



Diagnostics in User Program with S7-1500

STEP 7 (TIA Portal), S7-1500

<https://support.industry.siemens.com/cs/ww/en/view/98210758>

Siemens
Industry
Online
Support



Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://support.industry.siemens.com>) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place. For additional information on industrial security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: <https://www.siemens.com/industrialsecurity>.

Table of Contents

Legal information	2
1 Task	4
1.1 Overview	4
1.2 Requirements / Scenarios	5
2 Solution	6
2.1 Solution overview	6
2.2 Hardware and software components	8
2.2.1 Validity	8
2.2.2 Components used	8
3 Basics	10
4 Function Mechanisms	11
4.1 General overview	11
4.2 Global data block DiagDataDB [DB6]	13
4.3 Function block DiagStartupFB [FB100]	17
4.4 Function block DiagMainFB [FB1]	19
4.5 Function block DiagDevicesFB [FB2]	21
4.6 Function block DiagSignalFB [FB4]	24
4.7 Function block DiagUsrMsgFB [FB5]	26
4.8 Function block DiagPNIOFB [FB3]	28
5 Configuring the HMI Screens	30
5.1 Configuring a device in the plant overview	30
5.2 Configuring a device in the detail view	33
6 Installation and Commissioning	35
6.1 Installing the hardware	35
6.2 IP addresses and device names	35
6.3 Installing the software (download)	36
6.4 Assigning PROFINET device names	37
6.5 Loading the project	38
6.6 Integrating the application into an existing project	39
6.6.1 Configuring the diagnostic settings	39
6.6.2 Integration of the PLC elements	39
6.6.3 Integration of the HMI elements	42
7 Operating the Application	43
7.1 Overview	43
7.2 Diagnostics on the operator panel	44
7.2.1 Diagnostics "Value status on AI8"	44
7.2.2 Diagnostics "Wire break on the DI module of the ET 200SP"	45
7.2.3 Diagnostics "Supply voltage missing on the DI module of the ET 200MP"	46
7.2.4 Diagnostics "Overtemperature on G120 drive"	47
7.3 Diagnostics in the TIA Portal	48
7.3.1 Diagnostics "Value status on AI8"	48
7.3.2 Diagnostics "Wire break on the DI module of the ET 200SP"	49
8 Related Literature	51
9 History	51

1 Task

1.1 Overview

Introduction

The diagnostics of devices, modules and networks play an ever-increasing role in automation technology. By diagnosing with a user program, faulty modules can be detected. This enables you to also program responses to diagnostic messages, such as, for example, that your plant is stopped in the event of certain diagnostic messages.

Note

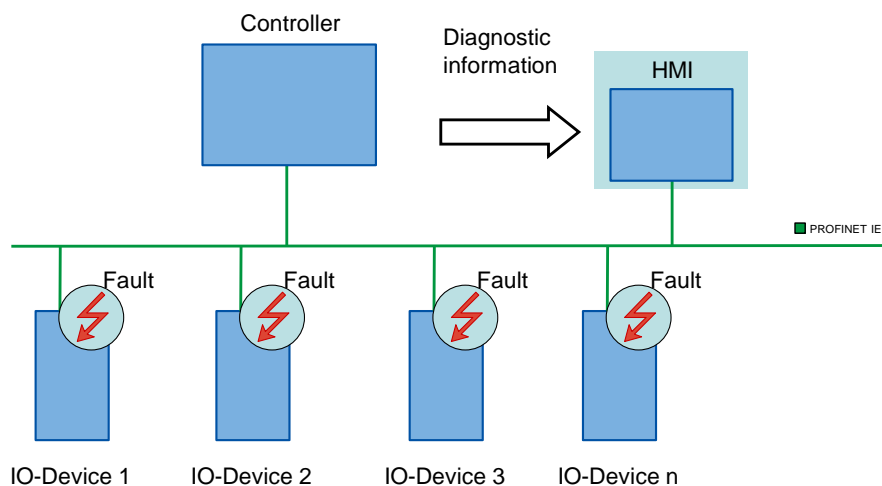
When diagnostic information is displayed on different visualization devices and not used in the user program, it is generally recommended to use the integrated system diagnostics of the S7-1500 controllers (see [System Diagnostics with S7-1500 and TIA Portal](#)).

Please note that the integrated system diagnostics also work in the “STOP” mode of the S7-1500 controller. This is not possible with the manual diagnostics in the user program.

Overview of the automation task

The figure below provides an overview of the automation task.

Figure 1-1



Description of the automation task

The automation tasks consists of monitoring a PROFINET IO distributed system with different network components. It is to guarantee the possibility of individual diagnostics of the devices and components. The user program takes on the system diagnostics of the plant with the help of the integrated diagnostic instructions. The diagnostic information detected is displayed on an operator panel.

1.2 Requirements / Scenarios

Requirements of the automation task

With the application, the programmer is to get an introduction to system diagnostics via the user program.

Apart from the integrated system diagnostics, a series of instructions for system diagnostics is provided in the user program with the TIA Portal. Based on an example, this application is to help you to describe the functions and the use of the diagnostic instructions to the plant programmers.

Scenarios

The example for the use of the diagnostic instructions is divided into different scenarios.

- User-defined alarm, based on the evaluation of the value status (quality information) on the AI module of the CPU S7-1516
- Wire break on the DI module of the ET 200SP
- Missing supply voltage on the DI module of the ET 200MP
- Overtemperature on the SINAMICS G120 drive

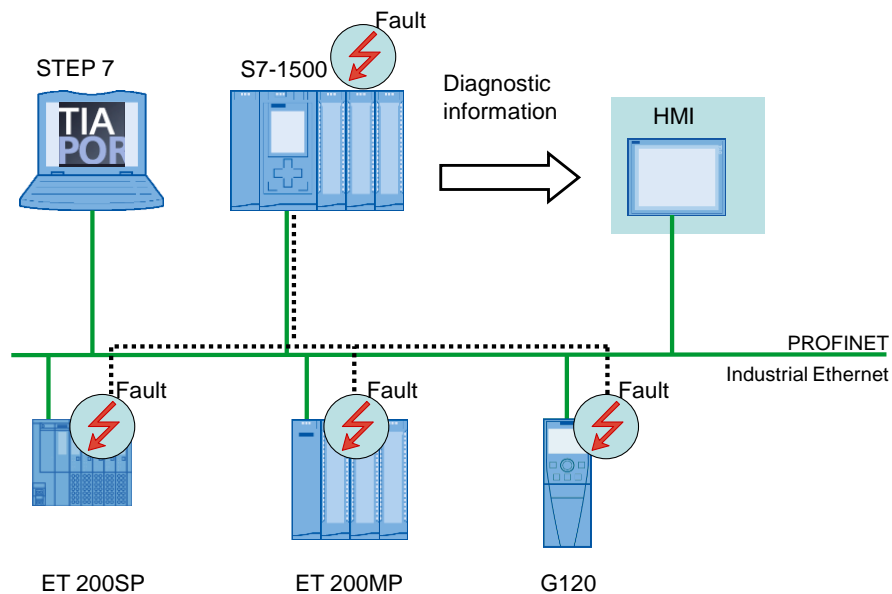
2 Solution

2.1 Solution overview

Schematic layout

The following figure displays the most important components of the solution:

Figure 2-1



Setup

The different distributed I/O devices are connected with a S7-1500 controller via PROFINET. The devices detect the faults on their modules and send the diagnostic data to the respective controller. The controller evaluates this diagnostic information with the help of the diagnostic instructions in the user program. The operator panel displays evaluated information graphically in a plant overview or in the respective device view.

Note

The diagnostics described here in the user program can also be used for PROFIBUS systems.

Delimitation

- This application does not include a description of the integrated system diagnostic.
- This application does not contain a complete discussion of all diagnostic possibilities with the user program. The extension of the present code by the user is therefore possible and necessary.
- This application does not include a detailed description of the diagnostic instructions.

Note

A more detailed description on the diagnostic instructions is available in the TIA Portal online help or in the [System Manual SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1](#).

- This application does not include a description of the diagnostic instructions of S7-1200 controllers.

Basic knowledge of these topics is assumed.

Required knowledge

Basic knowledge of the following issues is assumed:

- SIMATIC STEP 7 (TIA Portal)
- SIMATIC WinCC
- STEP 7 block architecture and programming
- PROFINET-IO

2.2 Hardware and software components

2.2.1 Validity

This application was tested with

- SIMATIC STEP 7 Professional V15.1 Update 1(TIA Portal)
- SIMATIC S7-1500 V2.6

2.2.2 Components used

Hardware components

Table 2-1

Component	No.	Article number	Note
CPU 1516-3 PN/DP	1	6ES7516-3AN01-0AB0	Alternatively, a different CPU S7-1500 can be used.
Memory card, 12 Mbytes	1	6ES7954-8LE02-0AA0	
Digital input, DI 32x24VDC HF	1	6ES7521-1BL00-0AB0	Diagnostics can be configured
Front connector, screw terminal, 40-pin	2	6ES7592-1AM00-0XB0	
Digital output, DQ 32x24VDC/0.5A ST	1	6ES7522-1BL00-0AB0	Diagnostics can be configured
Analog input, AI 8xU/I/RTD/TC ST	1	6ES7531-7KF00-0AB0	Diagnostics can be configured
Front connector, push-in technology, 40-pin	1	6ES7592-1BM00-0XB0	
IM 155-5 PN ST	1	6ES7155-5AA00-0AB0	ET 200MP
Digital input, DI 32x24VDC HF	1	6ES7521-1BL00-0AB0	Diagnostics can be configured
Front connector, screw terminal, 40-pin	2	6ES7592-1AM00-0XB0	
Digital output, DQ 32x24VDC/0.5A ST	1	6ES7522-1BL00-0AB0	Diagnostics can be configured
Analog input, AI 8xU/I/RTD/TC ST	1	6ES7531-7KF00-0AB0	Diagnostics can be configured
Front connector, push-in technology, 40-pin	1	6ES7592-1BM00-0XB0	
IM 155-6 PN ST with server module, with bus adapter 2xRJ45	1	6ES7155-6AA00-0BN0	ET 200SP
DI 16x24VDC ST	1	6ES7131-6BH00-0BA0	Diagnostics can be configured
BU type A0, 16 push-in, 2 infeed term. separate (digital/analog, max. 24VDC/10A)	1	6ES7193-6BP00-0DA0	
DQ 16x24VDC/0.5A ST	1	6ES7132-6BH00-0BA0	Diagnostics can be configured

2 Solution

2.2 Hardware and software components

Component	No.	Article number	Note
BU type A0, 16 push-in, 2 infeed term. jumpered (digital/analog, 24VDC/10A)	2	6ES7193-6BP00-0BA0	
AQ 4xU/I ST	1	6ES7135-6HD00-0BA1	Diagnostics can be configured
CU240E-2 PN-F	1	6SL3244-0BB13-1FA0	SINAMICS G120 with FW 4.6
PM340	1	6SL3110-1SB11-0AA0	
IOP	1	6SL3255-0AA00-4JA0	(optional)
TP1200 Comfort	1	6AV2124-0MC01-0AX0	
SIMATIC Field PG M4	1	6ES7716-.....0...	

Standard software components

Table 2-2

Component	No.	Article number	Note
SIMATIC STEP 7 Professional V15.1 Update 1	1	6ES7822-1..05-..	
SIMATIC WinCC Advanced V15.1 Update 1	1	6AV210-.....5-0	
SINAMICS Startdrive V15.1	1	6SL3072-4FA02-0XA0	can be downloaded for free see [3]

Sample files and projects

The following list includes all files and projects that are used in this example.

