAL is reset at 32767 on input rising edge. This variable can be modified by the program, its default value is 32767.
MAX_VAL	INT	Recording maximum time. This word records the maximum time during which EVENT input possessed the value specified by VALUE data. MAX_VAL is reset to 0 on the ED input rising edge. This variable can be modified by the program, its default value is 0.
INI_MIN	INT	MMIN initial value. This word indicates the initial value of MMIN time. This value is transferred into MMIN on start or cold restart. The default value is 0.
INI_MAX	INT	MMAX initial value. This word indicates the initial value of MMAX time. This value is transferred into MMAX on start or cold restart. The default value is 0.

Public variables associated with EVENT_T0 and T1 input

The following table describes the public variables associated with MV_DIA DFB EVENT_Ti (with I = 0 or 1) input:

Name	Type	Description
Ti	INT	Minimum time. This word defines the Ti maximum time for EVENT_Ti input to change from 0 to 1. If this change happens after Ti, the DFB indicates that there is a fault. This variable can be modified by the program, its default value is 0.
MIN_EVTi	INT	Recording the minimum time. This word records the minimum time which was required for EVENT_Ti input to change from 0 to 1. MIN_EVTi is initialized at 32767 on the rising edge of the ED input. This variable can be modified by the program, its default value is 32767.
MAX_EVTi	INT	Recording maximum time. This word records the maximum time which was required for EVENT_Ti input to change from 0 to 1. MAX_EVTi is initialized at 0 on the rising edge of the ED input. This variable can be modified by the program, its default value is 0.
INIT_Ti	INT	Initial value of Ti time. This word indicates the initial value of Ti time. This value is transferred to Ti data on start or cold restart. MIN_VAL is reset at 32767 on input rising edge. The default value is 0.

Detailed description of the operation of the MV_DIA function block

Introduction

As soon as one of the inputs monitored is no longer parameterized within the DFB, the DFB signals that there is a fault while updating output.

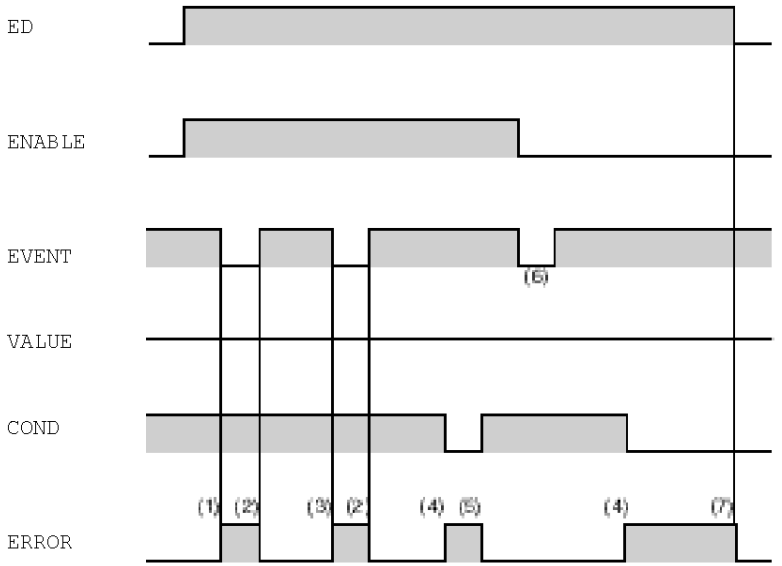
- setting the ERROR bit to 1,
- setting to 1 of the STATUS word bit that corresponds to the fault.

Any faults detected during a single monitoring cycle are picked up as they appear (STATUS word bit is set to 1, corresponding with output update).

At the end of a monitoring cycle (falling Edge ED input), the ERROR and STATUS outputs are reinitialized to 0.

Trend diagram

The following chart shows how the MV_DIA function block works .



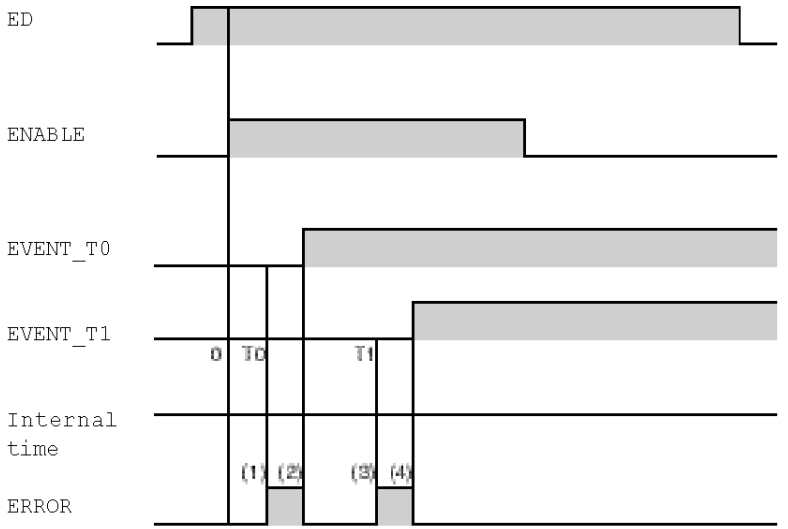
Operation

The following table describes the different phases illustrated by the chart below:

Phase	Description
1	When the EVENT input is different from the VALUE public variable (ENABLE = 1), a fault is detected.
2	The ERROR output changes to 0 when the EVENT input takes on the values of the VALUE public variable.
3	A fault is detected when the EVENT input becomes unstable. This type of fault appears after the status of EVENT input changes twice in the same monitoring cycle. The EVENT input unstable fault (bit 8 of Status word goes to 1) , becomes an EVENT different from VALUE fault (bit 0 of Status word goes to 1) if there are more than 1000 PLC cycles before a new fault is detected. The EVENT input unstable fault disappears if there are more than 1000 PLC cycles, and also if the EVENT input is always equal to the value specified by VALUE.
4	A fault is detected when COND input is anything other than 1.
5	ERROR output changes to 0 when COND input takes on the value of 1.
6	EVENT input is different from the VALUE public variable: there is no fault as ENABLE input = 0.
7	ERROR output changes to 0 when ED input takes on the value of 0.

Illustration of DFB operation, EVENT_T0 and EVENT_T1 inputs

The following chart shows how the MV_DIA function block works .



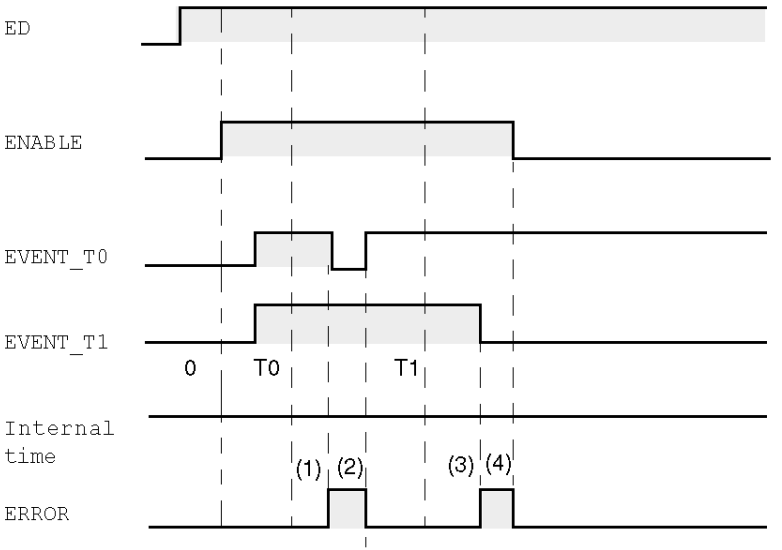
Description of operation for EVENT_T0 and EVENT_T1

The following table describes the different phases illustrated by the chart below:

Phase	Description
1	A fault is detected when the EVENT_T0 input does not change to 1 during T0 time.
2	ERROR output changes to 0 when EVENT_T0 input takes on the value of 1.
3	A fault is detected when EVENT_T1 input has not changed to 1 during T1 time.
4	ERROR output changes to 0 when EVENT_T1 input takes on the value of 1.

Illustration of DFB operation, EVENT_T0 and EVENT_T1 inputs

The following chart shows how the MV_DIA function block works .



Description of operation for EVENT_T0 and EVENT_T1 inputs

The following table describes the different phases illustrated by the chart below:

Phase	Description
1	A fault is detected when EVENT_T0 input has not remained at 1 after T0 time.
2	ERROR output changes to 0 when EVENT_T0 input takes on the value of 1.
3	A fault is detected when EVENT_T1 input has not remained at 1 after T1 time.
4	ERROR output changes to 0 when ENABLE input changes to 0.

Time base

The time base for counting T0 , T1 , MMIN and MMAX actual time is defined by BASE . A change in the BASE value is not taken into account during the monitoring cycle which is running at the time. It will be taken into account at the start of the next cycle.

DFB operation during power outage

During a cold restart, the DFB initializes the parameters and public variables :

- COND, EVENT_T0 and EVENT_T1 inputs are set at 1
- other inputs (ENABLE, EVENT) are set at 0
- ERROR, STATUS and TTIME outputs are set at 0,
- VALUE is set at 1
- INI_T0, INI_T1, INI_MIN and INI_MAX are transferred respectively to T0, T1, MMIN and MMAX,
- other data (PPRESET, DEFTIME, MAX_EVT0, MAX_EVT1 and MAX_VAL) are set at 0.

Example of using and programming the MV_DIA function block

Application description

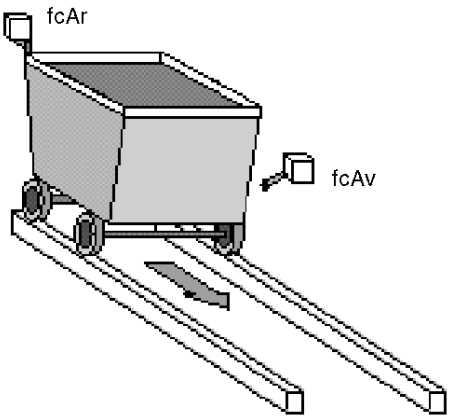
This example describes monitoring the movement of a cart.

Checks to be carried out:

- check that the `Forward` order has been correctly entered,
- after receiving the command `Forward`, ensure that the cart leaves the `fcAr` sensor before 1 second has passed,
- check that the `Forward` run-time period does not go beyond 10 seconds,
- check that the 2 sensors at the end of the run are never on 1 at the same time,
- check that the `fcAr` sensor is on 1 while the cart is at a halt.

Illustration of the application

The drawing below shows the application and the checks that have been carried out:



Program in ST language

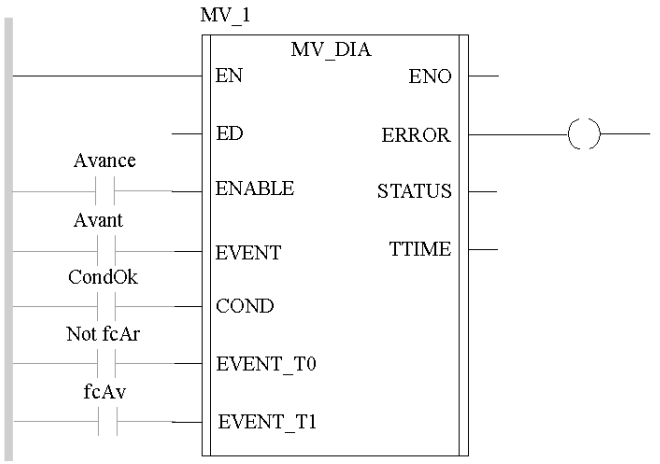
The application is programmed in text in this example.

```
%L0:
Advance := Forward AND NOT fcAv;
CondOK := Not (fcAv AND fcAr) AND (fcAr OR Advance OR fcAv)
NfcAr := Not fcAr;
MV_DIA1 (Advance, Forward, CondOK, NfcAr, fcAv, , ,) ;
```

- `EVENT` input is used to check that the Forward order has been correctly entered while the cart moves.
- `EVENT_T0` input is used to ensure that the card leaves the fcAr sensor before 1 second,
- `EVENT_T1` input checks that the run-time does not last longer than 10 seconds,
- `COND` input is monitored on 1 for the entire time that the DFB is running. It is used to check that:
  - the `fcAr` sensor is on 1 while the is at a stop,
  - the two `fcAr` and `fcAv` sensors are never on 1 at the same time.

DFB graphic presentation

The illustration below shows a graphic representation of the DFB diagnostics as it is hardwired in this example.



Chapter 14
NEPO_DIA, TEPO_DIA : Command and diagnostics of the operating section DFB

Object of this chapter

This chapter describes the NEPO_DIA, TEPO_DIA DFB.

What Is in This Chapter?

This chapter contains the following topics:

Description

Description of the Function

These DFBs are used to monitor, command, and carry out diagnostics of an operating part element, i.e. any device that acts directly on manufactured items and the environment.

These DFBs, defined by a "pre-actuator-actuator/sensor", maintain positioning between two points of reference (whether monitored or not), for linear or rotating movement carried out at a constant speed.

Areas of use

- commanding jacks (monostable, bistable or middle-point distributers),
- commanding certain positioning motors,
- binding, unit, machining, and turntable command.

Difference between the two DFBs

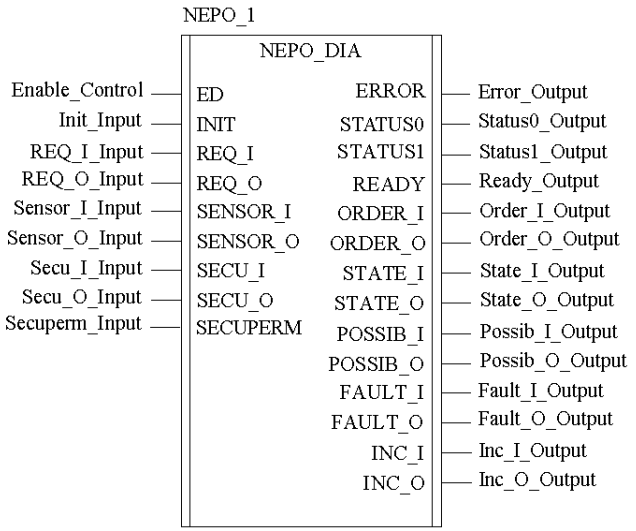
The `TEPO_DIA` DFB is identical to the `NEPO_DIA` DFB. Its sole limitation is that it can only manage linear movement (i.e. not rotating). As a result, the `ROTATION` and `ONEWAY` public variables do not exist for this DFB.

Additional parameters EN and ENO can be configured.

NOTE: This diagnostic DFB cannot be used in a user DFB.