Table 2-3

Component	Note
98210758_User_defined_diagnostics_DOKU_v11_en.pdf	This document.
98210758_User_defined_diagnostics_CODE_v11.zip	This zip file includes the STEP 7 project.

3 Basics

Basics on system diagnostics

In the SIMATIC environment the diagnostics of devices and modules are summarized by the term “system diagnostics”. The monitoring functions are automatically derived from the hardware configuration.

All the SIMATIC products refer to integrated diagnostic functions with which you can detect and repair faults. The components automatically report a possible operational fault and supply additional detailed information. Plant-wide diagnostics can minimize downtimes.

Diagnostic instructions

There is a cross-vendor structure for data records with diagnostic information. For the determination of the system diagnostics of a device in the user program, the following instructions are available in STEP 7.

Table 3-1

Instruction	Description
RD_SINFO	Read out start information of the current OB
RT_INFO	Read out runtime statistics (not part of this application)
LED	Read LED status
Get_IM_Data	Read identification and maintenance data
GET_NAME	Read out name of a module
GetStationInfo	Read out information of a IO device
DeviceStates	Read module state information in an IO system
ModuleStates	Read module status information of a module
GEN_DIAG	Generate diagnostic information (not part of this application)
GET_DIAG	Read diagnostic information (not part of this application)

Reporting instructions

The following instructions are available in STEP 7 for creating messages in the user program.

Table 3-2

Instruction	Description
Program_Alarm	Create program message with associated values
Get_AlarmState	Output message state
Gen_UsrMsg	Create user diagnostic messages

Note

For more detailed information about the instructions, please refer to the TIA Portal Online Help.

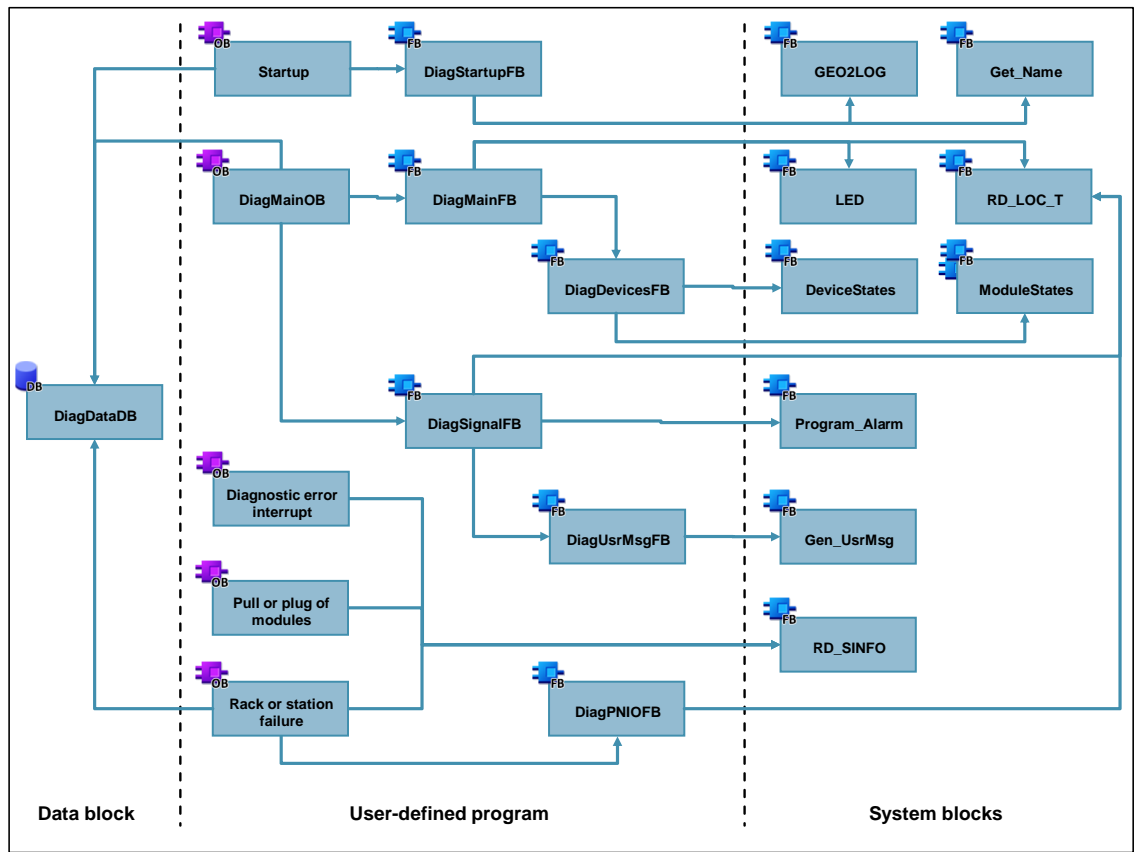
4 Function Mechanisms

4.1 General overview

Program overview

The following figure shows the program structure of the whole STEP 7 project.

Figure 4-1



In the following table, all blocks of the user program are described.

4 Function Mechanisms

4.1 General overview

The user program consists of the following elements:

Table 4-1

Symbolic name	Description
DiagDataDB	The global data block includes the data structures: <ul style="list-style-type: none">• IO system• Controller• Devices with their modules• Error buffer
Startup [OB100]	The startup OB calls the "DiagStartupFB" function block and transfers the structures of the global "DiagDataDB" block to the block.
DiagStartupFB [FB100]	The function block determines the hardware ID of the modules and the individual modules from the slot with the "GEO2LOG" instruction. The "Get_Name" instruction reads out the device name.
DiagMainOB [OB123]	The cycle OB calls the "DiagMainFB" function block and transfers the structures of the global "DiagDataDB" block to the block. In order to evaluate a digital signal, the block calls the "DiagSignalFB" function block.
DiagMainFB [FB1]	The function block evaluates the status of the error LED of the controller with the "LED" instruction. In the event of a fault, it calls the "DiagDevicesFB" function block.
DiagDevicesFB [FB2]	The function block reads the status of the PROFINET IO devices with the "DeviceStates" instruction and therefore detects the faulty IO devices. The "ModuleStates" instruction reads out the module status of the faulty IO devices and of the local modules of the controller.
DiagSignalFB [FB4]	In the block, a program message is created with "Program_Alarm". The block calls the "DiagUsrMsgFB" function block.
DiagUsrMsgFB [FB5]	The function block writes a user diagnostic message into the diagnostic buffer of the controller with the "Gen_UsrMsg" instruction.
Diagnostic error interrupt [OB82]	The diagnostic interrupt OB calls the "RD_SINFO" instruction. The OB is called by the operating system when a diagnostic-capable module detects a change of its diagnostic state.
Pull or plug of modules [OB83]	The pull/plug OB calls the "RD_SINFO" instruction. The OB is called by the operating system when a configured module or submodule of the distributed IO is pulled or plugged in.
Rack or station failure [OB86]	Der module rack failure OB calls the "RD_SINFO" instruction and the "DiagPNIOFB" function block. "RD_SINFO" reads the start information of the OB and transfers it to the "DiagPNIOFB" function block.
DiagPNIOFB [FB3]	The function block checks whether there is an error in the PROFINET IO system.

4.2 Global data block DiagDataDB [DB6]

Overview

The following figure shows the structure of the global “DiagDataDB” data block.

Figure 4-2

	Name	Datentyp	Startwert
1	Static		
2	IO_System	*IOSystemStruct*	
3	PLC	*PLCStruct*	
4	Devices	*DeviceStruct*	
5	ErrorList	*ErrorListStruct*	
6	AlarmsDeviceStates	Word	16#0
7	AlarmsModuleStates	Word	16#0

The data block is used as interface of the application for the outside. The DB “DiagDataDB” saves the hardware ID and the error status of the following components:

- PROFINET IO system
- Controller
- Devices
- Modules

In addition it saves the last ten errors in a buffer.

The individual tags of the components were summarized in the following structure for a better overview.

IOSystem [IOSystemStruct]

This structure includes the tags of the IO system.

Table 4-2

Tag name	Data type	Default value	Meaning
Laddr	HW_IOSYSTEM	-	Hardware ID of the IO system (system data type)
ErrorState	Bool	-	Error status of the IO system

PLC [PLCStruct]

This structure includes the tags of the controller and of the local modules.

Table 4-3

Tag name	Data type	Default value	Meaning
ModuleNumHigh	Int	4	Highest Slot number of local modules
DeviceIdPLC	HW_DEVICE	32	Hardware ID of the controller for module status (system data type)
ErrorState	Bool	-	Error status of the controller
SlotLaddr	Array[0..4] of HW_IO	-	Hardware ID of the local module (system data type) The array index corresponds to the slot number.
SlotErrorState	Array[0..4] of Bool	-	Error status of the local modules The array index corresponds to the slot number.

Note

If you change the default value of „ModuleNumHigh“, you also have to adjust the two arrays.

Devices [DeviceStruct]

This structure includes the tags of the devices.

Table 4-4

Tag name	Data type	Default value	Meaning
DeviceNumHigh	Int	3	Highest device number
ModuleNumHigh	Int	4	Highest Slot number of modules
ErrorLED	UInt	2	Identification number of the ERROR LED
ErrorLEDFlash	Int	4	LED status color 1 flashes
ProblemMode	UInt	5	Selection of status information to be read
Device	Array[0..3] of DeviceSingleStruct	-	See DeviceSingleStruct The array index corresponds to the device number.

Note

If you change the default value of „DeviceNumHigh“, you also have to adjust the array.

Note

If you change the default value of „ModuleNumHigh“, you also have to adjust the two arrays in „ModulStruct“.

Devices.Device[x] [DeviceSingleStruct]

This structure includes the tags of the individual devices.

Table 4-5

Tag name	Data type	Default value	Meaning
Laddr	HW_Device	-	Hardware ID of the device (system data type)
Name	String	-	Device name
ErrorState	Bool	-	Error status of the device
Module	ModulStruct	-	See ModulStruct

Devices.Device[x].Modul [ModulStruct]

This structure includes the module tags of a device.

Table 4-6

Tag name	Data type	Default value	Meaning
SlotLaddr	Array[0..4] of HW_IO	-	Hardware ID of the modules (system data type) The array index corresponds to the slot number.
SlotErrorState	Array[0..4] of Bool	-	Error status of the modules The array index corresponds to the slot number.

ErrorList [ErrorListStruct]

This structure includes the tags of the error list.

Table 4-7

Tag name	Data type	Default value	Meaning
Index	Int	0	Index indicates the last error entry
MaxError	Int	10	Max. Number of error entries
Error	Array[0..10] of ErrorSingleStruct	-	See ErrorSingleStruct

Note

You can specify the maximum number of error entries with MaxError. If you change the value you also have to adjust the array.

ErrorList.Error [ErrorSingleStruct]

This structure includes the tags of an error entry of the error list.

Table 4-8

Tag name	Data type	Default value	Meaning
ErrorState	Bool	-	Error status (0 = outgoing, 1 = incoming)
Laddr	HW_ANY	-	Hardware ID of the faulty components (system data type)
DeviceNr	String	-	Device number of the faulty device
DeviceName	String	-	Name of the faulty device
SlotNr	Int	-	Slot number of the faulty module
Timestamp	DTL	-	Time stamp of the fault

4.3 Function block DiagStartupFB [FB100]

The function block detects the hardware IDs of the components of the plant already when the controller starts up in order not to stress the running process further.

Interfaces

Figure 4-3 Call in "Startup [OB100]"

```

2  □ "DiagStartupFB_IDB" (IO_System:="DiagDataDB".IO_System,
3      PLC:="DiagDataDB".PLC,
4      Devices:="DiagDataDB".Devices);

```

Table 4-9

Type	Parameter	Data type	Description
InOut	IO_System	IOSystemStruct	Diagnostic data of the IO system
	PLC	PLCStruct	Diagnostic data of the controller and their local modules
	Devices	DeviceStruct	Diagnostic data of the devices and modules

Call of instruction „GEO2LOG“

Abbildung 4-4 Call „GEO2LOG“

```

40 // Determine hardware identifier and name from IO devices (1..DeviceNumHigh)
41 #GeoAddr.HWTYPE := 2; // Hardware type 2: IO device
42 #GeoAddr.AREA := 1; // Area ID 1: PROFINET IO
43 #GeoAddr.IOSYSTEM := 100; // PROFINET IO system (100)
44
45 □ FOR #DeviceNum := 1 TO #Devices.DeviceNumHigh DO
46     // Station number
47     #GeoAddr.STATION := INT_TO_UINT(#DeviceNum);
48     // read LADDR from devices
49     #Geo_Retval := GEO2LOG(GEOADDR := #GeoAddr, LADDR => #Geo_Laddr);
50     // check Retval
51 □ IF #Geo_Retval = 0 THEN ... END_IF;
68 END_FOR;

```

Call of instruction „Get_Name“

Abbildung 4-5 Call „Get_Name“

```

54 // Get device name
55 □ "Get_Name_DB" (LADDR := #IO_System.Laddr,
56     STATION_NR := #GeoAddr.STATION,
57     DONE => #GetName.Done,
58     BUSY => #GetName.Busy,
59     ERROR => #GetName.Error,
60     LEN => #GetName.Len,
61     STATUS => #GetName.Status,
62     DATA := #Devices.Device[#DeviceNum].Name);

```

Function description

The hardware IDs are required to determine the diagnostic information in the user program of the diagnostic instructions.

The block is used for detecting the hardware ID of the components automatically with the "GEO2LOG" instruction based on the slot information. The slot information is rewritten before each call of the instruction with a tag of the "GEOADDR" system data type.

The "GET_NAME" instruction additionally reads out the names of the PROFINET IO devices.

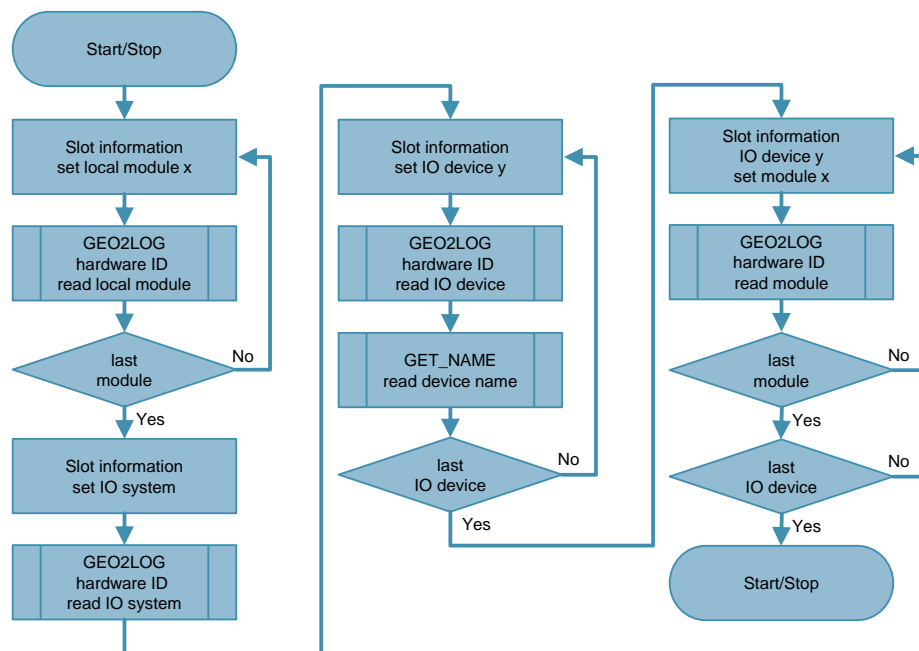
The hardware IDs and the names are saved in the respective structure in the global "DiagDataDB" data block.

The function block detects the hardware IDs of the following components:

- Local modules of the controller
- PROFINET IO system
- PROFINET IO devices
- Modules of the PROFINET IO devices

The following figure shows the principle flow of the function block.

Figure 4-6



4.4 Function block DiagMainFB [FB1]

The function block monitors the diagnostic status of the plant and calls the error diagnostics only if there is a fault.

Interfaces

Figure 4-7 Call in "DiagMainOB [OB123]"

```

3  □ "DiagMainFB_IDB" (IO_System:="DiagDataDB".IO_System,
4                        PLC:="DiagDataDB".PLC,
5                        Devices:="DiagDataDB".Devices,
6                        Error:="DiagDataDB".ErrorList,
7                        AlarmsDeviceStates:= "DiagDataDB".AlarmsDeviceStates,
8                        AlarmsModuleStates:="DiagDataDB".AlarmsModuleStates);

```

Table 4-10

Type	Parameter	Data type	Description
InOut	IO_System	IOSystemStruct	Diagnostic data of the IO system
	PLC	PLCStruct	Diagnostic data of the controller and their local modules
	Devices	DeviceStruct	Diagnostic data of the devices and modules
	Error	ErrorListStruct	Error list of the last error
	AlarmsDeviceStates	Word	Trigger tag for message texts of the "DeviceStates" instruction
	AlarmsModuleStates	Word	Trigger tag for message texts of the "ModuleStates" instruction

Call of instruction „LED“

Abbildung 4-8 Call „LED“

```

5  // check, if PLC error LED is flashing
6  #LedRetVal := LED(LADDR := "PLC_1[Common]", LED := #Devices.ErrorLED);

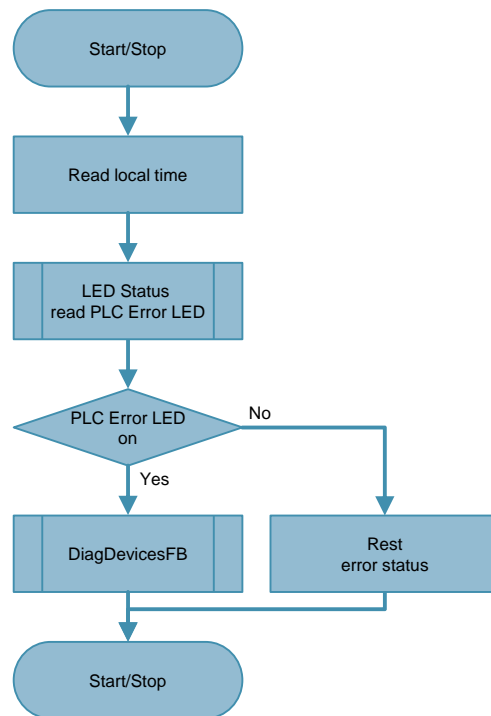
```

Function description

The function block evaluates the status of the error LED of the controller with the “LED” instruction. If the LED displays a fault (LED status = 4), the block calls the “DiagDevicesFB” function block to evaluate the diagnostics. If there is no fault, it resets the error status in the data block.

The following figure shows the principle flow of the function block.

Figure 4-9



4.5 Function block DiagDevicesFB [FB2]

The function block evaluates the status of the PROFINET IO devices and their modules.

Interfaces

Figure 4-10 Call in "DiagMainFB [FB1]"

```

11  #DiagDevicesFB_Instance(IO_System:=#IO_System,
12                               PLC:=#PLC,
13                               Devices:=#Devices,
14                               Error:= #Error,
15                               Local_Time:=#Local_Time,
16                               AlarmsDeviceStates:=#AlarmsDeviceStates,
17                               AlarmsModuleStates:=#AlarmsModuleStates);

```

Table 4-11

Type	Parameter	Data type	Description
InOut	IO_System	IOSystemStruct	Diagnostic data of the IO system
	PLC	PLCStruct	Diagnostic data of the controller and their local modules
	Devices	DeviceStruct	Diagnostic data of the devices and modules
	Error	ErrorListStruct	Error list of the last error
	Local_Time	DTL	Time stamp of the fault
	AlarmsDeviceStates	Word	Trigger tag for message texts of the "DeviceStates" instruction
	AlarmsModuleStates	Word	Trigger tag for message texts of the "ModuleStates" instruction

Call of instruction „DeviceStates“

Abbildung 4-11 Call „DeviceStates“

```

1  // Read out the device states of the complete IO System
2  #DeviceStates_RetVal := DeviceStates(LADDR := #IO_System.Laddr,
3                                     MODE := #Devices.ProblemMode,
4                                     STATE := #Device_State);

```

Call of instruction „ModuleStates“

Abbildung 4-12 Call „ModuleStates“

```

38  // Read module status information of modules from faulty IO devices
39  #ModuleStates_RetVal := ModuleStates(LADDR := #Devices.Device[#DeviceNum].Laddr,
40                                     MODE := #Devices.ProblemMode,
41                                     STATE := #Module_State);

```

Function description

The function block reads the status of the PROFINET IO devices with the "DeviceStates" instruction. If the instruction reports an error, the block interrupts the processing with a message on the operator panel. If the instruction is executed without errors, the block evaluates the status of the individual devices and saves the data in the global "DiagDataDB" data block.

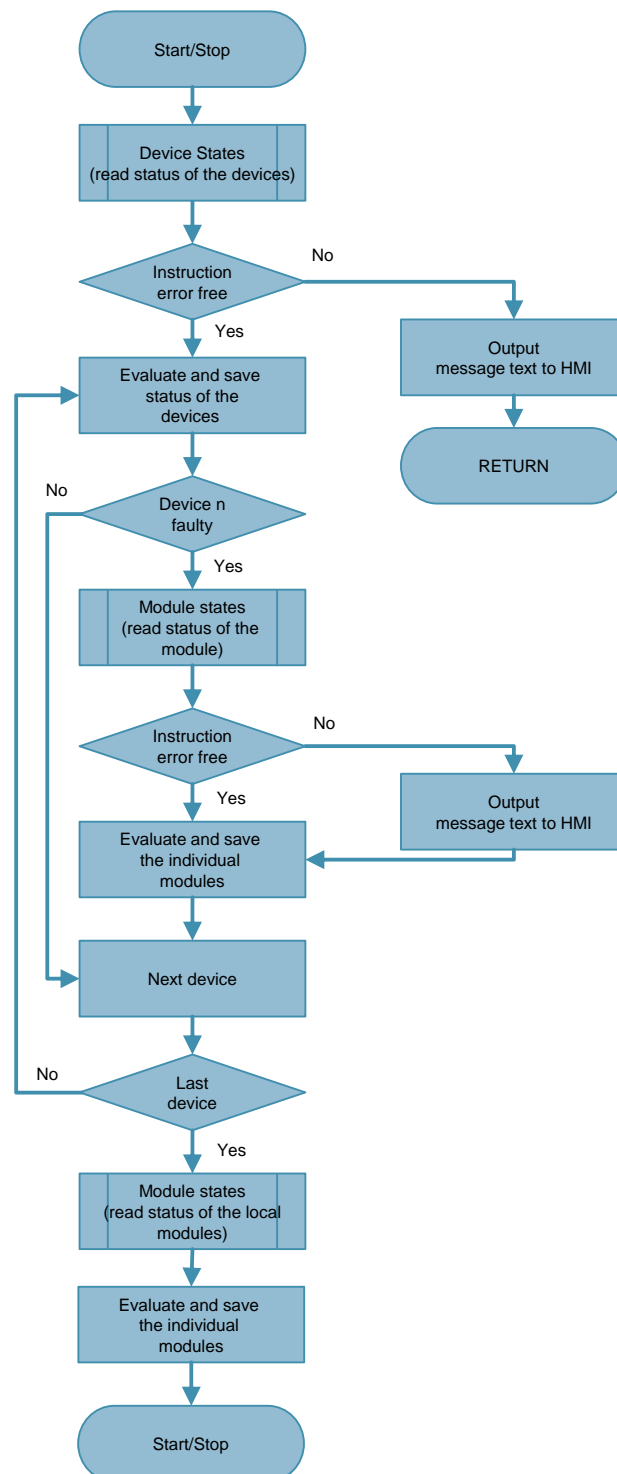
If the devices are faulty, the "ModuleStates" instruction reads out the status of the modules. The block evaluates the status and saves the data in the global "DiagDataDB" data block.