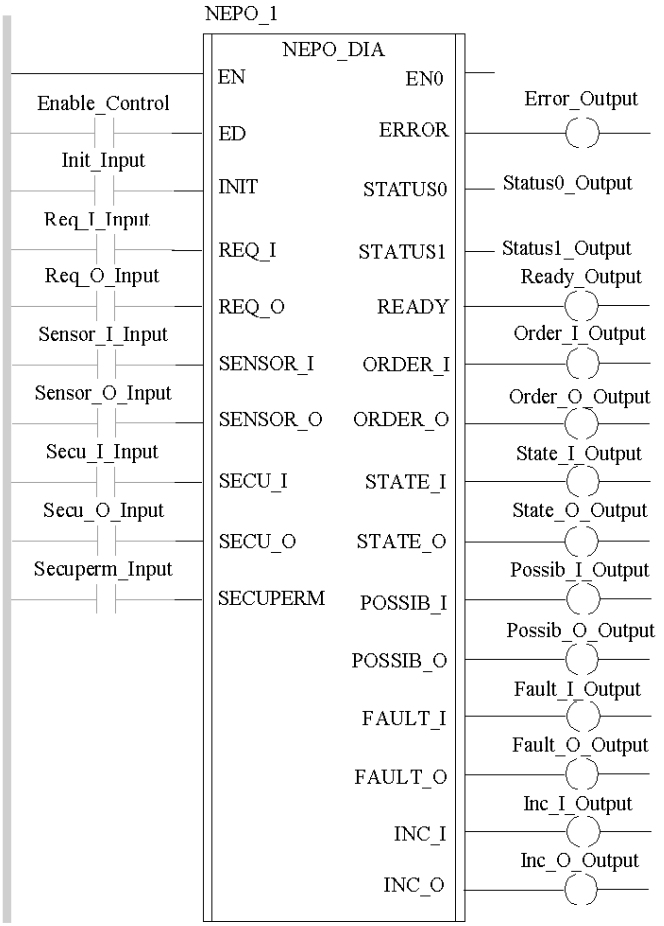
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL NEPO_1 (ED := Enable_Control, INIT := Init_Input, REQ_I :=
Req_I_Input, REQ_O := Req_O_Input, SENSOR_I := Sensor_I_Input, SENSOR_O
:= Sensor_O_Input, SECU_I := Secu_I_Input, SECU_O := Secu_O_Input,
SECUPERM := Secuperm_Input, ERROR => Error_Output, STATUS0 =>
Status0_Output, STATUS1 => Status1_Output, READY => Ready_Output, ORDER_I
=> Order_I_Output, ORDER_O => Order_O_Output, STATE_I => State_I_Output,
STATE_O => State_O_Output, POSSIB_I => Possib_I_Output, POSSIB_O =>
Possib_O_Output, FAULT_I => Fault_I_Output, FAULT_O => Fault_O_Output,
INC_I => Inc_I_Output, INC_O => Inc_O_Output,)
```

Representation in ST

Representation:

```
NEPO_1 (ED := Enable_Control, INIT := Init_Input, REQ_I := Req_I_Input,
REQ_O := Req_O_Input, SENSOR_I := Sensor_I_Input, SENSOR_O :=
Sensor_O_Input, SECU_I := Secu_I_Input, SECU_O := Secu_O_Input, SECUPERM
:= Secuperm_Input, ERROR => Error_Output, STATUS0 => Status0_Output,
STATUS1 => Status1_Output, READY => Ready_Output, ORDER_I =>
Order_I_Output, ORDER_O => Order_O_Output, STATE_I => State_I_Output,
STATE_O => State_O_Output, POSSIB_I => Possib_I_Output, POSSIB_O =>
Possib_O_Output, FAULT_I => Fault_I_Output, FAULT_O => Fault_O_Output,
INC_I => Inc_I_Output, INC_O => Inc_O_Output,);
```

Description of Parameters

The following table describes the input parameters:

Parameter	Type	Description
ED	EBOOL	DFB activation bit When ED = 0, the DFB is not executed. The default value is 0.
INIT	EBOOL	Fault acknowledgement bit When on 1, this bit acknowledges the faults indicated by the ERROR bit and the STATUS0 word. It is reset to 0 by the DFB. The default value is 0.
REQ_I, REQ_Q	EBOOL	Request bits These bits are set on 1 by the command part to request an "input" and "output" movement respectively. The default value is 0.
SENSOR_I, SENSOR_O	EBOOL	Information input bits These inputs receive position information from all the "input" and "output" position sensors respectively. The default value is 0.
SECU_I, SECU_O	EBOOL	Safety condition These inputs are used to link up the safety conditions of "input" and "output" movements respectively. The default value is 0.
SECUPERM	EBOOL	Operating conditions. This input is used to link up continuous operating conditions. The default value is 0.

The following table describes the output parameters:

Parameter	Type	Description
ERROR	EBOOL	Fault bit This bit is set on 1 as soon as a fault appears and providing that the fault has not been masked (see: *Selection mask for public variables, page 122*) The default value is 0.
STATUS0 STATUS1	INT	Fault type These two words indicate the type of fault. STATUS0 indicates faults linked to DFB operation. STATUS1 is kept for configuration faults. (see:*Description of NEPO_DIA and TEPO_DIA DFB status words, page 113*) The default value is 0.
READY	EBOOL	DFB availability ● when on 1, the DFB is in command mode (setting orders). ● when on 0, the DFB is in recalibration mode (awaiting reference point). The default value is 0.
ORDER_I, ORDER_O	EBOOL	Activation indicator When on 1, these bits indicate that the "input" and "output" commands have been activated respectively. The default value is 0.
STATE_I, STATE_O	EBOOL	Input position When on 1, these bits indicate that the "input" and "output" commands are being checked. The default value is 0.
POSSIB_I, POSSIB_O	EBOOL	Availability indicator These bits indicate that the DFB is ready to accept "input" and "output" movement requests respectively. The default value is 0.
FAULT_I, FAULT_O	EBOOL	Fault bit These bits indicate a constant fault during "input" and "output" movements (inoperative position) respectively. The default value is 0.
INC_I, INC_O	EBOOL	Fault bit In the absence of an order or request, these bits indicate an incoherence respectively: ● between the "input" state awaited by the automatic process (RESEQ_1 or ORIGIN data) and the position recorded by the DFB. ● between the "output" state awaited by the automatic process (RESEQ_O data) and the position recorded by the DFB. The default value is 0.

Description of NEPO_DIA and TEPO_DIA DFB status words

Introduction

When the DFB detects a fault, the latter is indicated via the words `STATUS0` and `STATUS1` (several faults can be indicated at the same time).

Whether faults are latched or not depends on the selection mask values of the DFB operation when the fault happens: `RST_ORD` and `RST_FB` :

- a fault selected in `RST_FB` will be recorded in `STATUS0` until it disappears and is acknowledged by `INIT` (the DFB changes to shift mode),
- a fault selected in `RST_ORD` will be recorded in `STATUS0` until it disappears and is acknowledged by `INIT` (the DFB remains in monitor/command mode),
- all other (non-selected) faults will stop being indicated when the cause of the fault disappears.

A fault selected in `SET_ERR` sets the `ERROR` bit to 1.

Status words 0

The following table describes the meaning of status word 0 bits for the NEPO_DIA and TEPO_DIA DFBs.

Bit	Error	Description
bit 0 =1	Error on command or abnormal sensor information	The DFB has detected an aberrant command or incoherent information about positions. Aberrant commands: "input" and "output" requests present at the same time, using the "input" command for a monostable actuator with a single request, expected "input" (RESEQ_1) and "output" RESEQ_0 states present at the same time. Incoherent position information : non-merged position sensors for a rotating movement, unchecked position with active position sensor, a position being checked by several sensors, and SENSOR_I/O and NOSENS_I/O variables active at the same time.
bit 1 =1, bit 2=1	Unexpected "input" sensor Unexpected "output" sensor	When in position, at least one opposing position sensor is active during time beyond the authorized timing, configured in APP_TIME. After having reset, the sensor for the position just left reappears during a time period outside the authorized time, which is defined in APP_TIME. During recalibration, at least one sensor is present at each position.
bit 3 =1, bit 4 =1	Mistimed "input" sensor Mistimed "output" sensor	At least one go to position sensor is present before the minimum movement time, defined in RMIN_I or RMIN_O .
bit 5 =1, bit 6 =1	Late "input" sensor Late "output" sensor	At least one go to position sensor is not yet in place beyond the maximum time given to a movement, and defined in RMAX_I or RMAX_O.
bit 7 =1, bit 8 =1	"Input" sensor disappearance "Output" sensor disappearance	While in position, at least one sensor has disappeared during a period outside the tolerated time, configured in DIS_TIME . While recalibrating, no position is found.
bit 9 =1,	Condition of continuous disappearance	Continuous conditions have disappeared during a movement.
bit 10 =1, bit 11=1	Disappearance of the safety condition for the "input" movement Disappearance of the safety condition for the "output" movement	The safety condition has disappeared during a movement.
bit 12 =1, bit 13=1	"Input" request refused "Output" request refused	A request cannot be accepted by the DFB (safety conditions and/or continuously absent conditions etc..)
bit 12 =1, bit 13=1	"Input" sensor that has not reset "Output" sensor that has not reset	At least one sensor for the position just left has not reset after the minimum movement time, defined in RMIN_I or RMIN_O .

Status words 1

Status word 1 detects configuration faults. During a DFB initialization (i.e. application transfer, cartridge changing etc), the DFB is in "outside operation mode", and is awaiting the reference point At this point it can detect configuration errors which impede its operation, and they are indicated by the STATUS1 output parameter.

The following table describes the meaning of status word 1 bits for the NEPO_DIA and TEPO_DIA DFBs.

Bit	Description
bit 0 =1	Invalid actuator type (faulty CONFIG value).
bit 1 =1	"input" position and "output" position selected are not monitored.
bit 2 =1	Rotating movement and one of the selected positions not monitored.
bit 3 =1	Rotating movement, monostable and in one direction only.
bit 4 =1	Maximum duration of a movement below or equal to the minimum duration.
bit 5 =1	Modeling and training modes of movement periods.
bit 6 =1	Translation and single directional movement only.
bit 7 =1	Training mode for movement periods and non-monitored positions.
bit 8 =1	Rotating movement and positions monitored differently.
bit 9 =1	Incompatible selected CONFIG and ET RST_ORD selection mask.
bit 10 =1	The CONFIG selected and ET unmonitored position are incompatible (actuator types 2, 7 or 11 and NBSENS_I or NBSENS_O = 0).
bit 11 =1	Selection masks RST_ORD and RST_FB are incompatible. (the errors selected in RST_FB must also be selected in RST_ORD).
bit 12 =1	Selection masks RST_ORD, RST_FB and SET_ERR are incompatible. (the errors selected in RST_FB and RST_ORD must also be selected in SET_ERR).
bit 13 =1	Rotating movement and selection mask RST_FB are incompatible. (ROTATION =1 and the error sensor(s) not reset and not selected in RST_FB).

Description of the NEPO_DIA and TEPO_DIA DFB time management variables

General

The time management public variable values express a time equal to N x 100 ms, where N is the value of the BASE constant.

Permitted values are integer numbers between 0 and 32767 inclusive.

Description of variables

The following table describes the public variables:

Name	Type	Description
RMIN_I, RMIN_O	INT	Minimum duration reference. These 2 words act as the minimum duration reference for "input" and "output" movements respectively. By default or on RESET_FB request, these words are initialized from the values of IMIN_I and IMIN_O (or at 0 if IMIN_I = IMAX_I = 0, IMIN_O = IMAX_O = 0) respectively. This variable can be modified by the program, its default value is 0.
RMAX_I, RMAX_O	INT	Maximum duration reference. These 2 words act as the maximum reference for RMAX_O and "input" and "output" movements respectively. By default or on RESET_FB request, these words are initialized from the values of IMAX_I and IMAX_O (or from 32767 if IMIN_I = IMAX_I = 0, IMIN_O = IMAX_O = 0) respectively. This variable can be modified by the program, its default value is 0.
TIME_I, TIME_O	INT	Time. These two words contain the current time for the "input" and "output" movements in progress respectively, or the time the last "input" and "output" movements occurred individually. The default value is 0.
TMIN_I, TMIN_O	INT	Automatic operation zone to be monitored. These 2 words memorize the minimum amount of time that was required for the "input" and "output" movements respectively. By default or on RESET_CT request, TMIN_I and TMIN_O take on the RMAX_I or RMAX_O value if ADJ_TIME = 1; and IMAX_I or IMAX_O if ADJ_TIME = 0. The default value is 0.

Name	Type	Description
TMAX_I, TMAX_O	INT	Acknowledge request. These 2 words memorize the maximum amount of time that was necessary for the "input" and "output" movements respectively. By default or on RESET_CT request, TMAX_I and TMAX_O take on the RMIN_I or RMIN_O value if ADJ_TIME = 1; and IMIN_I or IMIN_O if ADJ_TIME = 0. The default value is 0.
IMIN_I, IMIN_O	INT	Minimum time. These 2 words define the minimum authorized time for the "input" and "output" movements respectively. On DFB initialization, the values of IMIN_I and IMIN_O are recopied into RMIN_I and RMIN_O respectively (if IMIN_I and IMIN_O are not all 2 to 0). The default value is 0.
IMAX_I, IMAX_O	INT	Maximum time. These 2 words act as the minimum duration reference for "input" and "output" movements respectively. By default or on RESET_FB request, these words are initialized from the values of IMIN_I and IMIN_O (or at 0 if IMIN_I = IMAX_I = 0, IMIN_O = IMAX_O = 0) respectively. The default value is 0.
DIS_TIME	INT	Duration of sensor disappearance. These 2 words act as the maximum reference for RMAX_O and "input" and "output" movements respectively. By default or on RESET_FB request, these words are initialized from the values of IMAX_I and IMAX_O (or from 32767 if IMIN_I = IMAX_I = 0, IMIN_O = IMAX_O = 0) respectively. The default value is 0.
APP_TIME	INT	Duration of sensor appearance. These two words contain the current time for the "input" and "output" movements in progress respectively, or the time the last "input" and "output" movements occurred individually. The default value is 0.
BASE	INT	Base time coefficient. These 2 words memorize the minimum amount of time that was required for the "input" and "output" movements respectively. By default or on RESET_CT request, TMIN_I and TMIN_O take on the RMAX_I or RMAX_O value if ADJ_TIME = 1; and IMAX_I or IMAX_O if ADJ_TIME = 0. The default value is 1.

Description of NEPO_DIA and TEPO_DIA DFB specific request variables

Description of variables

The following table describes the public variables used for specific requests.

Name	Type	Description
RESET_CT	EBOOL	Reset counters. Set on 1, this bit re-initializes the counters while memorizing the minimum , maximum and actual time of "input" and "output" movements (TMIN_I, TMIN_O, TMAX_I, TMAX_O, TIME_I and TIME_O), the number of movement requests accepted (N_REQ), and the number of errors detected (N_ERROR). It is reset to 0 by the DFB. This variable can be modified by the program, its default value is 0.
RESET_FB	EBOOL	Reset DFB. Set on 1, this bit re-initializes the DFB (except for data managed by RESET_CT). It is reset to 0 by the DFB. This variable can be modified by the program, its default value is 0.

Description of NEPO_DIA and TEPO_DIA DFB configuration variables

Description of variables

The following table describes the public variables:

Name	Type	Description
CONFIG	INT	Type of actuator configuration. This word is used to configure the type of actuator command (see following table). By default, CONFIG = -1 (this value is deliberately faulty in order to make choosing the actuator type mandatory). The default value is -1.
NBSENS_I, NBSENS_O	INT	Position monitoring. These 2 words are used to define how the DFB monitors "input" and "output" positions respectively: ● NBSENS_I (or NBSENS_O) = 0; position is not monitored, ● NBSENS_I (or NBSENS_O) = 1; position is monitored with SENSOR_I (or SENSOR_O) input, ● NBSENS_I (or NBSENS_O) = 2; position is monitored with SENSOR_I (or SENSOR_O) input (working state of all sensors) and public variable NOSENS_I (or NOSENS_O) (resting state of all sensors). The default value is 1.
ROTATION	EBOOL	Movement type. This bit defines a rotating movement when set on 1. This parameter does not exist for the TEPO_DIA DFB The default value is 0.
ONEWAY	EBOOL	Movement sequence. When on 1, this bit defines a rotating movement with the possibility of linking up several movements in the same direction. This parameter does not exist for the TEPO_DIA DFB The default value is 0.
SIMUL	EBOOL	Simulation mode. When on 1, this bit sets the DFB into modeling mode. The default value is 0.

Selecting the actuator type

The CONFIG internal constant value is used to select the actuator and order type. The following different configurations are possible:

CONFIG	Actuator	Command	Command logic
0	monostable actuator, single order (ORDER_O)	single request (REQ_O)	order if requested (type 1)
1	monostable actuator, single order (ORDER_O)	two requests (REQ_O, REQ_I)	order maintained until reverse order (type 2)
2	monostable actuator, single order (ORDER_O)	two requests (REQ_O, REQ_I)	order if request and order clash in position, unlocking on reverse request or loss of position (type 5)
3	bistable actuator two separate orders (ORDER_O, ORDER_I)	two requests (REQ_O, REQ_I)	order if requested (type 1)
4	bistable actuator two separate orders (ORDER_O, ORDER_I)	two requests (REQ_O, REQ_I)	order maintained until reverse order (type 2)
5	bistable actuator two separate orders (ORDER_O, ORDER_I)	two requests (REQ_O, REQ_I)	order if request and position not achieved (type 3). The pre-actuator reacts on a pulse, pointless to maintain an order
6	bistable actuator two separate orders (ORDER_O, ORDER_I)	two requests (REQ_O, REQ_I)	order maintained until reverse request and in position (type 4)
7	bistable actuator two separate orders (ORDER_O, ORDER_I)	two requests (REQ_O, REQ_I)	order if request and order clash in position, unlocking on reverse request or loss of position (type 5)
8	multistable actuator with two separate orders (ORDER_O, ORDER_I)	two requests (REQ_O, REQ_I)	idem 4
9	multistable actuator	two requests (REQ_O, REQ_I)	idem 6
10	multistable actuator	two requests (REQ_O, REQ_I) and absence of request	idem 5 Intermediate stop permitted (absence of request)
11	multistable actuator	two requests (REQ_O, REQ_I) and absence of request	idem 7 Intermediate stop permitted (absence of request)

NOTE: CONFIG = 8 to 11 : intermediate stop possible through the default selected in RST_ORD.

Description of the NEPO_DIA and TEPO_DIA DFB fault management variables

Fault management public variables

The following table describes the public variables used to configure the DFB action during an error.

Name	Type	Description
SET_ERR	INT	Error selection. This word is used to select the errors which set the ERROR bit to 1. The default value is 16#0FE7.
RST_ORD	INT	Reset orders to 0. Reset orders to zero (ORDER_I and ORDER_O). These errors are stored in STATUS0 until they are acknowledged. They must also be selected in the SET_ERR mask. The default value is 16#0F87.
RST_FB	INT	Error selection. This word is used to select the faults which put the DFB in recalibration mode. These errors are stored in STATUS0 until they are acknowledged. They must also be selected in the SET_ERR mask. The default value is 16#0187.

Selection mask for public variables

The following table gives the selection mask default values for the SET_ERR, RST_ORD and RST_FB. variables

Bit	Meaning	SET_ERR (16#0FE7)	RST_ORD (16#0F87)	RST_FB (16#0187)
0	Command error	X	X	X
1	Unexpected "input" sensor	X	X	X
2	Unexpected "output" sensor	X	X	X
3	Mistimed "input" sensor	-	-	-
4	Mistimed "output" sensor	-	-	-
5	Late "input" sensor	X	-	-
6	Late "output" sensor	X	-	-
7	"Input" sensor disappearance	X	X	X
8	"Output" sensor disappearance	X	X	X
9	Condition of continuous disappearance	X	X	-
10	"Input" safety condition disappearance	X	X	-
11	"Output" safety condition disappearance	X	X	-
12	"Input" request refused	-	-	-
13	"Output" request refused"	-	-	-
14	"Input" sensor that has not reset	-	-	-
15	"Output" sensor that has not reset	-	-	-

NOTE: A bit indicated by a cross means that it is selected and that the corresponding error will not be masked.
The DFB is thus used to execute a movement together with an error and whatever the error may be.
For example, if bit 9, selecting the error; "disappearance of continuous operating conditions", is set to 0, the orders can be activated even if this condition disappears.

Description of the DFB NEPO_DIA and TEPO_DIA control variables

Public variables reliability indicators

The following table describes the public variables used as reliability indicators.

Name	Type	Description
N_REQ	INT	Storing the number of requests accepted by the DFB. This word takes value 0 when RESET_CT is set to state 1 or in the event of counter overrun (when limit value 32767 is reached). The counter overrun N_REQ sets both it and the N_ERROR counter back to zero The default value is 0.
N_ERROR	INT	Storing the number of errors detected by the DFB. This word takes value 0 when RESET_CT is set to state 1 or in the event of counter overrun (when limit value 32767 is reached). The counter overrun N_ERROR sets both it , and counter N_REQ back to zero. The default value is 0.

Cycle reset public variables

The following table outlines the public variables used for the cycle reset.

Name	Type	Description
OUTCTRL	EBOOL	Authorization to send orders. After a default selected in RST_FB, this data is used to authorize the DFB to send orders without monitoring the sensors, in order to set the operative section in a checked recalibration position. The SECU_I, SECU_O and SECUPERM inputs must be valid. This variable can be modified by the program, its default value is 0.
ORIGIN	EBOOL	Awaiting source position. This bit indicates that the automatic operation is awaiting the "source position" state (equivalent to RESEQ_I but priority). This variable can be modified by the program, its default value is 0.
RESEQ_I, RESEQ_O	EBOOL	Awaiting state. These 2 bits signal that the automatic operation is awaiting either the "re-input" state or "output" state respectively. This variable can be modified by the program, its default value is 0.

Position monitoring public variables

The following table describes the public variables used for position monitoring.

Name	Type	Description
NOSENS_I NOSENS_O	EBOOL	Position monitoring. These bits give the reverse position of the sensors twisted on the respective inputs SENSOR_I and SENSOR_O. These bits are only used if the DFB is configured to monitor the positions with the help of this data (constant internal NBSENS_I and/or NBSENS_O = 2).

State public variables

The following table outlines the public variables used as status indicators.

Name	Type	Description
ADJ_TIME	EBOOL	Reference time acquisition. This bit signals that the reference times of the movements have been acquired (training mode). The default value is 0.
MVT_I, MVT_O	EBOOL	Transient state of a movement. These 2 bits signal the transient state of a MVT_O movement "re-input" or "output" engaged and not completed (not reached determined position). The default value is 0.
EXPECTED	EBOOL	Awaiting state. This bit signals that the DFB is waiting for an end of movement sensor to appear (the movement has been engaged for more than RMIN_I or RMIN_O or has been interrupted). The default value is 0.

Operating mode public variables

The following table describes the public variables used for configuring the DFB on cycle resumption.

Name	Type	Description
ORD_MNT	EBOOL	Error selection. If this bit is at state 1, the orders are re-activated following the disappearance of the indication in STATUS0, or following faults which have reset the orders to 0. The default value is 0.
NEW_REQ	EBOOL	Reset orders to 0. If this bit is at state 1, new requests are required after the fault detection that has put the DFB in recalibration mode (i.e. detection of a fault selected in RST_FB). The default value is 1.

Description of NEPO_DIA and TEPO_DIA DFB general public variables

General public variables

The following table describes the general public variables.

Name	Type	Description
AREA_NR	INT	Automatic operation zone to be monitored. This word is used to specify which automatic operations zone is being monitored by the diagnostics DFB. Examples: ● Manufacture : n° 1 ● Reaming : n° 2 ● Threading: n° 3 AREA_ NR must have a value of 1, 2 or 3 for the user to identify which section of the automatic operation has a fault. It is advisable to make the above division correspond with the division in the function module. AREA_ NR can take on a value between 0 and 15. The default value is 0.
OP_CTRL	EBOOL	Acknowledge request. This bit signals whether or not a DFB instance must be acknowledged by the operator : ● OP_CTRL = 0 : not acknowledged by the operator, ● OP_CTRL = 1 : acknowledged by the operator, The default value is 0.

Operating mode public variables

The following table describes the public variables used for configuring the DFB on cycle resumption.

Name	Type	Description
ORD_MNT	EBOOL	Error selection. If this bit is at state 1, the orders are re-activated following the disappearance of the indication in STATUS0, or following faults which have reset the orders to 0. The default value is 0.
NEW_REQ	EBOOL	Reset orders to 0. If this bit is at state 1, new requests are required after the fault detection that has put the DFB in recalibration mode (i.e. detection of a fault selected in RST_FB). The default value is 1.

How to pre-program NEPO_DIA and TEPO_DIA DFBs

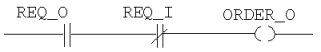
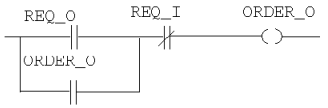
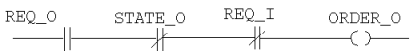
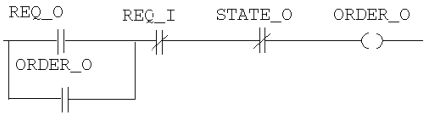
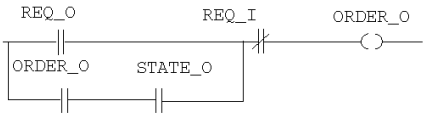
General

This operation defines NEPO_DIA and TEPO_DIA DFB functioning.

Procedure

The following table describes the procedure for pre-programming NEPO_DIA or TEPO_DIAfunction blocks:

Step	Actions
1	Select the type of actuator, defined by the CONFIG internal constant : monostable (ORDER_I unused) or bistable (ORDER_O and ORDER_I used).
2	Select the movement type, defined by the ROTATION constant : translation or rotation. If the rotating movement is chosen, the "input" and "output" sensors are merged and the ONEWAY constant defines whether the movement is in one or two directions of rotation.

Step	Actions
3	Select the type of orders given to the actuator. These orders are applied to the actuators according to the following equations for "output" movements. These equations are the same for "input" movements (replace _O with _I and vice versa) : Order if requested (type 1)  Order stored up to reverse request (type 2)  Order if requested and up to position (type 3)  Order stored up to reverse request and position (type 4)  Order if request and order clash on position (type 5) 
4	Select how the physical positions and the element of the operating section are being monitored by the DFB. It is defined by the `NBSENS_O` and `NBSENS_I` internal constants.
5	Select the action of the DFB on detection of an error: • `SET_ERR` data defines the error which sets the `ERROR` bit to 1, • `RST_ORD` data defines the faults which engage `ORDER_I` and `ORDER_O` outputs, • `RST_FB` data defines the faults which switch the DFB into "recalibration" mode. The setting of a bit in either `RST_ORD` or `RST_FB` to 1 selects the error associated with the bit of the same rank in `STATUS0`. • `ORD_MNT` data defines whether or not orders should be re-activated following the disappearance of the indication in `STATUS0`, or following faults which have set orders to 0 during a movement. • `NEW_REQ` data defines whether new requests are needed after a fault which has set the DFB to "recalibration" mode. By default, new requests are demanded.

Step	Actions
6	Select the movement duration. ● IMAX_I and IMAX_O data define the maximum duration of "input" and "output" movements, ● IMIN_I and IMIN_O data define the minimum duration of "input" and "output" movements. The values express times on a N x 100 ms base, where N is the value of the BASE. On DFB initialization, these values are copied into RMAX_I, RMAX_O, RMIN_I and RMIN_O. If IMIN_I and IMAX_I information (or IMIN_O and IMAX_O), which define movement duration, are on 0, the DFB will learn this duration.

NBSENS_O and NBSENS_I internal constants

The table below describes the coding of NBSENS_O and NBSENS_I internal constants.

NBSENS_O or NBSENS_I	Monitoring
0	Non-monitored position. This position is considered reached if the DFB is waiting for it to do so, or non-reached if the DFB is not waiting for it to do so. Any fault associated with this position (sensor not reset, not expected, etc.) will not be indicated. In other words, this means that if a position is selected that is not monitored, the DFB will stop the movement (towards this position) as soon as the RMAX_I or RMAX_O time limit has been reached and will examine the EPO potentially. On the other hand, on initialization or recalibration, the reference point can only continue from a monitored position.
1	Position monitored via SENSOR_O or SENSOR_I input.
2	Position monitored physically with several sensors. The DFB checks the position with 2 information sources: SENSOR_O (or SENSOR_I) and NOSENS_O (or NOSENS_I), with :POSITION_O = SENSOR_O . NOSENS_O and POSITION_I = SENSOR_I. NOSENS_ISENSOR_O or SENSOR_I represents the working state of all the sensors. NOSENS_O or NOSENS_I represents the resting state of all the sensors.

NOTE: The two positions cannot be chosen, and both cannot be monitored. If this is the case, the DFB indicates that there is a configuration fault (STATUS1) and becomes unusable

How the command function blocks and the operative section diagnostics work: NEPO_DIA and TEPO_DIA

General

The DFB places itself into the command by maintaining the link between the application program and the action and vice versa :

- inputs REQ_O and REQ_I enable requests to be received,
- outputs ORDER_O and ORDER_I send the orders through to the operator,
- Inputs SENSOR_O andSENSOR_I and if necessary the data NOSENS_O and NOSENS_I provide the DFB with information on the physical positions of "output" and "re-input".

The movement period is checked through the dataRMIN_O, RMAX_O, RMIN_I and RMAX_I .

The inputsSECU_O andSECU_I set the safety conditions before being accepted during the "re-input" and "output" movements.

Input SECUPERM shows the operating conditions of the machine which have to be accepted during the movements.

Operation

During normal function (Reset-command mode and bit READY =1), the DFB commands the movement(s) by carrying out the following operations.

Phase	Description
1	Sensor check (inputs SENSOR_I and SENSOR_O and if necessary NOSENS_I and NOSENS_O)
2	Request monitoring (inputs REQ_I and REQ_O)
3	monitoring of the movement period
4	minimum and maximum latch of movement periods
5	training on movement periods
6	detecting and reacting to errors
7	developing the reports for the functional command
8	developing the operator commands (outputs ORDER_I and ORDER_O)
9	updating the functional indicators
10	aiding the cycle relaunch

Movement authorization

Where movement requests are not present and if on the face of it they are authorized (the "request refused" information would not be activated in STATUS0), the DFB positions its outputsPOSSIB_I and POSSIB_O in state 1.

NOTE:

- SECUPERM (continuous operating conditions) or SECU_O/I (movement safety conditions) become part of the bit evaluation POSSIB_O/I if their absence brings the orders down again; or in other words, if the errors included in them are selected in the mask RST_ORD .
- a movement will be refused if an error selected in RST_ORD is present at the time the request is made.
- if the reverse request is present during a movement request, its execution will always be prevented (this error can not be masked). Furthermore, during execution of a movement, a reverse request will cancel the order, whether the request is accepted or not,
- in position a request has no effect on commands of order type up to position (type 3 or 4): POSSIB takes this condition into account.

Information sensor

In position, the disappearance of a sensor is only signaled at the end of the time specified by DIS_TIME . This reset is disabled as soon as a movement request has been accepted.

Outside recalibration mode, the appearance of an unexpected sensor is only signaled after the time specified by APP_TIME .

Information on movement

The DFB positions the data which provide information on the execution of the movement :

- the outputs STATE_I and STATE_O specify the state of the movement checked by the DFB (position reached). FAULT_I and FAULT_O signal an error on the movement as it happens,
- INC_I and INC_O signal a discrepancy between the expected position (data RESEQ_I, RESEQ_O and ORIGIN) and the outputs STATE_I and STATE_O in the absence of an order or a request,
- the internal data MVT_I and MVT_O signal that the involved movement has not yet finished (inoperative position).

During movement, the safety conditions linked to the movement and the permanent conditions have to remain enabled according to the masks RST_FB and RST_ORD.

Recalibration mode

Following an error configured in RST_FB or following a request RESET_FB, triggering off a switch to recalibration mode, the DFB performs the operations detailed below :

- deactivation of the bit READY,
- deactivation of outputs STATE_I/O and ORDER_I/O,
- consideration of its configuration data and continuation of the action if there is no configuration error in STATUS1 (only in the event of a request RESET_FB),
- pending request INIT to remove the faults which are no longer present in STATUS0 (only in the event of a fault). The DFB is then in a RESET state whereby it is "Frozen" : it no longer tests the constant conditions, the safety conditions and its outputs stop changing,
- switching to recalibration mode to relocate a source position,
- send to reset-command mode as soon as it detects a coherent sensor configuration.

Helping to resume the cycle.

The RESEQ_I, RESEQ_O and ORIGIN data inform the DFB of the state expected by the automation.

The DFB stores the last expected state (presented as 1 from RESEQ_I, RESEQ_O or ORIGIN).

If the state or the movement checked by the DFB does not match the expected state (the last one stored), the outputs INC_I and INC_O signal a discrepancy. When the DFB switches into recalibration mode, the states expected before the switch are stored.

Saving the minimum and maximum periods of the movements

For each movement executed, the DFB (in non-simulated mode) saves the period and stores the minimum and maximum periods in the TMIN_I, TMAX_I, TMIN_O and TMAX_O data.

The maximum periods are only stored if they are below the maximum reference values RMAX_I and RMAX_O. The RESET_CT data enables the minimum and maximum movement values to be reset.

Training on the movement periods,

It is possible for the DFB to learn the periods of the movements. In order to do this, the time management configuration data must be set at 0.

Whenever a movement is executed without interruption, the data RMIN_O (or RMIN_I)adopts a value equal to half of the movement period; whilst RMAX_O (or RMAX_I) adopts a value equal to 1 and a half times this value.

A movement can be said to have been executed without interruption when it has not stopped of its own accord, due either to the absence of requests for operators enabling it, or by default, which puts the commands at zero.

Once the periods of the two movements have been established, the ADJ_TIME bit adopts value 1.

Special features of the rotational movement

Position Evaluation

If the two inputs SENSOR_I and SENSOR_O (and if necessary NOSENS_I and NOSENS_O), are not identical, the "command error" fault will be signaled.

In position, if one or both of the two inputs fall back to 0, the DFB will begin to count the disappearance period of the sensor(s) and will do so until the 2 inputs readopt value 1 in the same time.

During movement, the position will be considered as "quit" if both of the 2 sensors are seen at 0 at least once. The position will be considered as "reached" if both of the sensors are seen at 1.

The only faults signaled concerning the sensor are therefore :

- in position : "disappeared sensor(s)" or "non returning sensor(s)",
- in movement : "misplaced sensor(s)" or "delayed sensor(s)",

Request maintained and position reached

In rotation only one position is checked (the two sensors are taken together). On position and conversely to the translation movement, the two requests are accepted and the two possible movements are begun.

When a movement is finished (position reached), if the "re-input" or "output" request is still present, the movement is automatically restarted. In order to prevent this with the rotation movement, the requests are interpreted on rising edge.

Manual mode

Execution of movements in manual mode (machine cycle inoperative) is dependent on functional command, independent of the DFB. The DFB reacts to the commands in the same way as it does in automatic mode.

However, in order for the DFB to be able to function in manual mode, it has to be executed in exactly the same way outside the machine cycle. In order to do this, if a manual command is anticipated for the DFB, it has to be executed in an easily accessible PL7 module, regardless machine cycle status: module executed during each PLC cycle (POST or SR) whose call can be controlled whilst in function or independently of the machine cycle.

Automatic operation modes

The DFB, during application transfer or cartridge replacement, resets all of its data, takes its configuration data into consideration and goes into recalibration mode (READY at 0).

On %S0 request or restart after a power outage, the DFB goes back into recalibration mode (READY at 0). The outputs ORDER_I/O and STATE_I/O are reset to 0. The counters run by RESET_CT are retained as are the reference times.

The reset-command mode will be activated when a position is found, where no fault is signaled and no request present (whatever the NEW_REQ value).

Chapter 15
ONLEVT: Online event

Description

Function description

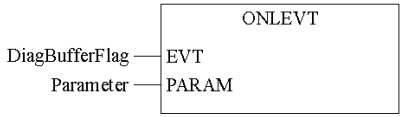
This procedure can enter online events in the diagnostics buffer. The error recognition `E_EFB_ONLEVT` and the `PARAM` input are used for this.

NOTE: If `EVT` is set permanently to TRUE, each acknowledgement of an event will lead to a new entry in the diagnostic buffer. In addition, keeping `EVT` input permanently on `TRUE` will generate unnecessary runtime, due to checking the diagnostic buffer in each scan. Therefore it is recommended to use a rising edge detection at the `EVT` input in the application.

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

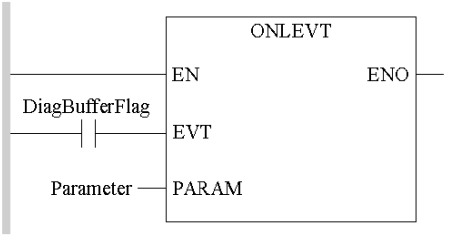
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD DiagBufferFlag
ONLEVT Parameter
```