In addition, the "ModuleStates" instruction reads out the local module status of the controller which is also evaluated and saved.

The block saves the incoming and outgoing error events in an error list.

The following figure shows the principle flow of the function block.

Figure 4-13



4.6 Function block DiagSignalFB [FB4]

The function block monitors a binary signal and creates program and diagnostic messages.

Interfaces

Figure 4-14 Call in "DiagMainOB [OB123]"

```
21 □ "DiagSignalFB_IDB" (Signal:=#AlarmSignal,
22   UserMsg:=#UserMsg);
```

Table 4-12

Type	Parameter	Data type	Description
Input	Signal	Bool	The signal to be monitored.
	UserMsg	UserMsgStruct	Data for user diagnostic message

Call of instruction „Program_Alarm“

Abbildung 4-15 Call „Program_Alarm“

```
4 // generate program alarm, if value status is not OK
5 □ #Program_Alarm_Instance (SIG:=#Signal,
6   TIMESTAMP:=#Local_Time,
7   Error=>#Program_Alarm_Error,
8   Status=>#Program_Alarm_Status);
```

Function description

The "Program_Alarm" instruction monitors the binary input signal and generates an incoming or outgoing program message if there is change of signal.

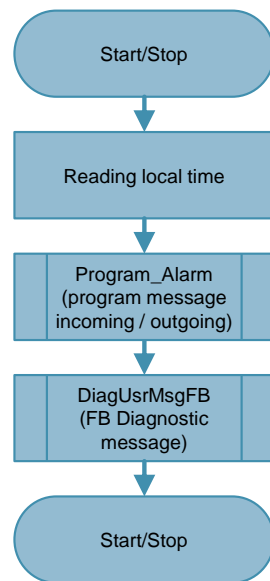
The block calls the "DiagUsrMsgFB" function block and in the process transfers the "UserMsg" input data for a user diagnostic message.

Note

If required, you can read out and evaluate the message status of the program message with the "Get_AlarmState" instruction. In the process, you can e.g. check whether a message was acknowledged in order to then continue the sequence in the program.

The following figure shows the principle flow of the function block.

Figure 4-16



4.7 Function block DiagUsrMsgFB [FB5]

The function block generates a user diagnostic message in the diagnostic buffer of the controller for a binary input signal. The block is required since the "Gen_UsrMsg" instruction otherwise writes the user diagnostic message cyclically into the diagnostic buffer.

Interfaces

Figure 4-17 Call in "DiagSignalFB [FB4]"

```
17 #DiagUsrMsgFB_Instance (UserMsg:= #UserMsg);
```

Table 4-13

Type	Parameter	Data type	Description
Input	UserMsg	UserMsgStruct	Data for a user diagnostic message

Call of instruction „Gen_UsrMsg“

Abbildung 4-18 Aufruf „Gen_UsrMsg“

```
22 // Generate user diagnostic alarm
23 #retval := Gen_UsrMsg (Mode := #Mode,
24                        TextID := #UserMsg.TextID,
25                        TextListID := #UserMsg.TextListID,
26                        AssocValues := #AssocValues);
```

UserMsgStruct

This structure includes the tags for a user diagnostic message.

Table 4-14

Tag name	Data type	Start value	Meaning
Sig	Bool		The signal to be monitored.
TextID	UInt		ID of the text list entry, which is to be used for the message text
TextListID	UInt		ID of the text list that includes the text list entry
Value1 to Value8	Int		Associated values 1 to 8 of the message

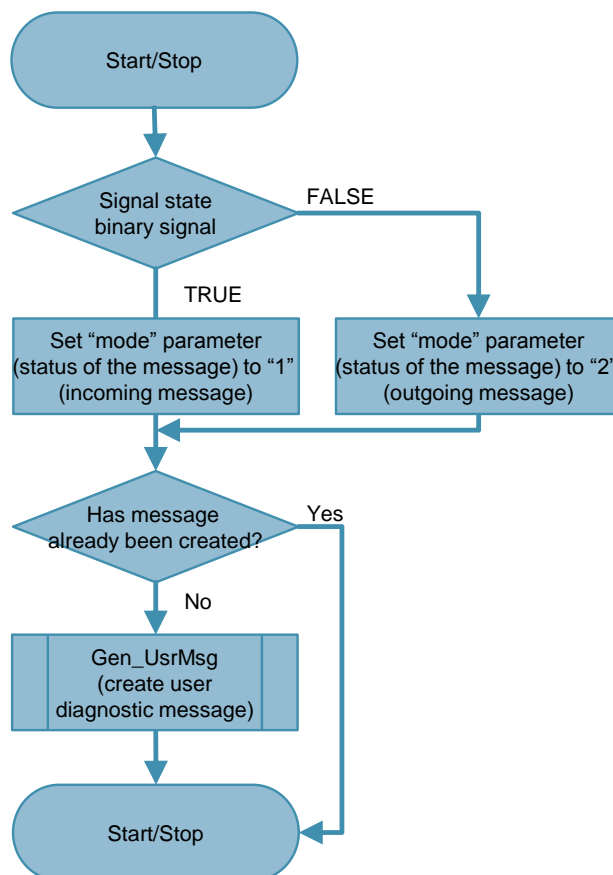
Function description

The function block checks the signal state of the input signal. If the signal status is TRUE, the status of the message ("mode") is set to incoming message or if it is FALSE to outgoing message.

The "Gen_UsrMsg" instruction writes a user diagnostic message into the diagnostic buffer of the controller based on the input parameter of the function block. A query prevents that the message is written cyclically into the diagnostic buffer.

The following figure shows the principle flow of the function block.

Figure 4-19



4.8 Function block DiagPNIOFB [FB3]

The function block evaluates the status of the PROFINET IO systems based on the start information of the OB.

Interfaces

Figure 4-20 Call in "Rack or station failure [OB86]"

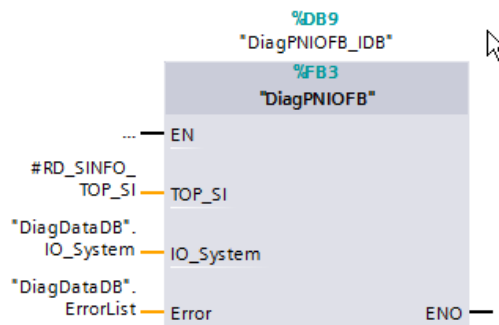


Table 4-15

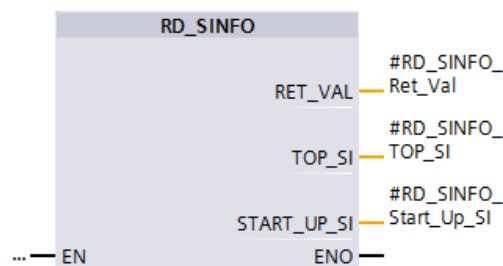
Type	Parameter	Data type	Description
Input	TOP_SI	SI_classic	Start information of the OB
InOut	IO_System	IOSystemStruct	Diagnostic data of the IO system
	Error	ErrorListStruct	Error list of the last error

Note

The description of the system data type (SDT) "SI_classic" can be found in the online help of SIMATIC STEP 7.

Call of instruction „RD_SINFO“ in „Rack or station failure [OB86]“

Abbildung 4-21 Call „RD_SINFO“ in „Rack or station failure [OB86]“



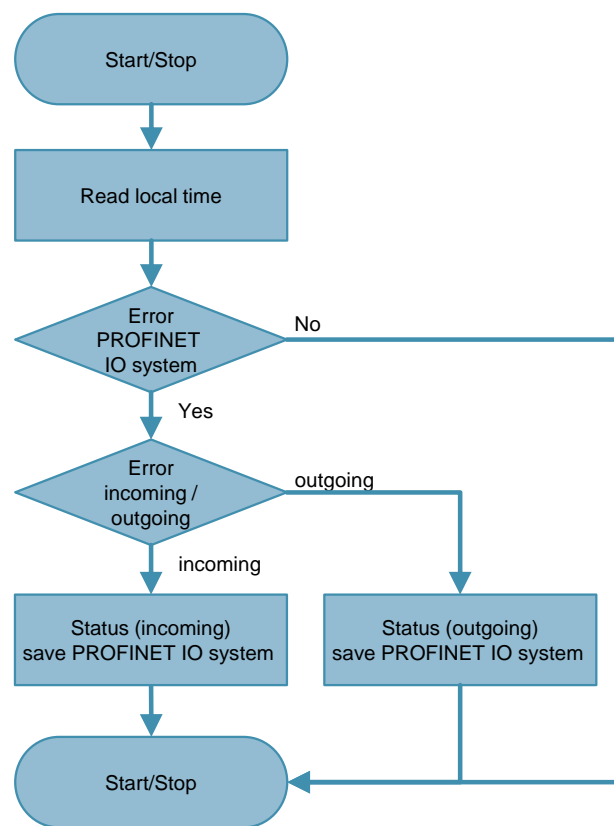
Function description

In the event of a failure of the PROFINET IO system, the operating system of the controller calls the “Rack or station failure” error OB for an incoming and also for an outgoing event. The OB therefore calls the “DiagPNIOFB” function block only once per event and in the process transfers the start information of the OB which is read out in the OB with the “RD_SINFO” instruction.

The “DiagPNIOFB” function block checks the start information of the OB based on whether an event occurred in the PROFINET IO system. If this is the case, it is polled whether an incoming or an outgoing event is pending and the respective status is saved in the global “DiagDataDB” data block.

The following figure shows the principle flow of the function block.

Figure 4-22



5 Configuring the HMI Screens

As an example, this chapter describes the integration of the collected diagnostic data of the controller to the screen elements of the HMI based on the “ET 200MP” device.

5.1 Configuring a device in the plant overview

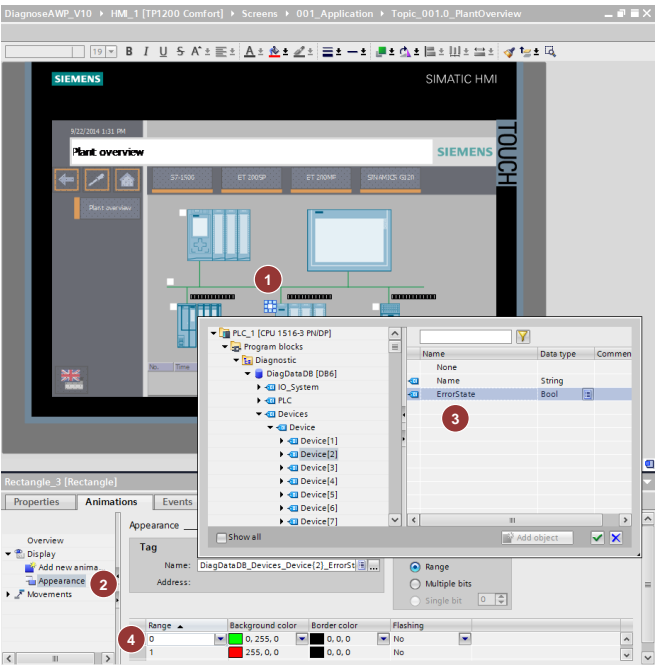
The “Topic_001.0_PlantOverview” screen shows a global overview of the plant. The integration of the “ET 200MP” device is described below.

Table 5-1

No.	Action
1.	Open the “Topic_001.0_PlantOverview” screen in the TIA portal.
2.	<div><div><div>1. Add a graphic object of the device. Alternatively, you can also use a rectangle.</div><div>2. Insert a rectangle into the screen for the status display.</div><div>3. Add an output field into the screen for the name of the STRING type.</div><div>4. Place a transparent button for the navigation in the detail view over the graphic object.</div></div><div></div></div>

5 Configuring the HMI Screens

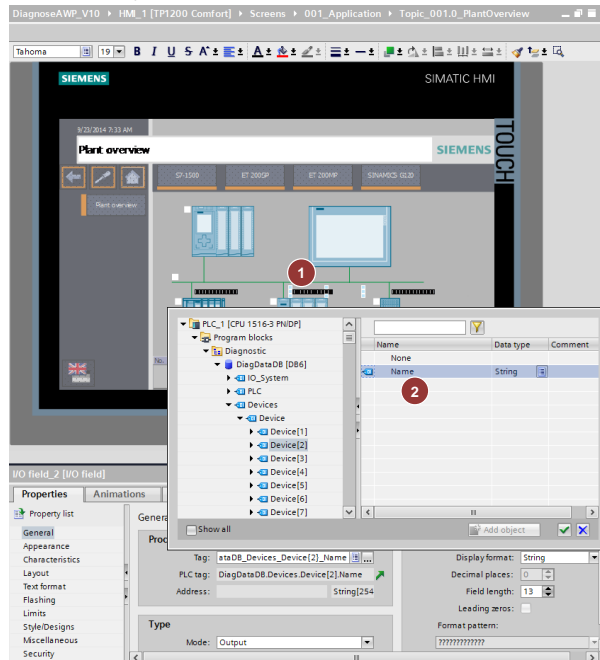
5.1 Configuring a device in the plant overview

No.	Action
3.	<ol style="list-style-type: none"> 1. Select the rectangle for the status display. 2. Add a new animation of the "Appearance" type. 3. Assign the tag the respective control tag of the device status from the global data block. In this example, the device has device number "2" (see device property). For this reason, select "ErrorState" of the device with index "2". 4. Specify the background colors for the ranges "0" and "1". 

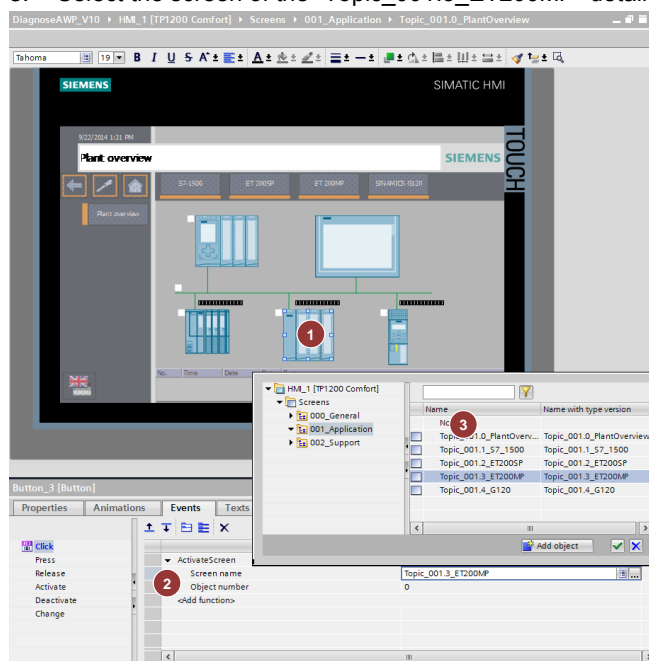
5 Configuring the HMI Screens

5.1 Configuring a device in the plant overview

4.
 1. Select the output field for the device name.
 2. Assign the tag the respective control tag of the device name from the global data block. In this example, the device has device number "2" (see device property). For this reason, select "Name" of the device with index "2".



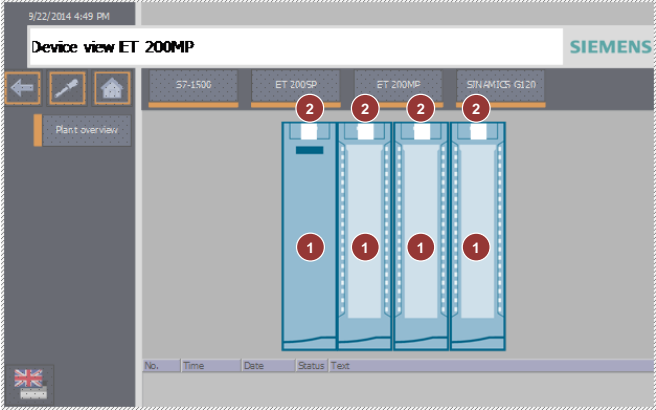
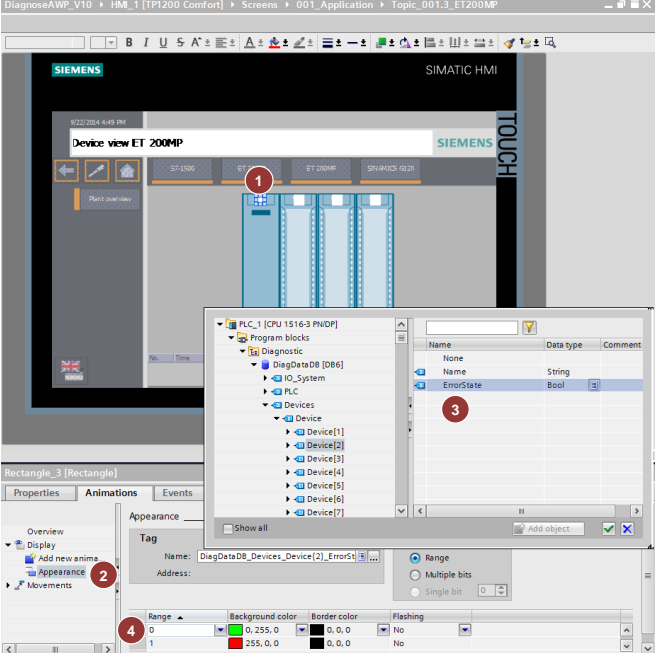
5.
 1. Select the transparent button for the navigation in the detail view.
 2. Add the "Activate Screen" function in "Click".
 3. Select the screen of the "Topic_001.3_ET200MP" detail view.



5.2 Configuring a device in the detail view

The “Topic_001.3_ET200MP” screen shows the detail view of the “ET 200MP” device. The integration of the modules is described below.

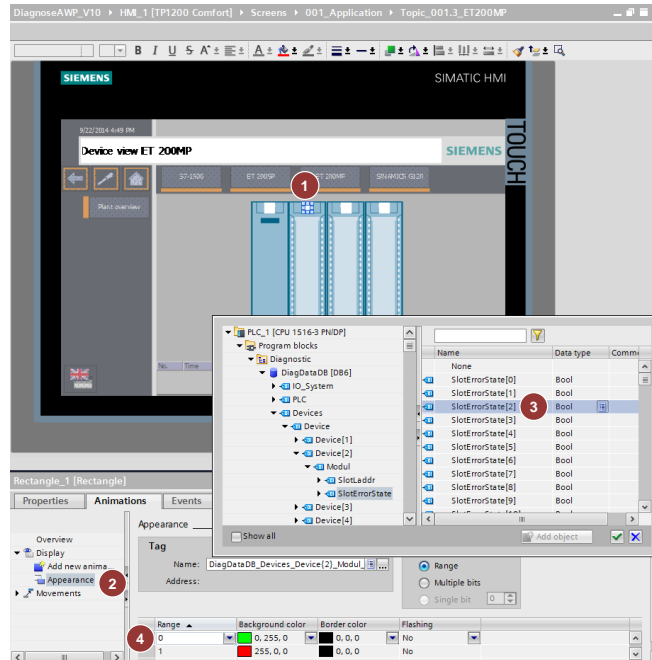
Table 5-2

No.	Action
1.	Open the “Topic_001.3_ET200MP” screen in the TIA Portal.
2.	<div><div>1. Add the graphic objects of the modules. Alternatively, you can also use a rectangle.</div><div>2. Insert a rectangle into the screen for the status display.</div></div> 
3.	<div><div>1. Select the rectangle for the status display of the head module.</div><div>2. Add a new animation of the “Appearance” type.</div><div>3. Assign the tag the respective control tag of the device status from the global data block. In this example, the device has device number “2” (see device property). For this reason, select “ErrorState” of the device with index “2”.</div><div>4. Specify the background colors for the ranges “0” and “1”.</div></div> 

5 Configuring the HMI Screens

5.2 Configuring a device in the detail view

4.
 1. Select the rectangle for the status display of the first module.
 2. Add a new animation of the “Appearance” type.
 3. Assign the tag the respective control tag of the module from the global data block. In this example, the module is inserted in slot 2 (see device view). For this reason, select “SlotErrorState” with index “2”.
 4. Specify the background colors for the ranges “0” and “1”.

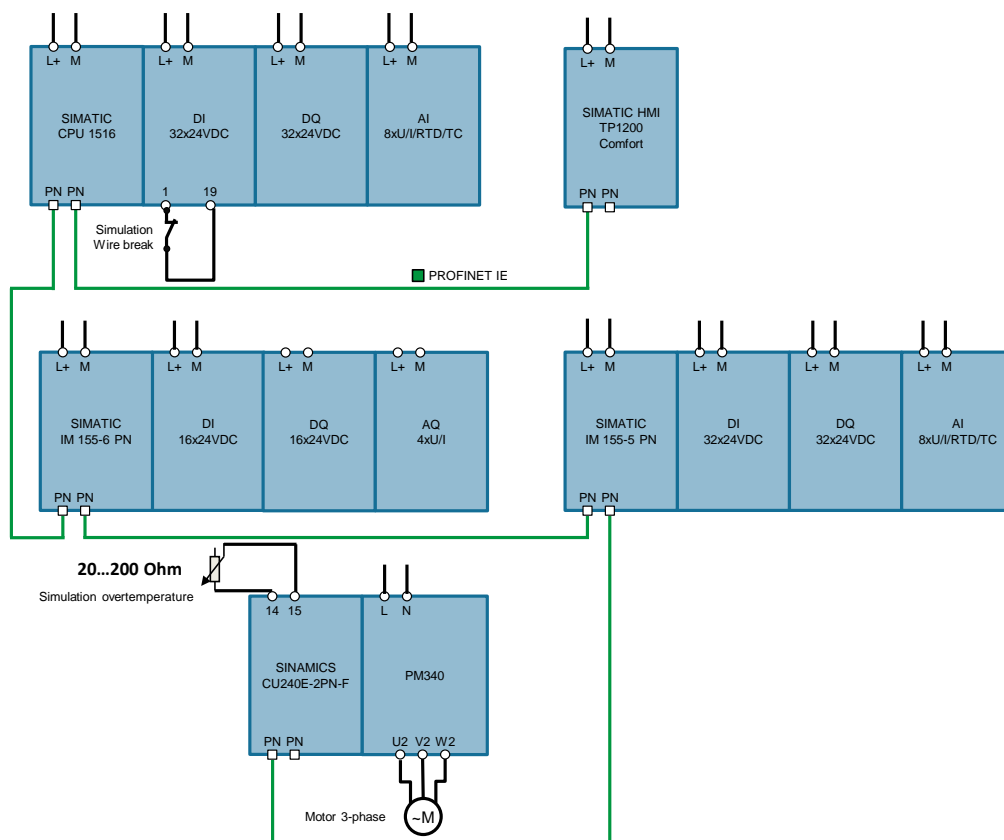


6 Installation and Commissioning

6.1 Installing the hardware

The figure below shows the hardware setup of the application.