Representation in ST

Representation:

```
ONLEVT (DiagBufferFlag, Parameter);
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
DiagBufferFlag	BOOL	"1": Entry in diagnostics buffer.
Parameter	WORD	Parameter transferred to the diagnosis buffer.

Runtime error

NOTE: For a list of all block error codes and values, see *Diagnostics, page 190*.

Chapter 16
REGDFB: Alarm saving and dating

Description

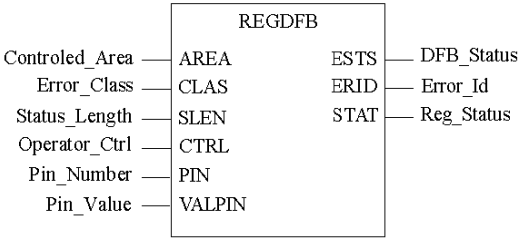
Description of the function

The REGDFB function, entered in the code of a user DFB *(see page 172)*, saves and dates an alarm in the diagnostics buffer.

Additional parameters EN and ENO can be configured.

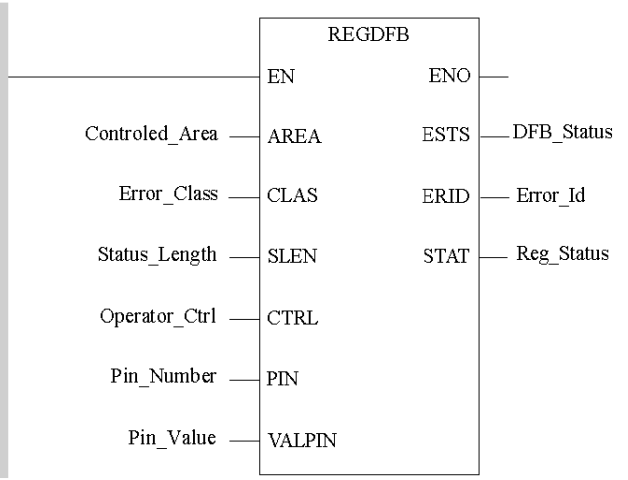
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Controlled_Area
REGDFB Error_Class, Status_Length, Operator_Ctrl, Pin_Number,
Pin_Value, DFB_Status, Error_Id, Reg_Status
```

Representation in ST

Representation:

```
REGDFB(Controlled_Area, Error_Class, Status_Length,
Operator_Ctrl, Pin_Number, Pin_Value, DFB_Status, Error_Id,
Reg_Status);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Controlled_Area	INT	Area of the machine monitored by the DFB: 0 at 15.
Error_Class	INT	Class of error: ● 16#0062
Status_Length	INT	Length of the status: 0, 2 or 4 bytes: ● 0 = no status managed, ● 2 = status managed on one word, ● 4 = status managed on a double word,
Operator_Ctrl	BOOL	1 = Operator acknowledgement required. 0 = no acknowledgement required.
Pin_Number	INT	Incorrect input number. Rule for numbering inputs: only inputs with a "Diag" attribute are taken into account in the numbering. The first input is assigned the number 1.
Pin_Value	BOOL	Value attained on the error input.

The following table describes the output parameters:

Parameter	Type	Comment
DFB_Status	DINT	Status of the DFB (declared in the OUT parameter to switch by address and not by value). Must be updated by the DFB before this function is called.
Error_Id	INT	Error identifier.

Reg_Status	INT	Error registration report.
		● if the registration is successful: Reg_Status = 0 and Error_id is valid,
		● if the registration fails: Error_id is invalid and
		  ● Reg_Status = 1: diagnostics buffer not configured,
		  ● Reg_Status = 2: diagnostics buffer full.
		The system word %SW76 is reserved for reception of the diagnostics DFB registration result (use not compulsory but recommended).

Chapter 17
REGEXT: Registration of expanded FFB errors

Description

Function description

The REGEXT procedure transfers any error information to a diagnosis display and registers the error in the diagnosis buffer.(COND=1). For COND=0 the error is de-registered.

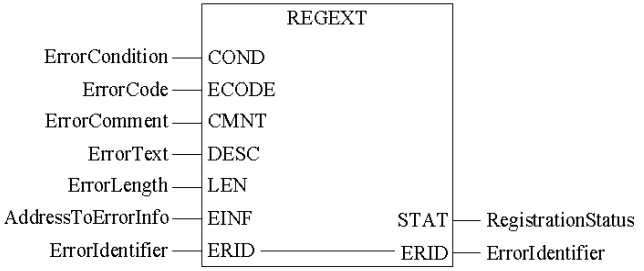
REGEXT makes it possible to transfer an error code, an error description and a description.

REGEXT uses a predefined error type and the diagnostics display shows that the reported error has more information associated with it.

EN and ENO can be configured as additional parameters.

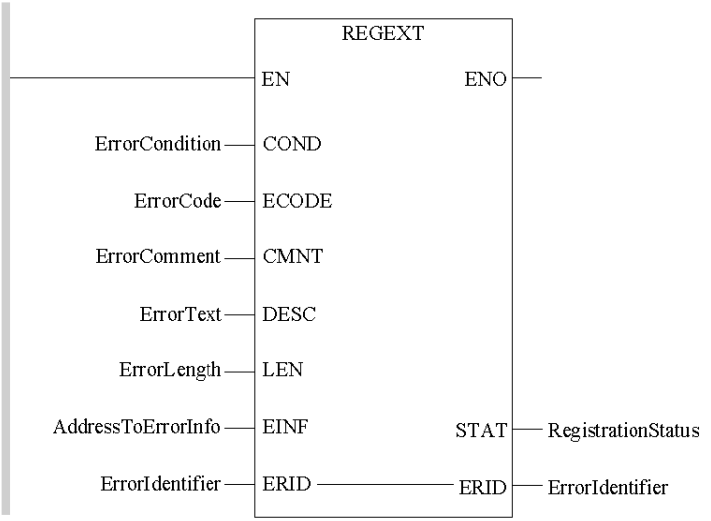
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD ErrorCode
REGEXT ErrorCondition, ErrorText,  ErrorLength,
       AddressToErrorInfo, ErrorIdentifier,
       RegistrationStatus
```

Representation in ST

Representation:

```
REGEXT (ErrorCondition, ErrorText,  ErrorLength,
         AddressToErrorInfo, ErrorIdentifier,
         RegistrationStatus);
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
COND	BOOL	Error condition 0: Error registration 1: Error registration
ECODE	UDINT	Error Code **NOTE:** ECODE must be set to 2. All other values are reserved and give incorrect detected errors in the **Diagnostic Viewer**.
CMNT	STRING	Comment on the error description
DESC	STRING	Error description
LEN	INT	Length of error information (ADR) (max. 48 bytes)
EINF	ANY	Error information Any data to be transferred to the diagnostics display. The format of the data to be transferred must be the same as the data format for the diagnosis display. If the format of the data is not the same the transferred data is displayed in the diagnostics display as hex values.

Description of the input/output parameters:

Parameter	Data type	Meaning
ERID	INT	Error recognition used by the function DEREG *(see page 47)* to de-register active errors (COND=0). **Note:** The connection to other active errors is lost if the same error recognition variable is used for different errors.

Description of the output parameters:

Parameter	Data type	Meaning

STAT	INT	RegistrationStatus
		● If registration is successful: STAT = 0 and ERID is valid
		● If the registration fails: ERID is invalid and
		  ● STAT = 1: diagnosis buffer is not configured
		  ● STAT = 2: Diagnostics buffer full.
		The system word %SW76 is reserved to receive the result of the diagnostic DFB registration (not mandatory but recommended).

Chapter 18
REGIO: I/O Alarm saving and dating

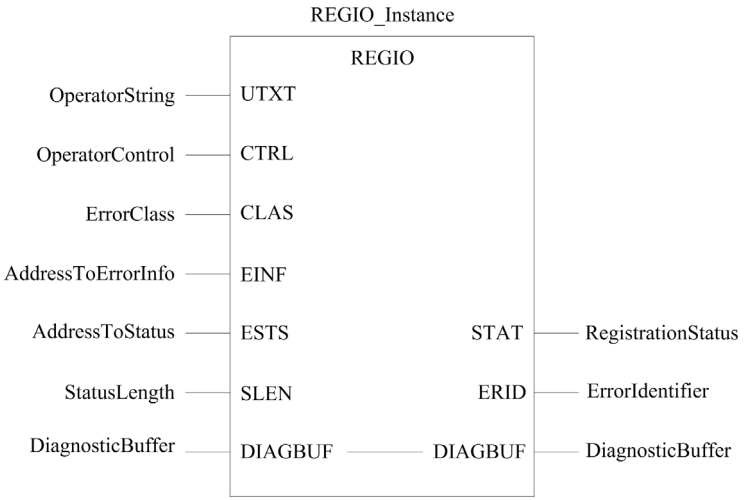
Description

Description of the function

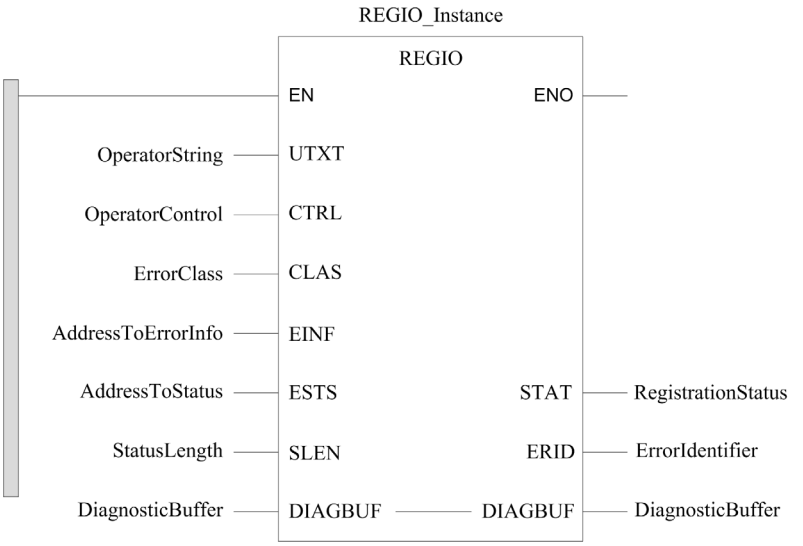
The REGIO function saves I/O alarm in the diagnostics buffer.

Additional parameters EN and ENO can be configured.

Representation in FBD



Representation in LD



Representation in IL

```
LD OperatorString
REGIO ErrorClass, StatusLength, OperatorControl,
DiagnosticBuffer, ErrorIdentifier, RegistrationStatus
```

Representation in ST

```
REGIO(OperatorString, ErrorClass, StatusLength,
OperatorControl, ErrorIdentifier,
RegistrationStatus);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
UTXT	STRING	User text can be used instead of instance comment.
CLAS	INT	Class of error: 16#0062
SLEN	INT	Length of the status: 0, 2 or 4 bytes: ● 0 = no status managed ● 2 = status managed on one word ● 4 = status managed on a double word
CTRL	BOOL	● 1 = Operator acknowledgement is required. ● 0 = no acknowledgement is required.
EINF	ANY ARRAY OF INT	Address of extra information
ESTS	ANY	Address of error status

The following table describes the output parameters:

Parameter	Type	Comment
ERID	INT	Error identifier.

STAT	INT	Registration status.
		● if the registration is successful: STAT= 0 and ERID is valid,
		● if the registration fails: ERID is invalid and
		  ● STAT = 1: diagnostic buffer not configured,
		  ● STAT = 2: diagnostic buffer is full.
		See DIAGBUF.

The following table describes the input/output parameters:

Parameter	Type	Comment
DIAGBUF	DWORD	Diagnostic buffer that contains the registration result

Chapter 19
SAFETY_MONITOR: Safety DFB

Description

Description of the Function

The SAFETY_MONITOR DFB allows data processed by the security monitor to be obtained. Its implementation is identical to that of a DIAG AS-i DFB *(see Premium and Atrium using Unity Pro, AS-i Bus, User manual)*: it can be programmed in any program module (Main, SR or section) in the Ladder (LD), Structured Text (ST) and Instruction List (IL) languages.

The SAFETY_MONITOR DFB is:

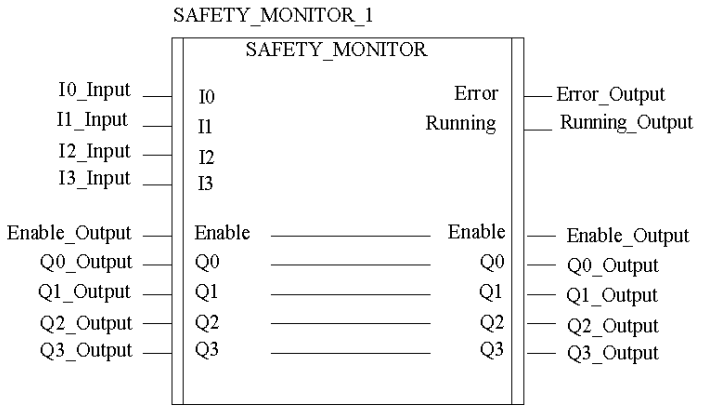
- read- and write-protected
- dedicated to a single security monitor

Additional parameters EN and ENO can be configured.

NOTE: This diagnostic DFB cannot be used in a user DFB.

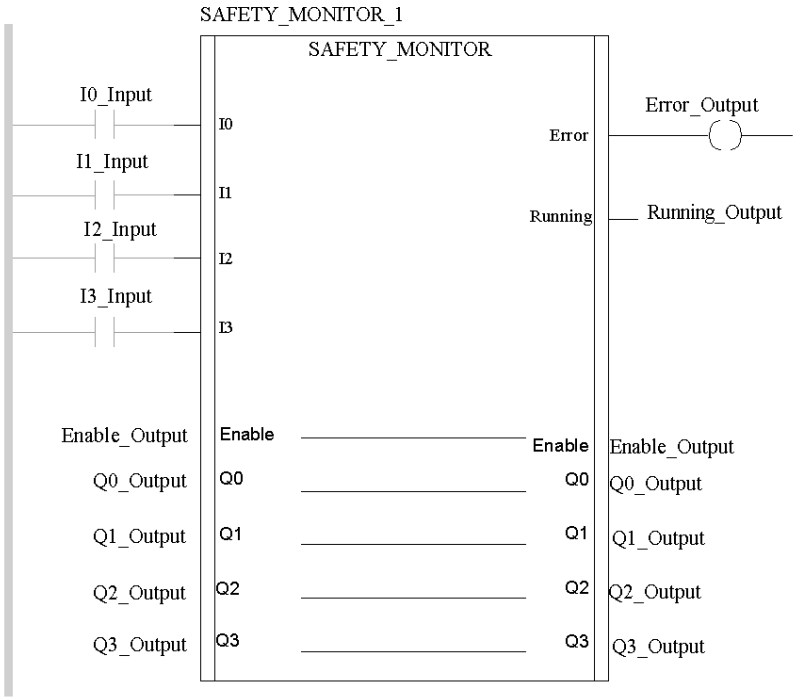
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SAFETY_MONITOR_1 (IO := I0_Input , I1 := I1_Input, I2 := I2_Input,
I3 := I3_Input, Enable := Enable_Output, Q0 :=Q0_Output,  Q1 :=Q1_Output,
Q2 :=Q2_Output, Q3 :=Q3_Output, Error => Error_Output, Running =>
Running_Output)
```

Representation in ST

Representation:

```
CAL SAFETY_MONITOR_1 (IO := I0_Input , I1 := I1_Input, I2 := I2_Input,
I3 := I3_Input, Enable := Enable_Output, Q0 :=Q0_Output,  Q1 :=Q1_Output,
Q2 :=Q2_Output, Q3 :=Q3_Output, Error => Error_Output, Running =>
Running_Output)
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Description
I0	EBOOL	Input variable 0.
I1	EBOOL	Input variable 1.
I2	EBOOL	Input variable 2.
I3	EBOOL	Input variable 3.

The following table describes the input/output parameters:

Parameter	Type	Description
Enable	EBOOL	Activation of the DFB (Cold Start): if this bit is at "1", the DFB executes, otherwise it is deactivated. The information is only usable if Enable = 0.
Q0	EBOOL	Output variable 0.
Q1	EBOOL	Output variable 1.
Q2	EBOOL	Output variable 2.
Q3	EBOOL	Output variable 3.

The following table describes the output parameters:

Parameter	Type	Description
Error	EBOOL	This bit is set at "1" if a detected error appears: DFB error or security bus error (at least one slave is inoperative): ● if it is a DFB error (enable = 0): see Dfb_error (≠ 0) for more information. In the case of a DFB error, the information on the security project is no longer valid. ● it is a case of a security project error (Dfb_error = 0 and Enable = 1): see S1_ to find out the faulty slaves.
Running	EBOOL	This bit is set at "1" during execution of the DFB.

The following table describes the internal public variables:

Name	Type	Description
Abort	EBOOL	If this bit is at "0" in one cycle and at "1" in the next, all the changes between CPU and the security monitor are stopped and the DFB reinitializes. All the internal data of the DFB are set at 0.
Timeout	INT	Timeout of the data exchanges (time base of 100 ms). If the DFB does not receive a correction before this period, the transaction is cancelled, the DFB is deactivated and the error bit is set at "1" (Dfb_stat and Dfb_err are updated).
Moni_err	EBOOL	This bit is at "1" if the monitor is faulty.
Out_1	EBOOL	This bit is at "1" if the OUT1 contact is closed.
Out_2	EBOOL	This bit is at "1" if the OUT2 contact is closed.
SI_ready	DINT	Each bit corresponds to the index of the security device which is in test or read status.
SI_off	DINT	Each bit corresponds to the index of the security device that is deactivated.
SI_error	DINT	Each bit corresponds to the index of the security device that is faulty.
Dfb_stat	INT	This is the state of the DFB; this variable allows the user to check the progress of the DFB.
Dfb_err	INT	This word gives the error type: This word gives the detected error type: ● 16#90; the reply sent by the monitor is invalid, ● 16#91: the DFB has been deleted by the user, ● 16#92: the exchange has stopped on a Timeout, the DFB cannot receive data.

Chapter 20
SAFETY_MONITOR_V2: DFB for AS-i Safety Monitor

Introduction

This chapter describes the SAFETY_MONITOR_V2 DFB for the AS-i Safety Monitor.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function description

The SAFETY_MONITOR_V2 DFB allows data processed by the safety monitor to be obtained. It cannot be used to control the AS-i bus or its blocks.

The DFB can manage up to 48 devices and supports either sorting according to OSSDs or the display of all devices.

It can be programmed in any program module (Main, SR or section).

It is dedicated to a single safety monitor.

Additional parameters EN and ENO can be configured.

Rules

For reasons of performance, we recommend running the SAFETY_MONITOR_V2 in the MAST task.

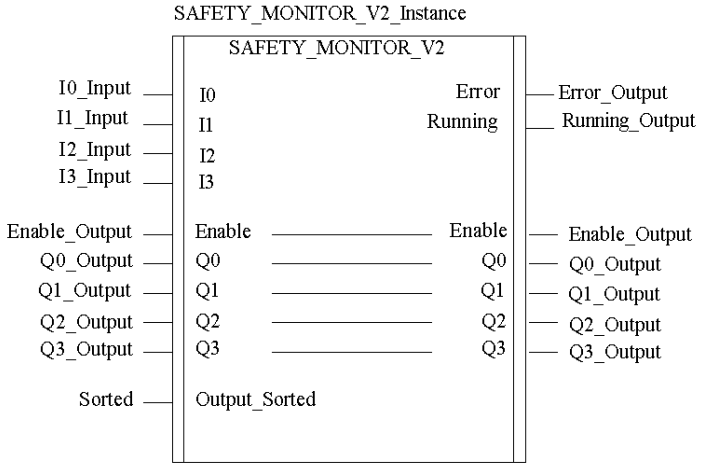
We strongly recommend programming only 1 instance of SAFETY_MONITOR_V2 in your project.

The following conditions are mandatory in order to run SAFETY_MONITOR_V2:

- The DFB can be programmed in the various tasks (with the exception of the event tasks) and in the sections of the project.
- The DFB must be called (the program element to which it is assigned must be run).
- The Enable input must be set to 1.
- The Output_Sorted input must be set accordingly (output sorted or not sorted).
- The AS-i Monitor must be configured in Unity Pro.

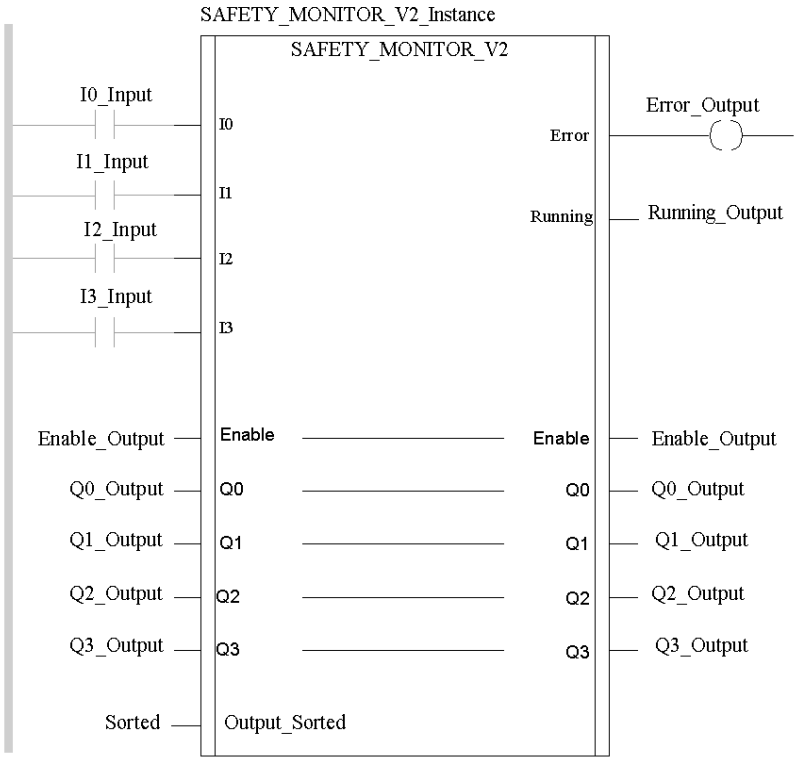
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SAFETY_MONITOR_V2_Instance (IO:=I0_Input, I1:=I1_Input,
I2:=I2_Input, I3:=I3_Input, Enable:=Enable_Output,
Q0:=Q0_Output, Q1:=Q1_Output, Q2:=Q2_Output, Q3:=Q3_Output,
Output_Sorted:=Sorted, Error=>Error_Output, Running=>Running_Output)
```

Representation in ST

Representation:

```
SAFETY_MONITOR_V2_Instance (IO:=I0_Input, I1:=I1_Input, I2:=I2_Input,
I3:=I3_Input, Enable:=Enable_Output,
Q0:=Q0_Output, Q1:=Q1_Output, Q2:=Q2_Output, Q3:=Q3_Output,
Output_Sorted:=Sorted, Error=>Error_Output, Running=>Running_Output);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Description
I0	EBOOL	Input variable 0
I1	EBOOL	Input variable 1
I2	EBOOL	Input variable 2
I3	EBOOL	Input variable 3
Output_Sorted	BOOL	• Bit = 1: Diagnostics sorted according to OSSDs (no preprocessing) • Bit = 0: Diagnostics of all devices

The following table describes the input/output parameters:

Parameter	Type	Description
Enable	EBOOL	• Bit = 1: Activate DFB (cold start) Setting this bit to 1 will execute the DFB, enabling information to be processed. Information can only be processed if Enable = 1 • Bit = 0: Deactivate DFB. The bit is set to 0 by the DFB at time-out.
Q0	EBOOL	Output variable 0
Q1	EBOOL	Output variable 1
Q2	EBOOL	Output variable 2
Q3	EBOOL	Output variable 3

The following table describes the output parameters:

Parameter	Type	Description
Error	EBOOL	Bit = 1: DFB or safety bus fault (At least 1 slave faulty). **Note:** ● DFB error (Enable = 0 and Dfb_err = 1)) A DFB error will invalidate safety project data. ● Bus error (Enable = 1 and Dfb_err = 0)) In the event of a device error in the safety project, the faulty addresses will be displayed in the public ARRAY variable Device.Device_error.
Running	EBOOL	Bit = 1: DFB running

Internal Public variables

The following table describes the internal public variables:

Name	Type	Description
Abort	EBOOL	If this bit is set to 0 in one cycle and to 1 in the next cycle, all exchanges between the CPU and the Safety Monitor will be aborted. The DFB will reinitialize and all internal data of the DFB will be set to 0.
Timeout	INT	Time-out during data exchange If the DFB does not receive a correct data set before this time elapses, ● transmission will be aborted, ● the DFB deactivated and ● the Error output will be set to 1 (Dfb_stat and Dfb_err are updated).
Moni_err	EBOOL	Bit = 1: Monitor error
Out_1	EBOOL	Bit = 1: 1. OSSD (OUT1) activated
Out_2	EBOOL	Bit = 1: 2. OSSD (OUT2) activated
Device.Device_ready	ARRAY[0..47] OF BOOL	Device ready Each index corresponds to the index of the safety device which is ready but still in test mode or waiting for another condition such as local acknowledgment, the activation of the Start button, etc.
Device.Device_off	ARRAY[0..47] OF BOOL	Device deactivated Each index corresponds to the index of the deactivated safety device. **Note:** Deactivated devices (including NOPs) set to FALSE are also transmitted as Device_off!