Figure 6-1



Note

Always note the setup guidelines of the devices.

6.2 IP addresses and device names

The following device numbers, IP addresses and device names are used in the example:

Table 6-1

Component	Device number	IP address	Device name (PROFINET name)
SIMATIC CPU 1516	0	192.168.0.1	PLC_1
SIMATIC IM 155-6 PN	1	192.168.0.2	ET200SP
SIMATIC IM 155-5 PN	2	192.168.0.3	ET200MP
SINAMICS CU240E	3	192.168.0.10	Drive_1
SIMATIC HMI TP1200	-	192.168.0.4	HMI_1

6.3 Installing the software (download)

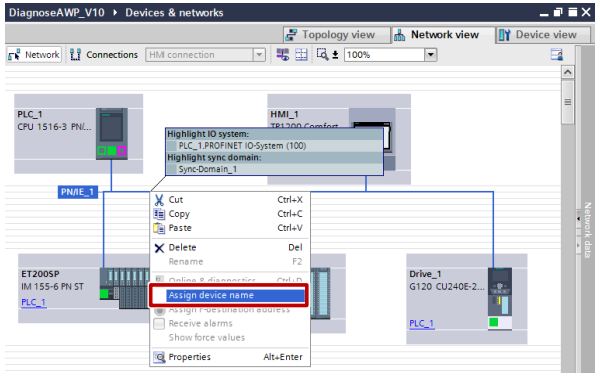
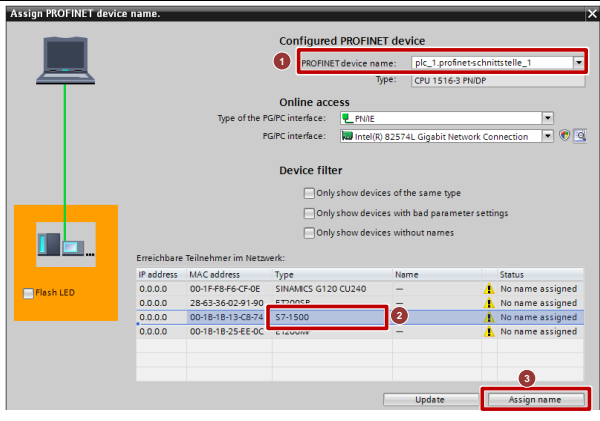
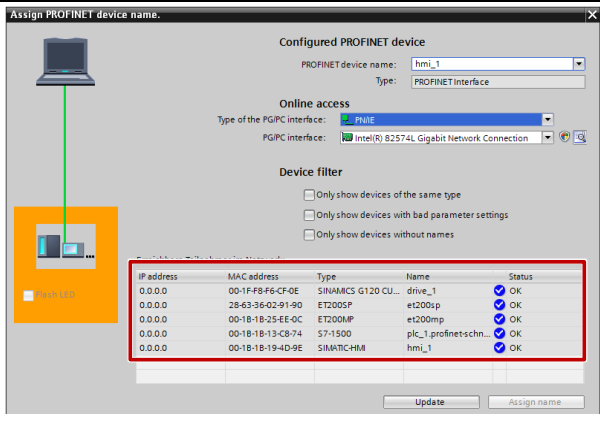
Note

At this point, it is assumed that the necessary software has been installed on your computer and that you are familiar with the software.

6.4 Assigning PROFINET device names

In order for all PROFINET devices to be able to communicate with each other, a PROFINET device name must be assigned. The configured IP addresses of the devices are automatically transferred when downloading the project.

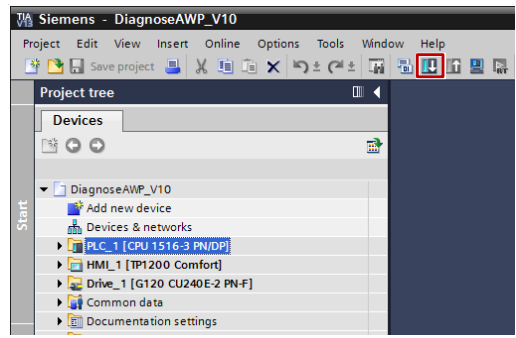
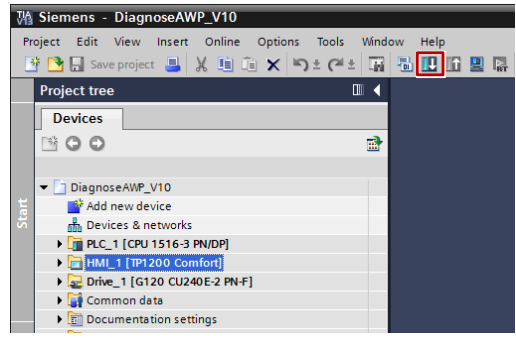
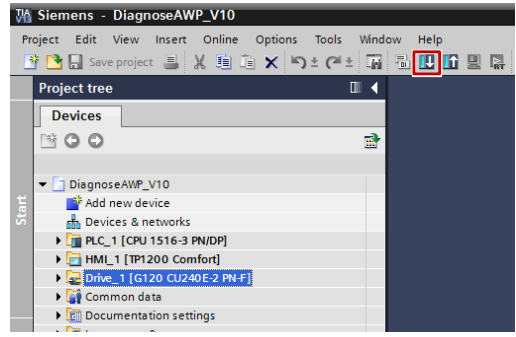
Table 6-2: Instruction –PROFINET in the TIA Portal

No.	Action	Remark
1.	Start the TIA Portal and open the example project.	-
2.	Open “ Devices & Network ” and enable the “ Network view ”. Right click on the PROFINET connection and select “ Assign device name ”.	
3.	In this window, assign the PROFINET device names to all of the devices. 1. Select the “ PROFINET device name ”. 2. Assign the correct devices to the PROFINET device names. 3. Click “ Assign name ”.	
4.	Repeat step 3 until all devices have a PROFINET device name.	

6.5 Loading the project

The software example is available on the HTML page from which you downloaded this document.

Table 6-3

No.	Action	Remark
1.	Unzip the compressed code folder 98210758_CODE_V10.zip into a directory of your choice.	
2.	Open the "DiagnoseAWP_V11.ap15_1" project with the TIA Portal V15.1 Update 1.	
3.	Select the "PLC_1" folder of the controller in the project navigation and click the "Download to device" button in the toolbar.	
4.	Select the "HMI_1" folder in the project navigation of the operator panel and click the "Download to device" button in the toolbar.	
5.	Select the "Drive_1" folder in the project navigation of the operator panel and click the "Download to device" button in the toolbar. Notice All motor data is always downloaded. This may possibly result in a faulty configuration. If you do not want to do this, only change the parameter p601 = [2] KTY.	

6.6 Integrating the application into an existing project

You can completely integrate the application described here into your project. The required steps are described below.

6.6.1 Configuring the diagnostic settings

You can release the module-specific diagnostic settings for every module of your project separately.

The following diagnostic settings are e.g. possible:

- Missing supply voltage L+
- Wire break
- Short circuit to ground
- Value status
- Channel diagnostics for drives
- etc.

Note

For information of how to configure the diagnostic settings, please refer to [System diagnostics with S7-1500 and TIA Portal](#) in chapter 5.

6.6.2 Integration of the PLC elements

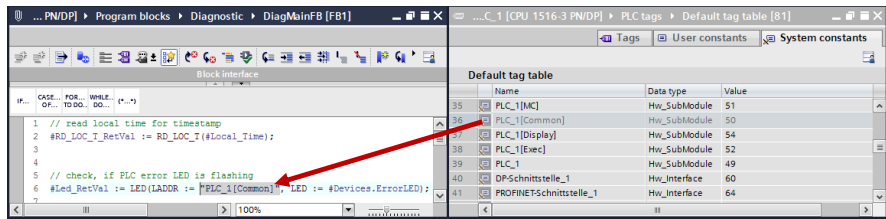
Table 6-4

No.	Action
1.	Copy the “Diagnostic” folder in your project in “PLC_1 > PLC data types”.
2.	Copy the “Diagnostic” folder in your project in “PLC_1 > Program blocks”.
3.	If your project already includes error OBs, copy the networks of the error OBs of the application to their respective error OBs. Note that the position of the inserted code may have an effect on the program sequence. Then delete the error OBs of the application.
4.	Open the data type „PLCStruct“ and adapt the default value of the tag „ModuleNumHigh“ as well as the size of the arrays „SlotLaddr“ and „SlotErrorState“ to the highest slot number occupied by the local modules.

PLCStruct			
	Name	Data type	Default value
1	ModuleNumHigh	Int	4
2	DeviceIdPLC	HW_DEVICE	32
3	ErrorState	Bool	false
4	▶ SlotLaddr	Array[0 .. 4] of HW_IO	
5	▶ SlotErrorState	Array[0 .. 4] of Bool	

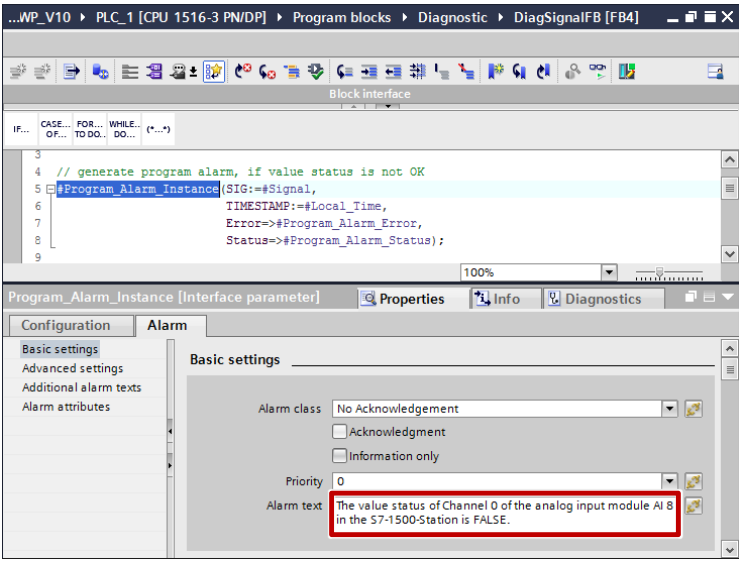
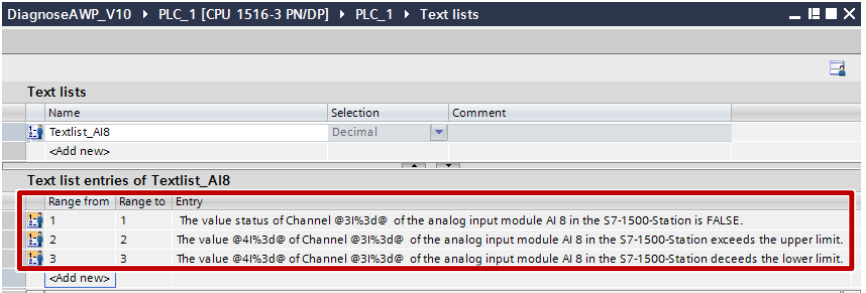
6 Installation and Commissioning

6.6 Integrating the application into an existing project

No.	Action																																																
5.	<p>Open the data types „DeviceStruct“ and „ModulStruct“.</p> <p>Adapt the default value of the tag „DeviceNumHigh“ as well as the size of the array „Device“ to the highest device number.</p> <p>Adapt the default value of the tag „ModuleNumHigh“ as well as the size of the arrays „SlotLaddr“ and „SlotErrorState“ to the highest slot number occupied by the modules.</p> <table><thead><tr><th colspan="4">DeviceStruct</th></tr><tr><th></th><th>Name</th><th>Data type</th><th>Default value</th></tr></thead><tbody><tr><td>1</td><td>DeviceNumHigh</td><td>Int</td><td>3</td></tr><tr><td>2</td><td>ModuleNumHigh</td><td>Int</td><td>4</td></tr><tr><td>3</td><td>ErrorLED</td><td>UInt</td><td>2</td></tr><tr><td>4</td><td>ErrorLEDFlash</td><td>Int</td><td>4</td></tr><tr><td>5</td><td>ProblemMode</td><td>UInt</td><td>5</td></tr><tr><td>6</td><td>Device</td><td>Array[1] of "DeviceSingleStruct"</td><td></td></tr></tbody></table> <table><thead><tr><th colspan="4">ModulStruct</th></tr><tr><th></th><th>Name</th><th>Data type</th><th>Default value</th></tr></thead><tbody><tr><td>1</td><td>SlotLaddr</td><td>Array[0] of HW_IO</td><td></td></tr><tr><td>2</td><td>SlotErrorState</td><td>Array[0] of Bool</td><td></td></tr></tbody></table>	DeviceStruct					Name	Data type	Default value	1	DeviceNumHigh	Int	3	2	ModuleNumHigh	Int	4	3	ErrorLED	UInt	2	4	ErrorLEDFlash	Int	4	5	ProblemMode	UInt	5	6	Device	Array[1] of "DeviceSingleStruct"		ModulStruct					Name	Data type	Default value	1	SlotLaddr	Array[0] of HW_IO		2	SlotErrorState	Array[0] of Bool	
DeviceStruct																																																	
	Name	Data type	Default value																																														
1	DeviceNumHigh	Int	3																																														
2	ModuleNumHigh	Int	4																																														
3	ErrorLED	UInt	2																																														
4	ErrorLEDFlash	Int	4																																														
5	ProblemMode	UInt	5																																														
6	Device	Array[1] of "DeviceSingleStruct"																																															
ModulStruct																																																	
	Name	Data type	Default value																																														
1	SlotLaddr	Array[0] of HW_IO																																															
2	SlotErrorState	Array[0] of Bool																																															
6.	<p>Open the FB “DiagMainFB” and the “Default tag table”. Replace the HW ID “PLC_1[Common]” with the HW ID of your controller. To do this, drag the respective tag from the tag table via drag & drop to “PLC_1[Common]” in the FB.</p> 																																																
7.	<p>Open the OB “DiagMainOB” and assign a new signal of your plant to the “AlarmSignal” tag. Replace “ValueStatus_AI8”.%X0.</p> <p>If required, duplicate the program part for further signals or delete the program part if you do not wish to evaluate signals.</p> <pre>11 // The following example shows, how the value status (quality information) can be evaluated. 12 // evaluate value status from analog input 1; value status = 1 --> status OK 13 // 14 #AlarmSignal := NOT ("ValueStatus_AI8".%X0); 15 16 #UserMsg.Sig := #AlarmSignal; 17 #UserMsg.TextID := 1; //Text ID 1 for AI8 (see Textlist_AI8) 18 #UserMsg.TextListID := 513; //Textlist ID for AI8 (see Textlist_AI8) 19 #UserMsg.Valuel := 0; 20 21 "DiagSignalFB_IDB"(Signal:=#AlarmSignal, 22 UserMsg:=#UserMsg);</pre>																																																

6 Installation and Commissioning

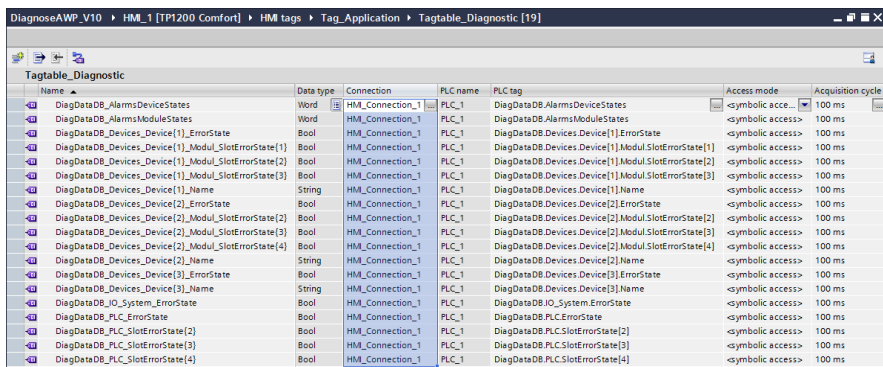
6.6 Integrating the application into an existing project

No.	Action												
8.	<p>Open the FB “DiagSignalFB” and adjust the texts to the new signal for the “Program_Alarm” instance “#Program_Alarm_Instance”.</p> 												
9.	<p>Copy the “Textlist_AI8” text list into your project in “PLC_1 > Text lists”.</p>												
10.	<p>Open the text lists in “PLC_1 > Text lists”. Adjust the texts to the new signal and, if required, add new text lists for further signals.</p>  <table data-bbox="489 1303 1327 1404"><thead><tr><th>Range from</th><th>Range to</th><th>Entry</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>The value status of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station is FALSE.</td></tr><tr><td>2</td><td>2</td><td>The value @4%3d@ of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station exceeds the upper limit.</td></tr><tr><td>3</td><td>3</td><td>The value @4%3d@ of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station exceeds the lower limit.</td></tr></tbody></table>	Range from	Range to	Entry	1	1	The value status of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station is FALSE.	2	2	The value @4%3d@ of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station exceeds the upper limit.	3	3	The value @4%3d@ of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station exceeds the lower limit.
Range from	Range to	Entry											
1	1	The value status of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station is FALSE.											
2	2	The value @4%3d@ of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station exceeds the upper limit.											
3	3	The value @4%3d@ of Channel @3%3d@ of the analog input module AI 8 in the S7-1500-Station exceeds the lower limit.											

6.6.3 Integration of the HMI elements

This chapter describes the integration of the complete HMI application.

Table 6-5

No.	Action
1.	Copy the “Diagnostic” folder to your HMI project in “HMI_1 > HMI tags”. The HMI messages are automatically copied in the process.
2.	Open the “HMI tags” “Tagtable_Diagnostic” and set the HMI connection of your HMI in the “Connection” column. 
3.	Copy the folders “000_General”, “001_Application” and “002_Support” in your HMI project in “HMI_1 > Screens”. The “Templates” are automatically copied in the process.
4.	Link the screens with the already existing screens.

Note

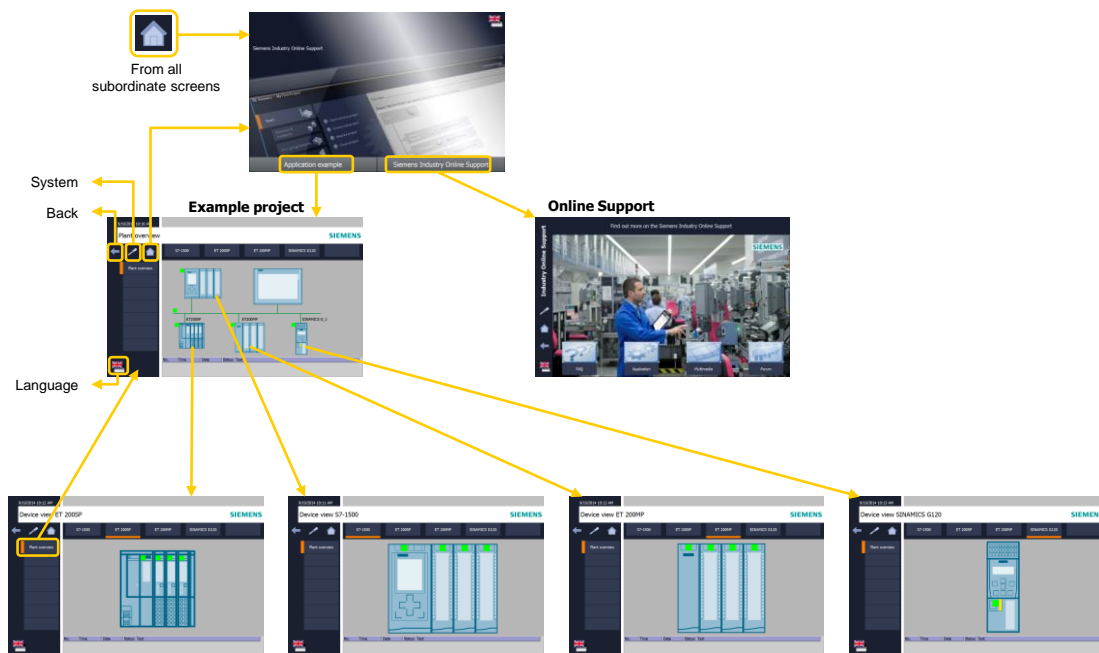
You can also only copy the tags and screens relevant for you. You have to adjust the elements in order for the HMI application to be compiled without errors.

7 Operating the Application

7.1 Overview

The screen below shows the user interface of the operator panel.

Figure 7-1



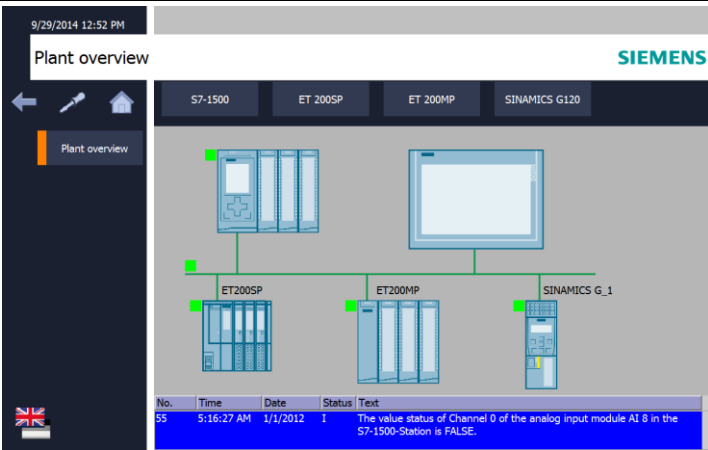
The “Plant overview” screen displays the configured PROFINET IO devices. The detail view of the device opens by clicking on a device. By clicking on the “Plant overview” button you go back to the “Plant overview” screen.

7.2 Diagnostics on the operator panel

7.2.1 Diagnostics “Value status on AI8”

The value status (quality information) is enabled on the analog input module AI8 of the CPU S7-1516. This example evaluates the value status and creates a program message with the “Program_Alarm” instruction as well as a diagnostic message with the instruction “Gen_UsrMsg” which is entered in the diagnostic buffer. In order to diagnose the fault, please proceed as follows.