Name	Type	Description
Device.Device_error	ARRAY[0..47] OF BOOL	Device error Each index corresponds to the index of the faulty safety device.
Device.Device_noCom	ARRAY[0..47] OF BOOL	Device not communicating Each index corresponds to the index of the safety device which is not communicating on the AS-i bus.
Device.Device_allocation	ARRAY[0..47] OF INT	Device receiving instruction Each index corresponds to the index of the safety device. An integer value corresponding to the processing loop is assigned to the safety device. ● 1 = 1st OSSD ● 2 = 2nd OSSD ● 3 = Preprocessing (only appears in the event of an error; at all other times, 0 is displayed.) ● 4 = Both OSSDs **Note:** This information is only transmitted if Output_Sorted is set to 0!
Dfb_stat	STRING	DFB processing status in plain text
Dfb_err	INT	Indicates the following error types: ● 16#90: The response sent by the Monitor is invalid. ● 16#91: Data exchange has been aborted by the user. ● 16#92: The exchange has been aborted due to time-out, the DFB is unable to receive data.

Method of Operation

General

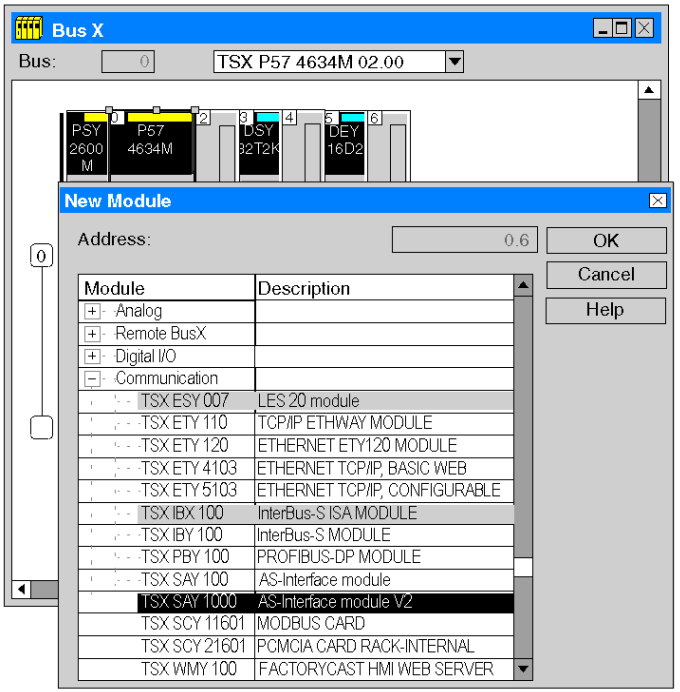
All the information used in the SAFETY_MONITOR_V2 is taken from language objects linked to the AS-i TSXSAY100 and AS-i TSXSAY1000 (V2) modules modules.

Method of Operation of the SAFETY_MONITOR_V2 DFB

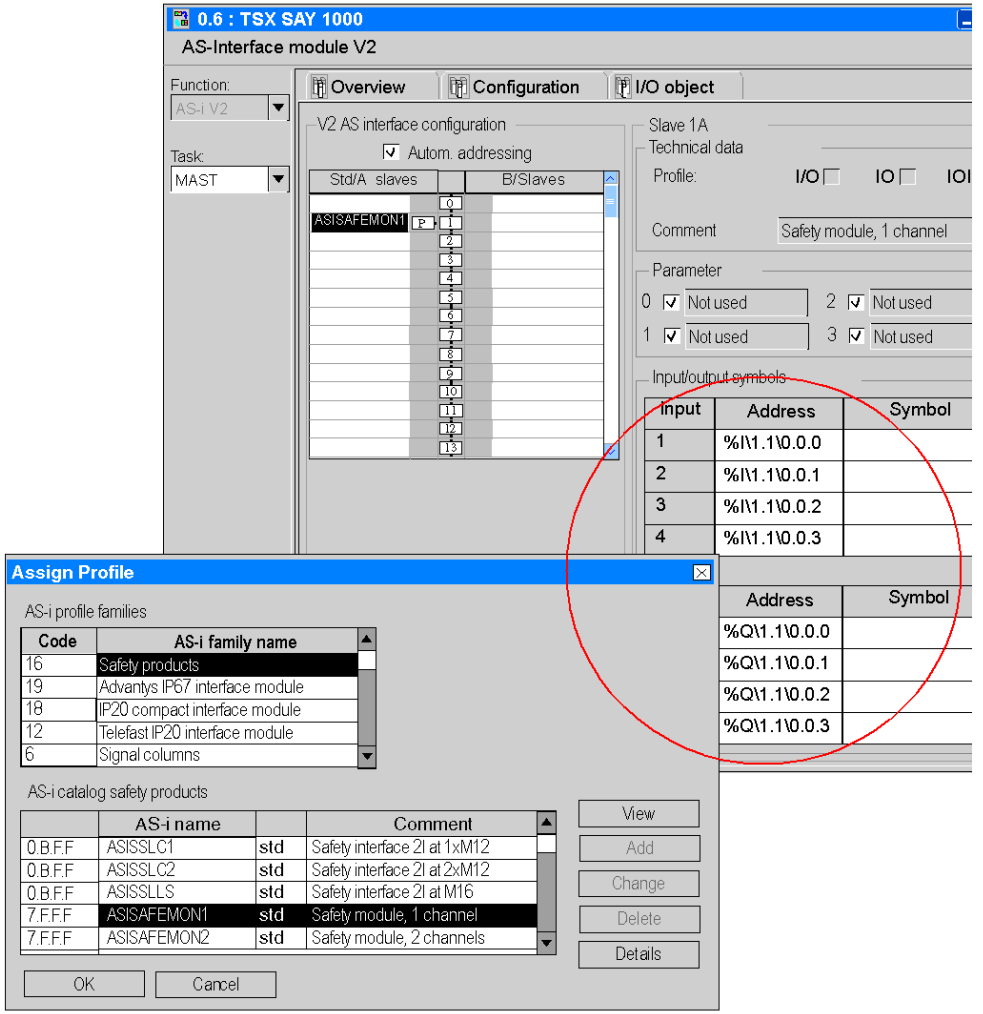
Phase	Description
1	The bus master polls the monitor to test it.
2	The bus master polls the monitor to prompt it to copy its status to the static memory.
3	The bus master analyzes the monitor data.
4	The data of all safety devices is restored.

Configuration

Configuring the AS-i Master Module

Step	Action
1	Add the TSXSAY100 or TSXSAY1000 (V2) module from the module library to the configuration. 
2	Double-click the module. **Result:** A configuration dialog box appears.

Step	Action
3	Enter the AS-i configuration settings. **Result:** As soon as you add the AS-i monitor, a list of addresses for the DFB inputs and outputs will appear.



Note: You can also view the AS-i bus configuration in the project browser under **Configuration**.

Sorting Outputs

CAUTION

MISINTERPRETATION OF DIAGNOSIS

The settings of the DFB for `Output_Sorted` must tally with the settings in the ASISWIN software in the **Monitor/Bus Information** dialog box, **Diagnostics/Service →Data Selection** tab.

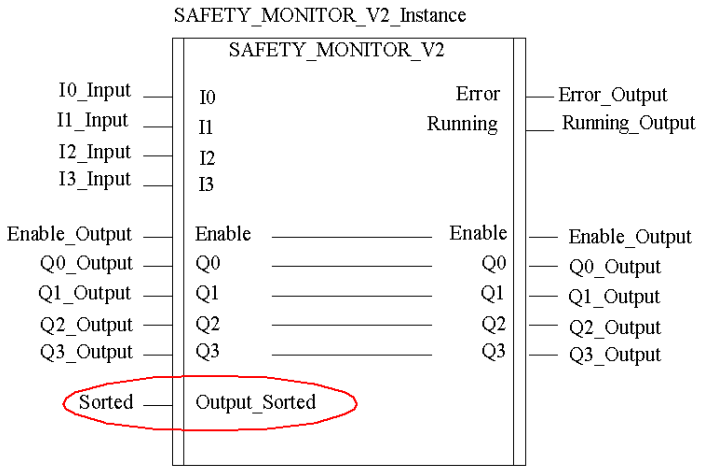
Otherwise, the diagnostics information will be misinterpreted in Unity Pro.

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

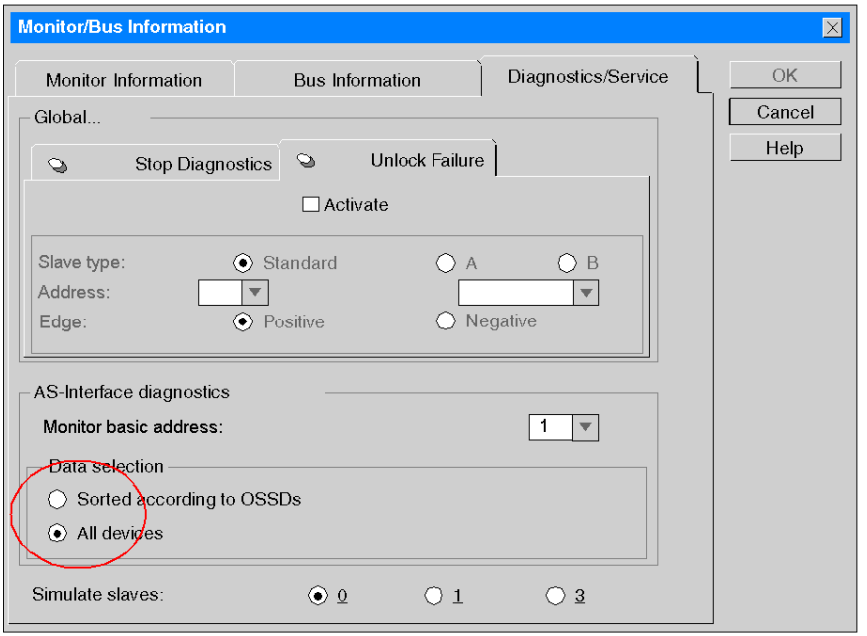
The following options are available:

`Output_Sorted`	**Meaning**
1	Diagnostics sorted according to OSSDs (no preprocessing)
0	Diagnostics of all devices

DFB setting in Unity Pro



Dialog box setting in ASISWIN



Chapter 21
UREGDFB: Registration of error messages in the diagnosis block

Introduction

This chapter describes the `UREGDFB` block.

What Is in This Chapter?

This chapter contains the following topics:

Description

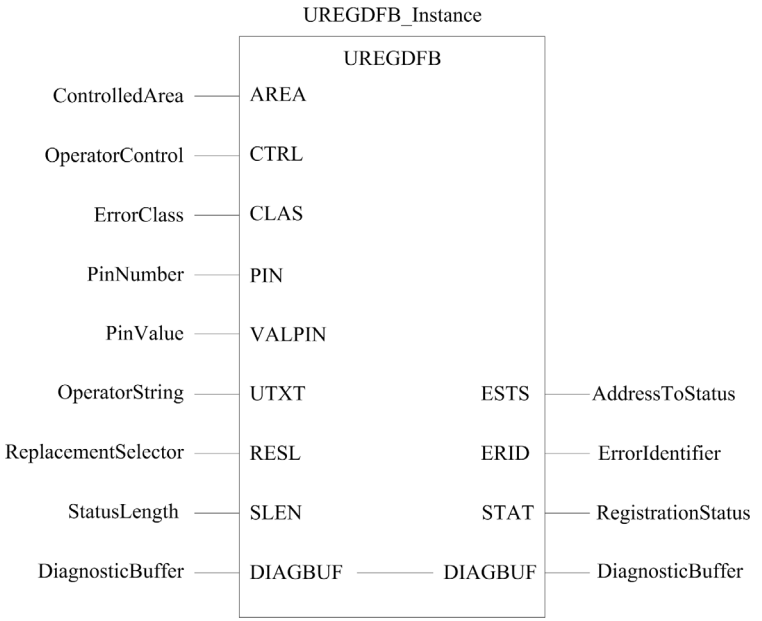
Function description

The UREGDFB procedure registers the error messages from the diagnostic DFB.

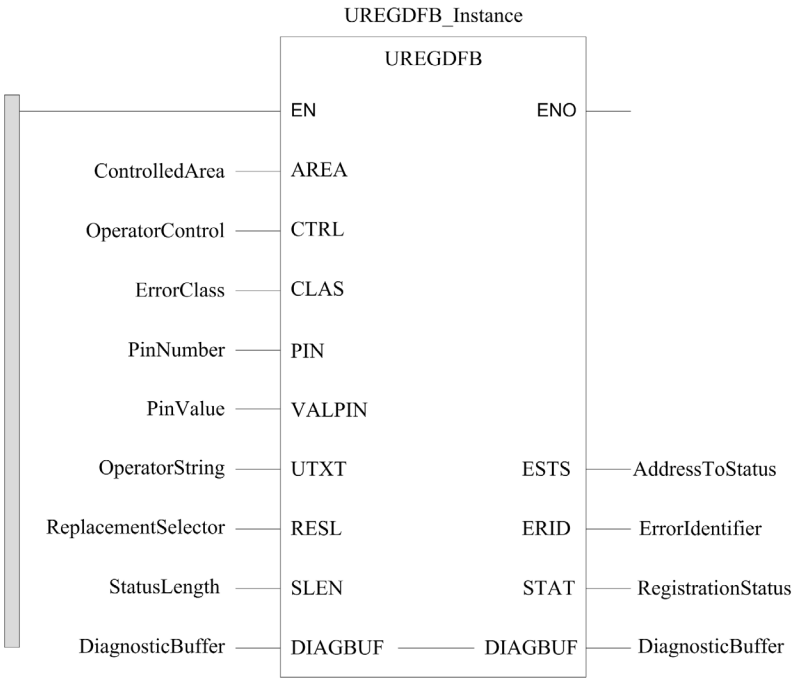
It also makes it possible for the UREGDFB diagnosis DFB to assign an error description to the error types. This makes it possible to distinguish between different error sub-types. The parameter RSEL can be used to define which default message is to be replaced by the user text.

EN and ENO can be configured as additional parameters.