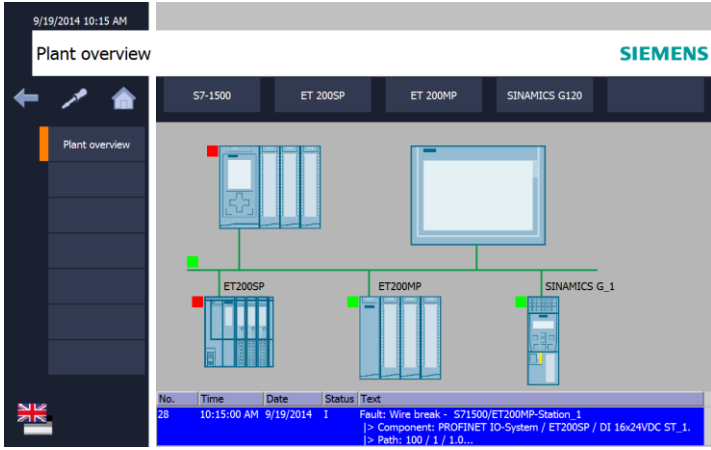

Table 7-1

No.	Action	Remark
1.	Pull, for example, the front plug of the analog input module AI8 in order to set the value status to FALSE.	
2.	Open the “Plant Overview” screen on the operator panel. The program message can be seen in the bottom part.	 The screenshot shows the Siemens Plant Overview interface. At the top, it displays the date and time '9/29/2014 12:52 PM' and the 'SIEMENS' logo. Below this, there's a navigation bar with icons for back, search, and home, and a list of components: S7-1500, ET 200SP, ET 200MP, and SINAMICS G120. The main area shows a hierarchical diagram of the plant components. At the bottom, there's a table with columns 'No.', 'Time', 'Date', 'Status', and 'Text'. The table contains one entry: '55 5:16:27 AM 1/1/2012 1 The value status of Channel 0 of the analog input module AI 8 in the S7-1500-Station is FALSE.'

7.2.2 Diagnostics “Wire break on the DI module of the ET 200SP”

The diagnostics “Wire break” are enabled on the digital input module DI16 of the ET 200SP. In order to diagnose the fault, please proceed as follows.

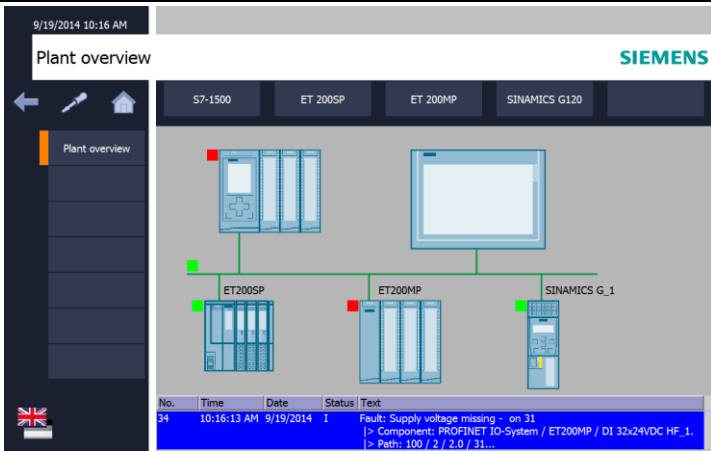

Table 7-2

No.	Action	Remark
1.	Simulate a wire break with a switch on channel 0.	
2.	Open the “Plant overview” screen on the operator panel. The screen displays the fault on the ET 200SP and on the controller. The error message created by the TIA Portal can be seen in the bottom part. For more detailed information of the fault, click the symbol of the ET 200SP or the “ET 200SP” button.	
3.	You can see the modules of the ET 200SP in the “ET 200SP” screen. The head module and the faulty module show an error.	

7.2.3 Diagnostics “Supply voltage missing on the DI module of the ET 200MP”

The diagnostics “Supply voltage L+ missing” are enabled on the digital input module DI32 of the ET 200MP. In order to diagnose the fault, please proceed as follows.

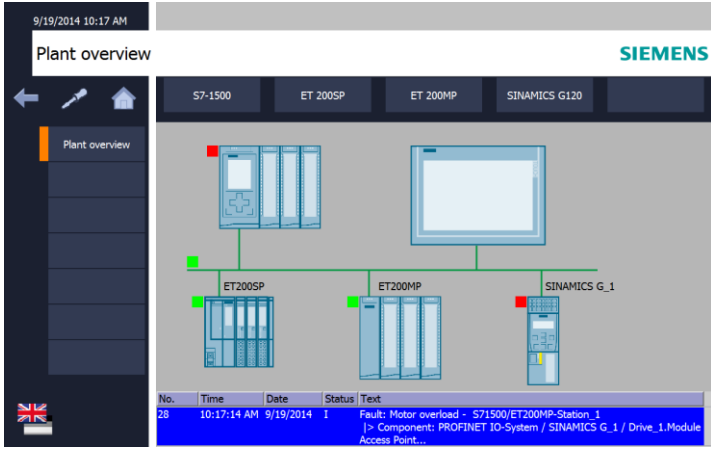

Table 7-3

No.	Action	Remark
1.	Pull the front plug of the input module DI32.	
2.	Open the “Plant overview” screen on the operator panel. The screen displays the fault on the ET 200MP and on the controller. The error messages created by the TIA Portal can be seen in the bottom part. For more detailed information of the fault, click the symbol of the ET 200SP or the “ET 200SP” button.	
3.	You can see the modules of the ET 200MP in the “ET 200MP” screen. The head module and the faulty module show an error.	

7.2.4 Diagnostics “Overtemperature on G120 drive”

The channel diagnostics are enabled on the “Drive_1” drive. In order to diagnose the fault, please proceed as follows.

Table 7-4

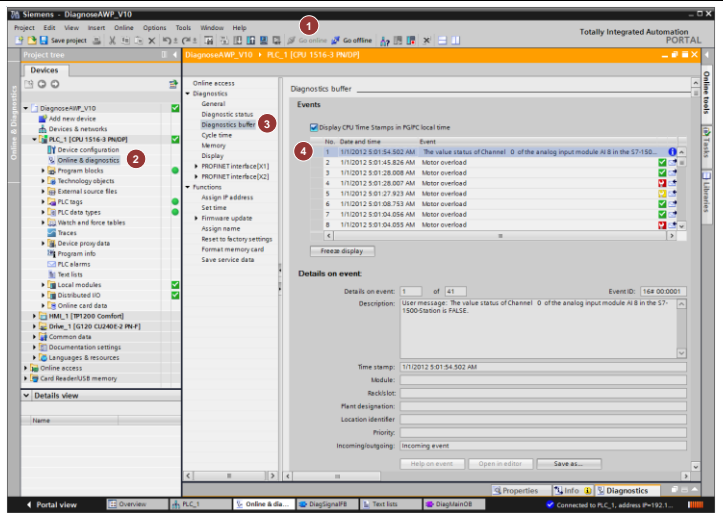
No.	Action	Remark
1.	Simulate the overtemperature with the potentiometer on the drive. See Figure 6-1	
2.	Open the “Plant overview” screen on the operator panel. The screen displays the fault on the drive and the controller. The error messages created by the TIA Portal can be seen in the bottom part. For more detailed information of the fault, click the symbol of the drive or the “SINAMICS G120” button.	
3.	You can see the drive in the “Device view SINAMICS G120” screen. The drive shows an error.	

7.3 Diagnostics in the TIA Portal

7.3.1 Diagnostics “Value status on AI8”

The value status (quality information) is enabled on the analog input module AI8 of the CPU S7-1516. This example evaluates the value status and creates a program message with the “Program_Alarm” instruction as well as a diagnostic message with the instruction “Gen_UsrMsg” which is entered in the diagnostic buffer. In order to diagnose the fault, please proceed as follows.

Table 7-5

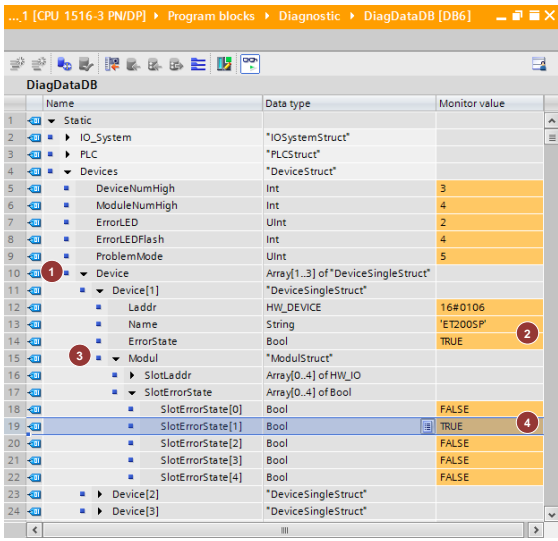
No.	Action	Remark
1.	Pull, for example, the front plug of the analog input module AI8 in order to set the value status to FALSE.	
2.	To display the diagnostic message, proceed as follows: <ol style="list-style-type: none">1. Select the “PLC_1” folder and click “Go online”2. Open the online and diagnostic window by double clicking "Online & diagnostics"3. Click “Diagnostic buffer”4. In the “Events” area, you will find the diagnostic message.	

7.3.2 Diagnostics “Wire break on the DI module of the ET 200SP”

The diagnostics “Wire break” are enabled on the digital input module DI16 of the ET 200SP. In order to diagnose the fault, please proceed as follows.

Table 7-6

No.	Action	Remark
1.	Briefly simulate a wire break on channel 0 with a switch.	
2.	<p>The fault is saved in the global data block “DiagDataDB” in an “ErrorList”. The error list is to be interpreted as follows:</p> <ol style="list-style-type: none"> Open the global data block “DiagDataDB” in the TIA Portal. Click the “Watch all” button. Open the “ErrorList” folder in the global “DiagDataDB” data block. The “Index” tag indicates the index of the last error entry. The entry with index 1 displays the “incoming” fault on device 1 and slot 1. The entry with index 2 (last entry) displays the “outgoing” fault. 	

No.	Action	Remark
3.	<p>The pending fault is saved in “DiagDataDB” in a “Devices” device list. The list is to be interpreted as follows:</p> <ol style="list-style-type: none">1. Open the “Device > Device[1]” folder.2. This is where the error status and the device name is displayed.3. Open the “Module > SlotErrorState” folder.4. Here, an error on slot 1 is displayed.	

Note For the scenarios “Missing supply voltage” and “Overtemperature on the drive” you can proceed as described above.

8 Related Literature

Table 8-1

	Topic	Title / Link
\1\	Siemens Industry Online Support	http://support.industry.siemens.com
\2\	Download page of this entry	https://support.industry.siemens.com/cs/ww/en/view/98210758
\3\	SINAMICS Startdrive V15.1	Commissioning tool for SINAMICS drives as option package for SIMATIC STEP 7 V15.1 https://support.industry.siemens.com/cs/ww/en/view/109760845
\4\	System Diagnostics with S7-1500 and TIA Portal	https://support.industry.siemens.com/cs/ww/en/view/68011497
\5\	SIMATIC STEP 7 Basic/Professional V15.1 und SIMATIC WinCC V15.1 Systemhandbuch	https://support.industry.siemens.com/cs/ww/en/view/109755202
\6\	S7-1500 System Manual	https://support.industry.siemens.com/cs/ww/en/view/59191792

9 History

Table 9-1

Version	Date	Modifications
V1.0	09/2014	First version
V1.1	07/2019	Project upgraded to TIA Portal V15.1