Representation in FBD



Representation in LD



Representation in IL

```
LD ControlledArea
UREGDFB ErrorClass, StatusLenght,  OperatorControl, OperatorString,
        ReplacementSelector, PinNumber, PinValue,
        AddressToStatus, ErrorIdentifier,
        RegistrationStatus
```

Representation in ST

```
UREGDFB (ontrolledArea, ErrorClass, StatusLenght,
          ControlSwitch, UserText, ReplacementSelector,
          PinNumber, PinValue, AddressToStatus,
          ErrorIdentifier, RegistrationStatus);
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
AREA	INT	Machine area that will be monitored by the diagnostics EFB: 0 to 15
CLAS	INT	Error type: 16#004A
SLEN	INT	Status length: 0, 2 or 4 bytes: ● 0 = no scheduled status ● 2 = status is scheduled by a word ● 4 = status is scheduled by a double word
CTRL	BOOL	● 1 = User acknowledgement required ● 0 = User acknowledgement not required
UTEXT	STRING	User-defined error description (only active for RSEL = 1-4)
RSEL	INT	Replaced by UTEXT Possible values: ● 0: UTEXT is not used. ● 1: the instance comment is replaced by UTEXT ● 2: the instance name is replaced by UTEXT ● 3: the DFB type name is replaced by UTEXT ● 4: the pin name is replaced by UTEXT
PIN	INT	Number of monitored input
VALPIN	BOOL	the value expected at the monitored output

Description of output parameters:

Parameter	Data type	Description
ESTS	ANY	Diagnosis DFB status (declared in parameter OUT to be passed on to address and not value). Before this procedure is called it must be updated by the diagnosis DFB .
ERID	INT	Error recognition used by the function DEREG *(see page 47)* to de-register active errors. **Note:** The connection to other active errors is lost if the same error recognition variable is used for different errors.

STAT	INT	RegistrationStatus
		● If registration is successful: STAT = 0 and ERID is valid
		● If the registration fails: ERID is invalid and
		  ● STAT = 1: diagnostics buffer is not configured
		  ● STAT = 2: diagnostics buffer full.

Description of input/output parameters:

Parameter	Type	Comment
DIAGBUF	DWORD	Diagnostic buffer that contains the registration result

Example

Example

```
(* Register, if an error 1 occurs *)
IF ErrId_1 = 0 THEN              (* if error 1 was
                                    not registered *)
   UREGDFB(AREA:=N,                      (* N = 1..15 *)
           CLAS:=16#004A,        (* error type *)
           SLEN:=0,              (* length des ESTS fields,
                                    if used *)
           CTRL:=0,              (* 0: without,
                                    1: with acknowledgement *),
           UTXT:= Fehler_1_Text,    (* STRING *)
           RSEL:=1,              (* What should be replaced *)
           PIN:=0,               (* <> 0: monitor pin
                                    in the case of
                                    reasons for error:
                                    monitoring *)
           VALPIN:=0,            (* Expected value on
                                    monitored pin *)
           ESTS=>DfbStatus,      (* error status *)
           ERID=>ErrId_1,        (* error recognition *)
           STAT=>RegStatus);     (* Status *)
   IF RegStatus = 0 THEN         (* registration was
                                     successful *)
   ELSE                          (* if an error occurs *)
   END_IF;
(* update the system word %SW76 *)
   %SW76 := RegStatus;
END_IF;
(* De-registration, if  error 1 is possible *)
IF ErrId_1 <> 0 THEN
   DeRegStatus:=DEREG(ErrId_1);    (* De-registration
                                      of the error *)
   IF DeRegStatus = 0 THEN
      ErrId_1 := 0;                (* reset the
                                      Error identification:  *)
   END_IF;
(* Update of the system word %SW77 for operations which have been run *)
   %SW77:= DeRegStatus;
END_IF;
```

Chapter 22
USER_DIAG_ST_MODEL : Diagnostics DFB model

Subject of this Chapter

This chapter describes the USER_DIAG_ST_MODEL diagnostics DFB model.

What Is in This Chapter?

This chapter contains the following topics:

Description

Description of the Function

This DFB model detects the change to zero of the COND input.

To create a customized diagnostic DFB, you can use this model and modify it according to your needs.

It is unprotected. Its operation is described on the following pages.

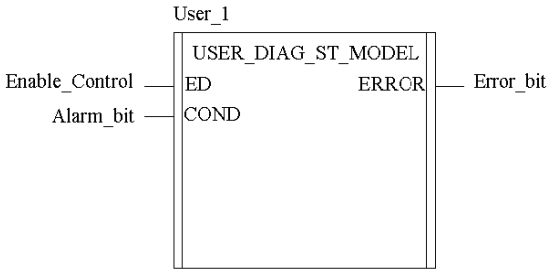
You can modify all the elements of this type of DFB to create your own diagnostic DFB.

Additional parameters EN and ENO can be configured.

NOTE: This diagnostic DFB cannot be used in a different user DFB. The model itself must be modified.

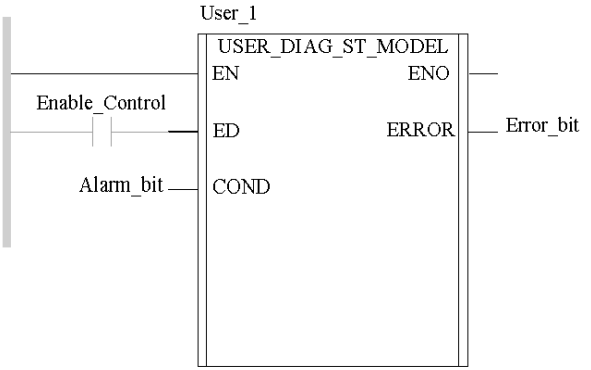
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL User_1 (ED: = Enable_Control, COND: = Alarm_bit,
ERROR => Error_bit)
```

Representation in ST

Representation:

```
User_1 (ED: = Enable_Control, COND: = Alarm_bit,
ERROR => Error_bit);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Description
ED	EBOOL	DFB activation bit. If ED = 0, theCOND input is not monitored.
COND	EBOOL	Bit monitored by the DFB: ● if COND = 1 the alarm is de-registered and the ERROR output changes to 0. The de-registration status (output of the DEREG *(see page 48)* function used in the code) is stored in the %SW77 system word. ● if COND = 0 and ERROR = 0 the alarm is saved and the ERROR output changes to 1. The status of the save is stored in the %SW76 system word (result of the save by the REGDFB *(see page 139)*) function. ● if COND = 0 and ERROR = 1 and there was a previous write attempt in the diagnostics buffer whereas it was full (BUFFULL=1), the alarm is saved. The status of the save is stored in the %SW76 system word (result of the save by the REGDFB *(see page 139)* function). If %SW76 = 0 this means the save was performed correctly and the BUFFULL indicator is set to 0.

The following table describes the output parameters:

Parameter	Type	Description
ERROR	BOOL	Output indicating that the alarm is detected by the DFB. ● If ERROR = 0 the last alarm has disappeared and has been handled, and there have been no new alarms. ● If ERROR = 1 an alarm is in progress, it may not have been saved in the PLC's diag buffer if the buffer was full when the alarm occurred (BUFFULL = 1). If the alarm was saved in the PLC's diag buffer, BUFFULL = 0.

The following table describes the public variables:

Parameter	Type	Description
AREA_NR	INT	Process zone monitored by the DFB: This zone is numbered 0 to 15 for which a zone has been determined in the Viewer.
OP_CTRL	EBOOL	If OP_CTRL = 1 the acknowledgement of the alarm on the Viewer is requested. If OP_CTRL = 0 the operator does not need to acknowledge the alarm message on the Viewer.

The following table describes the private variables:

Parameter	Type	Description
ERROR_ID	INT	Variable containing the error's identifier. This unique identifier is generated automatically by the system when the REGDFB *(see page 139)* function is called (use of the ERROR_ID variable as an output parameter of the REGDFB *(see page 139)* function). It is used on input in the DEREG *(see page 48)* function to deregister the error associated with this identifier.
STATUS	DINT	Double passworded as a parameter of the REGDFB *(see page 139)* function. When used, this double password must contain a value characteristic of the error. This status is displayed in the Viewer.
BUFFULL	EBOOL	BUFFULL = 1 indicates that the error was not correctly saved because the PLC's diag buffer was full.
PIN_NB	INT	Default input number which the REGDFB *(see page 139)* function must process. This number corresponds to the order number of the inputs for which the Diag property has been selected. In our model, the COND input is input 1 because it is the first diagnostic input. Indeed, the ED input is not used for the diagnostic.
PIN_VAL	BOOL	Value expected on the PIN_NB input. In this model, the alarm is detected when COND changes to zero whereas the expected value is 1.

NOTE: It is very important to fill in the comment of the DFB instance created, because this comment is displayed in the Diagnostic Viewer.

Detailed description

The ST code

The detailed operation of the diagnostic DFB model is given via the listing of the DFB code in ST language.

```
(* Initialization of PIN_VAL and PIN_NB *)
IF (COND = FALSE)
THEN
        (* error on 1st monitored input pin *)
        PIN_NB := 1;
        PIN_VAL := TRUE;
ELSE
        (* error not linked with a monitored input pin *)
        PIN_NB := 0;
        PIN_VAL := FALSE;
END_IF;

(* DFB not active *)
IF (NOT ED) THEN
    (* current error *)
    IF ERROR THEN
        (* deregistration *)
        %SW77:=DEREG(ERROR_ID);
        (* reset Error and Status *)
        RESET( ERROR);
        STATUS:=0;
    END_IF;
  (* Initialization of the full diagnostics buffer write indicator *)
    RESET (BUFFULL);
    RETURN;
END_IF;

(* Disappearance of the error *)
(* -------------------------- *)

(* Condition monitored correct *)
IF (COND) THEN
    (* current error *)
    IF ERROR THEN
        (* deregistration *)
        %SW77:=DEREG(ERROR_ID);
        (* reset Error *)
        RESET( ERROR);
    END_IF;
  (* Initialization of the full diagnostics buffer write indicator *)
    RESET (BUFFULL);
```

The ST code (cont.)

The rest of the section is as follows:

```
(* Appearance of the error *)
(* ----------------------- *)
ELSE    (* Condition monitored incorrect *)
    (* no current error *)
    IF NOT ERROR THEN
        (* registration *)
        REGDFB(AREA :=AREA_NR,    (* Machine zone monitored by the DFB *)
               CLAS :=16#0062,    (* Error class *)
               SLEN :=0,          (* Status length : 0, 2 or 4 bytes *)
               CTRL :=OP_CTRL,    (* Operator acknowledgment *)
               PIN  :=PIN_NB,     (* error Pin Number *)
               VALPIN := PIN_VAL, (* Expected Value *)
               ESTS => STATUS,    (* Status : not used in this DFB *)
               ERID =>ERROR_ID,   (* Error identifier *)
               STAT => %SW76);    (* Error registration report *)
        (* updating Error *)
        SET (ERROR);
        (* Processing of %SW76 report *)
        IF (%SW76 <> 0)
        THEN
                (* Error not saved but memorized *)
                SET( BUFFULL);
        END_IF;
    ELSE
        (* There is still an error *)
       (* Has there been an attempt to write on the full diagnostics buffer? *)
        IF (BUFFULL)
        THEN
                (* Retry saving *)
                REGDFB(AREA :=AREA_NR,     (* Machine zone monitored by the DFB *)
                       CLAS :=16#0062,     (* Error class *)
                       SLEN :=0,           (* Status length : 0, 2 or 4 bytes *)
                       CTRL :=OP_CTRL,     (* Operator acknowledgment *)
                       PIN  :=PIN_NB,   (* error Pin Number *)
                       VALPIN := PIN_VAL,  (* Expected Value *)
                       ESTS => STATUS,     (* Status : not used in this DFB *)
                       ERID =>ERROR_ID,    (* Error identifier *)
                       STAT => %SW76);     (* Error registration report *)
                (* Processing of %SW76 report *)
                IF (%SW76 = 0 )
                THEN
               (* OK it was possible to save the error reset everything or do nothing *)
                        RESET (BUFFULL);
                END_IF;
        END_IF;
     END_IF;
END_IF;
```

Chapter 23
ASI_DIA

Object of this Chapter

This chapter describes the ASI_DIA DFB.

What Is in This Chapter?

This chapter contains the following topics:

Description

Function Description

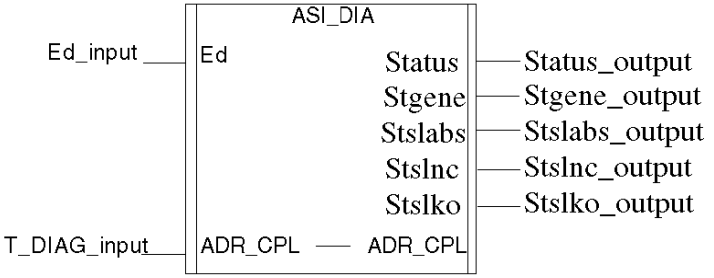
This DFB enables error occurences on the AS-Interface bus to be monitored:

- Module or bus error
- Missing slave(s)
- Not configured slave(s)
- Slave(s) error(s)

Representation in FBD

Representation:

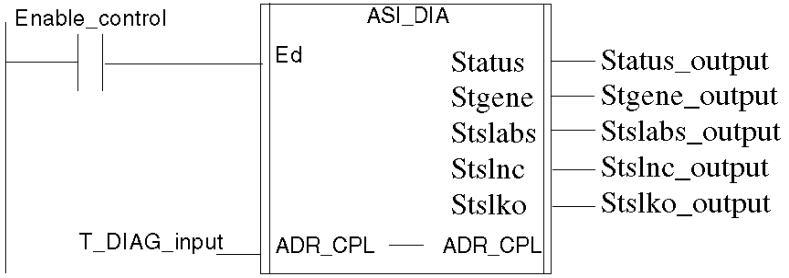
ASI_DIA instance name : ASI_1



Representation in LD

Representation:

ASI_DIA instance name : ASI_1



Representation in IL

Representation:

Cal ASI_1(Ed:=Enable_control, ADR_CPL:=T_DIAG_input, Status=>Status_output, Stgene=>Stegene_ouput, Stslabs=>Stslabs_ouput, Stslnc=>Stslnc_output, Stslko=>Stslko_output)

Representation in ST

Representation:

ASI_1(Ed:=Enable_control, ADR_CPL:=T DIAG_input, Status=>Status_output, Stgene=>Stegene_ouput, Stslabs=>Stslabs_ouput, Stslnc=>Stslnc_output, Stslko=>Stslko_output);

Description of the parameters

The following table describes the input parameter:

Name	Type	Description
ED	EBOOL	DFB activation bit if ED=0, the AS-Interface bus is not monitored **NOTE:** when ED=1, we can't open the TSXSAY1000 Debug screen from Unity module Editor

The following table describes the input/ output parameter:

Name	Type	Description
ADR_CPL	T_COM_ASI_DIAG	Adress of the AS-Interface Master Channel (IODDT)

⚠CAUTION

UNEXPECTED EQUIPMENT BEHAVIOR

T_DIAG_output must not be connected

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

The following table describes the ouput parameters:

Name	Type	Role	Description
STATUS	WORD	Error Type	The next bits indicate the type of detected error: ● Bit 0 =1: Module or bus error ● Bit 1 =1: Missing slave(s) ● Bit 2 =1: Not configured slave(s) ● Bit 3 =1: Slave(s) error(s)
STGENE	WORD	Module or bus error	Detail of the module or bus error: ● Bit 0 =1: The AS-Interface module does not give OK response to module identification request ● Bit 1 = 1: Slave with 0 address detected on the AS-Interface bus ● Bit 2 = 1: AS-Interface Power supply error ● Bit 3 = 1: OFFLINE phase active ● Bit 4 = 1: DATA_EXCHANGE mode inactive ● Bit 5 = 1: No slave presence on the bus ● Bit 6 = 1: Peripheral error
STSLABS	ARRAY [0..3] of WORD	List of absent slaves	STSLABS[0] slaves 0A to 15A: ● Bit 0: Not significant, always set to 0 ● Bit 1 = 1: The configured slave at adress 1A is absent, [...] ● Bit 15 = 1: The configured slave at adress 15A is absent STSLABS[1]: slaves 16A to 31A: ● Bit 0 = 1: The configured slave at adress 16A is absent, [...] ● Bit 15 = 1:The configured slave at adress 31A is absent STSLABS[2]: slaves 0B to 15B: ● Bit 0: Not significant, always set to 0 ● Bit 1 = 1: The configured slave at adress 1B is absent, [...] ● Bit 15 = 1: The configured slave at adress 15B is absent STSLABS[3]: slaves 16B to 31B: ● Bit 0 = 1: The configured slave at adress 16B is absent, [...] ● Bit 15 = 1: The configured slave at adress 31B is absent Default values = 0

STSLNC	ARRAY [0..3] of WORD	List of not configured slaves	STSLNC[0]: slaves 0A to 15A: ● Bit 0: Not significant, always set to 0 ● Bit 1 = 1: The detected slave at adress 1A is not configured, [...] ● Bit 15 = 1: The detected slave at adress 15A is not configured STSLNC[1]: slaves 16A to 31A: ● Bit 0 = 1: The detected slave at adress 16A is not configured, [...] ● Bit 15 = 1:The detected slave at adress 31A is not configured STSLNC[2]: slaves 0B to 15B: ● Bit 0: Not significant, always set to 0 ● Bit 1 = 1: The detected slave at adress 1B is not configured, [...] ● Bit 15 = 1: The detected slave at adress 15B is not configured STSLNC[3]: slaves 16B to 31B: ● Bit 0 = 1: The detected slave at adress 16B is not configured, [...] ● Bit 15 = 1: The detected slave at adress 31B is not configured Default values = 0
STSLKO	ARRAY [0..3] of WORD	List of slaves with error(s)	STSLKO[0]: slaves 0A to 15A: ● Bit 0: Not significant, always set to 0 ● Bit 1 = 1: Either an error is detected on the slave at adress 1A, or this slave is incorrectly configured, [...] ● Bit 15 = 1: Either an error is detected on the slave at adress 15A, or this slave is incorrectly configured. STSLKO[1]: slaves 16A to 31A: ● Bit 0 = 1: Either an error is detected on the slave at adress 16A, or this slave is incorrectly configured, [...] ● Bit 15 =1: Either an error is detected on the slave at adress 31A, or this slave is incorrectly configured. STSLKO[2]: slaves 0B to 15B: ● Bit 0: Not significant, always set to 0 ● Bit 1 = 1: Either an error is detected on the slave at adress 1B, or this slave is incorrectly configured, [...] ● Bit 15 = 1: Either an error is detected on the slave at adress 15B, or this slave is incorrectly configured. STSLKO[3]: slaves 16B to 31B: ● Bit 0 = 1: Either an error is detected on the slave at adress 16B, or this slave is incorrectly configured, [...] ● Bit 15 = 1: Either an error is detected on the slave at adress 31B, or this slave is incorrectly configured. Default values = 0

NOTE: EN *(see page 19)* input and ENO *(see page 19)*output can be configured as additionnal parameters.

Public variables

The following table describes the public variables:

Parameter	Data type	Meaning
AREA_NR	WORD	Automation area to be monitored. This WORD specifies which area will be monitored by the diagnostics EFB. It is advisable to assign the numbers according to the functional module. AREA_NR can be a value from 0 to 15. The standard value is 0. Example: Cutting: No.1 Milling: No. 2 Thread cutting: No. 3 In the example, AREA_NR must have the value 1, 2 ou 3, so that they can recognize the error affected area.
OPT_CTRL	BOOL	This bit specifies whether a diagnostic will request a user acknowledgement. 0: no user acknowledgement required 1: user acknowledgement required The standard value is 0.

Description of T_COM_ASI_DIAG IODDT

All the information in the DFB results from the exploitation from T_COM_ASI_DIAG IODDT. T_COM_ASI_DIAG supports READ_STS, READ_TOPO_ADDR and is usable by DFB.

The following table describes the T_COM_ASI_DIAG IODDT:

Standard symbol	Type	Access	Meaning	Number	Rank	Bit	EXCH
CH_ERROR	BOOL	R	Channel error	%I	ERR		IMP
FLT_SLAVES_0A_15A	INT	R	Slave error 0A to 15A	%IW	0		IMP
FLT_SLAVES_16A_31A	INT	R	Slave error 16A to 31A	%IW	1		IMP
FLT_SLAVES_0B_15B	INT	R	Slave error 0B to 15B	%IW	2		IMP
FLT_SLAVES_16B_31B	INT	R	Slave error 16B to 31B	%IW	3		IMP
STS_IN_PROGR	BOOL	R	Status parameter read in progress	%MW	0	X0	SYS
STS_ERR	BOOL	R	Error while reading channel status	%MW	1	X0	SYS
CH_FLT	INT	R	Channel errors	%MW	2		STS
SLAVE_FLT	BOOL	R	1 in case of one error slave	%MW	2	X1	STS
ASI_CONF_FLT	BOOL	R	Physical configuration different from logical configuration	%MW	2	X3	STS
INTERNAL_FLT	BOOL	R	Internal error : channel inoperative	%MW	2	X4	STS
CONF_FLT	BOOL	R	Hardware or software configuration error	%MW	2	X5	STS
COM_FLT	BOOL	R	Bus communication error	%MW	2	X6	STS

SLAVE_0_PRESENT	BOOL	R	Slave 0 present on the bus	%MW	3	X1	STS
ASI_SUPPLY_FLT	BOOL	R	AS-Interface supply error	%MW	3	X6	STS
OFFLINE_MODE_ACTIVE	BOOL	R	Offline mode active	%MW	3	X7	STS
DATA_EXCHANGE_OFF	BOOL	R	Data exchange inactive	%MW	3	X8	STS
PERIPH FAULT	BOOL	R	Peripheral error on a bus device	%MW	3	X9	STS
LDS_0A_15A	INT	R	List of detected slaves 0A to 15A	%MW	4		STS
LDS_16A_31A	INT	R	List of detected slaves 16A to 31A	%MW	5		STS
LDS_0B_15B	INT	R	List of detected slaves 0B to 15B	%MW	6		STS
LDS_16B_31B	INT	R	List of detected slaves 16B to 31B	%MW	7		STS
MASTER_TYPE	INT	R	AS-Interface Master Type	%KW	0		CONST
LPS_0A_15A	INT	R	List of projected (configured) slaves 0A to 15A	%KW	1		CONST
LPS_16A_31A	INT	R	List of projected (configured) slaves 16A to 31A	%KW	2		CONST
LPS_0B_15B	INT	R	List of projected (configured) slaves 0B to 15B	%KW	3		CONST
LPS_16B_31B	INT	R	List of projected (configured) slaves 16B to 31B	%KW	4		CONST

NOTE: Please note that, depending on your hardware platform, using ASI_DIA DFB does not give the same diagnosis:

- Premium/ Atrium platform: ASI_DIA operates in the same way as in PL7: no indication about error slave number in case of Peripheral errors on one slave.
- Modicon M340 platform: indication of error slave number in case of Peripheral errors on one slave.

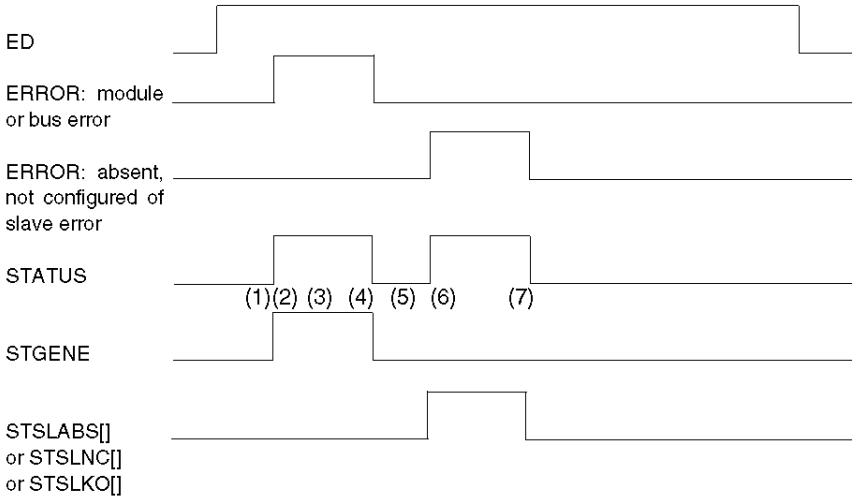
How the ASI_DIA Function Block works

General

All information used in the ASI_DIA DFB is obtained from the language object associated with the AS-Interface module.

Timing Diagram

The following diagram is an example, which shows how the ASI_DIA DFB works :



Description of operations

The following table describes the different phases shown in the timing diagram.

Phase	Description
1	A Module or Bus error is registered by the DFB in case of a break in the AS-Interface supply, bit 0 in STATUS and bit 2 in STGENE are set to 1.
2	A slave with 0 address is detected on the AS-Interface bus, bit 1 in STGENE is set to 1.
3	AS-Interface power is restored, but the Module or Bus error is not erased because a slave with 0 address is still detected on the AS-Interface bus.
4	The slave with 0 address is no longer detected on the AS-Interface bus, the error has disappeared. The STATUS and STGENE words are set to 0.
5	An Absent Slave(s), Not configured slave(s) or Error Slave(s) error is built into the STATUS word (bit = 1, 2 or 3) and bit 10 of STSLABS[0], STSLNC[0] or STSLKO[0] or STSLKO[0] is set to 1 indicating that AS-Interface slave 10 is in error.
6	AS-Interface slave 14 is disconnected, only bit 14 of STSLABS[0], STSLNC[0] or STSLKO[0] is set to 1.
7	AS-Interface slaves 10 and 14 are again present on the AS-Interface Bus. Bit 1 of STATUS is set to 0 and STSLABS[0], STSLNC[0] or STSLKO[0] are set to 0.

Appendices

Appendix A
EFB Error Codes and Values

Introduction

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

What Is in This Chapter?

This chapter contains the following topics:

Tables of Error Codes for the Diagnostics Library

Introduction

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

Diagnostics

Table of error codes and errors values created for EFBs of the `Diagnostics` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
ONLEVT	E_EFB_ONLEVT	T/F	-30196	16#8A0C	Error of EFB ONLEVT ENO states ● True = Error registration OK ● False = Error registration failed

Common Floating Point Errors

Introduction

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

Common Floating Point Errors

Table of common floating point errors

Error codes	Error value in Dec	Error value in Hex	Error description
FP_ERROR	-30150	16#8A3A	Base value (not apearing as an error value)
E_FP_STATUS_FAILED_IE	-30151	16#8A39	Illegal floating point operation
E_FP_STATUS_FAILED_DE	-30152	16#8A38	Operand is denormalized - not a valid REAL number
E_FP_STATUS_FAILED_ZE	-30154	16#8A36	Illegal divide by zero
E_FP_STATUS_FAILED_ZE_IE	-30155	16#8A35	Illegal floating point operation / Divide by zero
E_FP_STATUS_FAILED_OE	-30158	16#8A32	Floating point overflow
E_FP_STATUS_FAILED_OE_IE	-30159	16#8A31	Illegal floating point operation / Overflow
E_FP_STATUS_FAILED_OE_ZE	-30162	16#8A2E	Floating point overflow / Divide by zero
E_FP_STATUS_FAILED_OE_ZE_IE	-30163	16#8A2D	Illegal floating point operation / Overflow / Divide by zero
E_FP_NOT_COMPARABLE	-30166	16#8A2A	Internal error

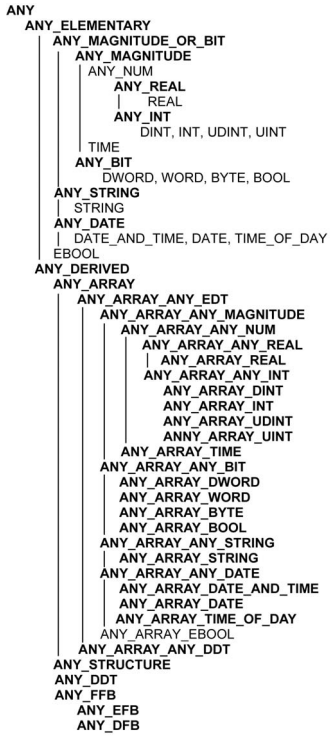
Glossary

A

ANY

There is a hierarchy among the various data types. In the DFBs, it is sometimes possible to declare variables that can contain several types of values. In that case we use `ANY_xxx` types.

The figure below describes this hierarchical structure:



ARRAY

An `ARRAY` is a table containing elements of a single type.

The syntax is as follows: `ARRAY [<limits>] OF <Type>`

Example:

`ARRAY [1..2] OF BOOL` is a one-dimensional table with two elements of type `BOOL`.

`ARRAY [1..10, 1..20] OF INT` is a two-dimensional table with 10x20 elements of type `INT`.

B

BOOL

BOOL is the abbreviation for the Boolean type. This is the basic data type in computing. A BOOL variable can have either of the following two values: 0 (FALSE) or 1 (TRUE).

A bit extracted from a word is of type BOOL, for example: %MW10.4.

BYTE

When 8 bits are grouped together, they are called a BYTE. You can enter a BYTE either in binary mode or in base 8.

The BYTE type is encoded in an 8 bit format which, in hexadecimal format, ranges from 16#00 to 16#FF.

D

DINT

DINT is the abbreviation of Double INTeger (encoded in 32 bits).

The upper/lower limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

-2147483648, 2147483647, 16#FFFFFFFF.

DWORD

DWORD is the abbreviation of Double Word.

The DWORD type is encoded in a 32 bit format.

This table shows the upper/lower limits of each of the bases that can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFFFFFF
Octal	8#0	8#37777777777
Binary	2#0	2#11111111111111111111111111111111

Examples of representation:

Data	Representation in one of the bases
00000000000010101101110011011110	16#ADCDE
00000000000000010000000000000000	8#200000
00000000000010101011110011011110	2#10101011110011011110

E

EBOOL

EBOOL is the abbreviation of Extended BOOLean. An EBOOL type has a value (0 (FALSE) or 1 (TRUE)), but also rising or falling edges and forcing functions.

An EBOOL variable occupies one byte in memory.

The byte contains the following information:

- one bit for the value;
- one bit for the history (whenever the object changes state, the value is copied to the history bit);
- one bit for forcing (equal to 0 if the object is not forced, or 1 if the bit is forced).

The default value of each bit is 0 (FALSE).

EN

EN stands for **EN**able; it is an optional block input. When the EN input is enabled, an ENO output is set automatically.

If EN = 0, the block is not enabled; its internal program is not executed, and ENO is set to 0.

If EN = 1, the block's internal program is run and ENO is set to 1. If an error occurs, ENO is set to 0.

If the EN input is not connected, it is set automatically to 1.

ENO

ENO stands for **E**rror **NO**tification; this is the output associated with the optional input EN.

If ENO is set to 0 (because EN = 0 or in case of an execution error):

- the status of the function block outputs remains the same as it was during the previous scanning cycle that executed correctly;
- the output(s) of the function, as well as the procedures, are set to "0".

I

INT

INT is the abbreviation of single INTeger (encoded in 16 bits).

The upper/lower limits are as follows: -(2 to the power of 15) to (2 to the power of 15) - 1.

Example:

```
-32768, 32767, 2#1111110001001001, 16#9FA4.
```

S

STRING

A STRING variable is a series of ASCII characters. The maximum length of a string is 65,534 characters.

T

TIME

The TIME type expresses a time in milliseconds. Encoded in 32 bits, this type can be used to obtain times from 0 to 2^{32}-1 milliseconds.

The TIME type has the following units: days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms). A literal value of type TIME is represented by a combination of the preceding types prefixed with T#, t#, TIME# or time#.

Examples: T#25h15m, t#14,7S, TIME#5d10h23m45s3ms

U

UDINT

UDINT is the abbreviation of Unsigned Double INTeger (encoded in 32 bits). The upper/lower limits are as follows: 0 to (2 to the power of 32) - 1.

Example:

0, 4294967295, 2#11111111111111111111111111111111, 8#37777777777, 16#FFFFFFFF.

W

WORD

The type WORD is encoded in a 16 bit format and is used to perform processing on series of bits.

This table shows the upper/lower limits of each of the bases that can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Examples of representation

Data	Representation in one of the bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index