PROFIdrive Application example AC1

Application example

https://support.industry.siemens.com/cs/ww/en/view/109751271

Siemens Industry Online Support

Warranty and liability

Table of contents

1 Preface

1.1 Purpose of the manual

The user documentation describes the software functionality of the PROFIdrive application example of Application Class 1 (AC1) based on the Evaluation Kit, EK-ERTEC 200P-2 PN IO V4.4.0. It is based on the Programming and Operating Manual "Interface description PROFINET IO Development Kits V4.4.0" /2/ of the Evaluation Kit. The PROFIdrive application example is realized according to PROFIdrive V4.2/1/.

1.2 Target group for the manual

This manual is intended for software developers and application developers who want to use the Evaluation Kit for new products with the PROFIdrive profile or to port the example implementation of the PROFIdrive profile to a different PROFINET hardware platform.

This manual applies to ERTEC 200P-based platforms.

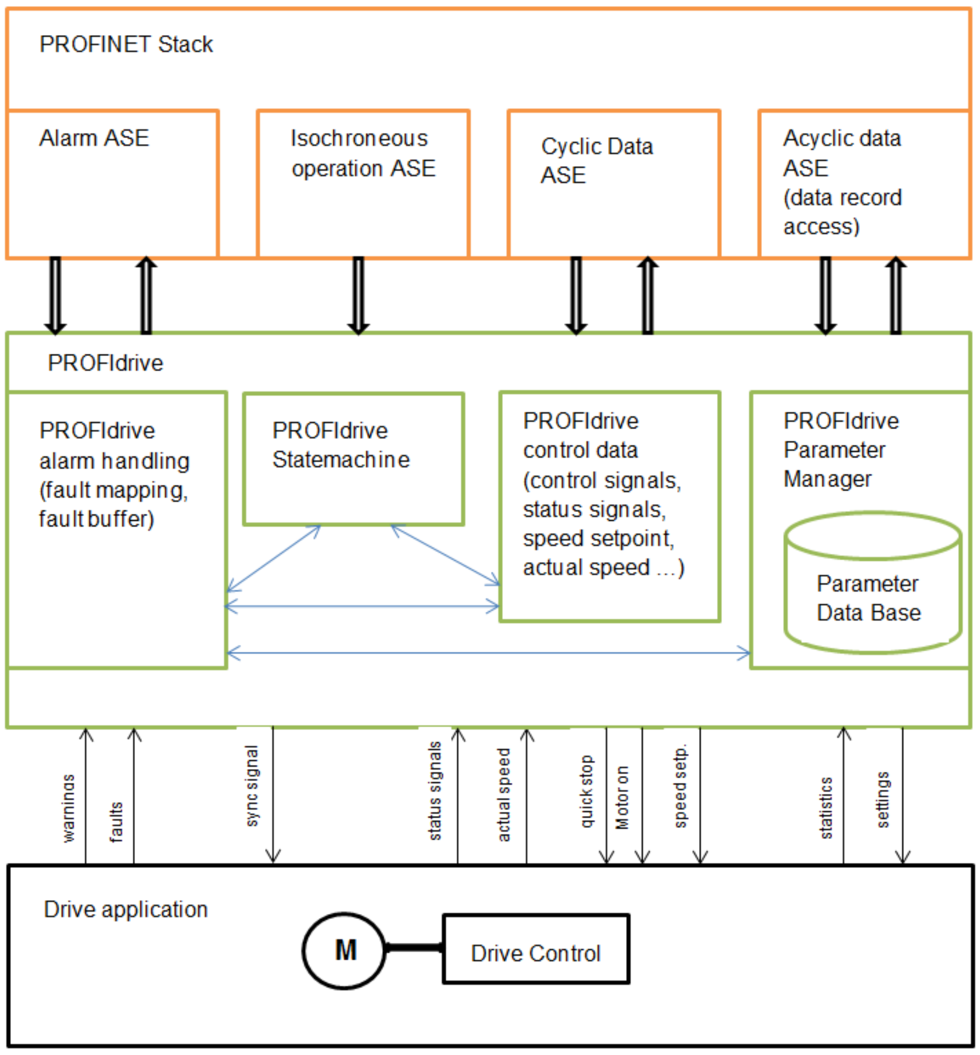
A good understanding of PROFINET IO and PROFIdrive are required in order to be able to use the PROFINET firmware stack and the PROFIdrive application example.

2 Introduction

The PROFIdrive profile is the tried and tested device profile for drive units based on PROFIBUS and PROFINET. With the application example presented here, you can shorten the development time for realization of a PROFIdrive connection of a drive unit. The application example is based on the PROFINET stack for the EK ERTEC 200P-2 Evaluation Kit. It can also serve as the basis for porting to other platforms and communication stacks.

PROFIdrive is a standardized application interface for drives and encoders and is defined in IEC61800-7 as well as Chinese Recommendatory National Standard GB/T 25740.

Figure 2-1 Overview: PROFIdrive (green) as the application layer between the PROFINET stack (orange) and the drive application (black) from drive perspective


PROFINET Stack
Alarm ASE
Isochroneous operation ASE
Cyclic Data ASE
Acyclic data ASE (data record access)
PROFIdrive
PROFIdrive alarm handling (fault mapping, fault buffer)
PROFIdrive Statemachine
PROFIdrive control data (control signals, status signals, speed setpoint, actual speed ...)
PROFIdrive Parameter Manager
Parameter Data Base
warnings
faults
sync signal
status signals
actual speed
quick stop
Motor on
speed setp.
statistics
settings
Drive application
M
Drive Control

The PROFINET stack is the firmware version delivered with the Evaluation Kit. PROFIdrive is implemented by the application example presented here. The "Drive application" is simulated in the application example by a simple simulation (PT1 element).

The functionality of the PROFIdrive application example includes:

- Implementation of a drive object (a P-device consisting of exactly one Drive Unit and exactly one Drive Object) of the application class (AC) 1 with speed setpoint interface.
- Cyclic data exchange via standard telegram 1 (control word, 16-bit speed setpoint, status word, 16-bit actual speed value) and standard telegram 2 (control word, 32-bit speed setpoint, status word, 32-bit actual speed value, no sign of life)
- Acyclic data exchange via PROFIdrive parameter channel (BMP protocol)
- PROFIdrive diagnostics
- PROFIdrive General State Machine
- Simulation of a drive control system

There is a GSDML file for the PROFIdrive application example. The GSDML file is based on the GSDML file of the Evaluation Kit supplemented with entries for the PROFIdrive application example (PROFIdrive API).

The application example was tested with the PROFIdrive PROFINET Profile Tester V50. The PROFIdrive PROFINET Profile Tester is the test tool that the PI test labs use for PROFIdrive certification. The profile tester can also be used for the accompanying test in the development of PROFIdrive profile drives.

Link to the PROFIdrive PROFINET Profile Tester:

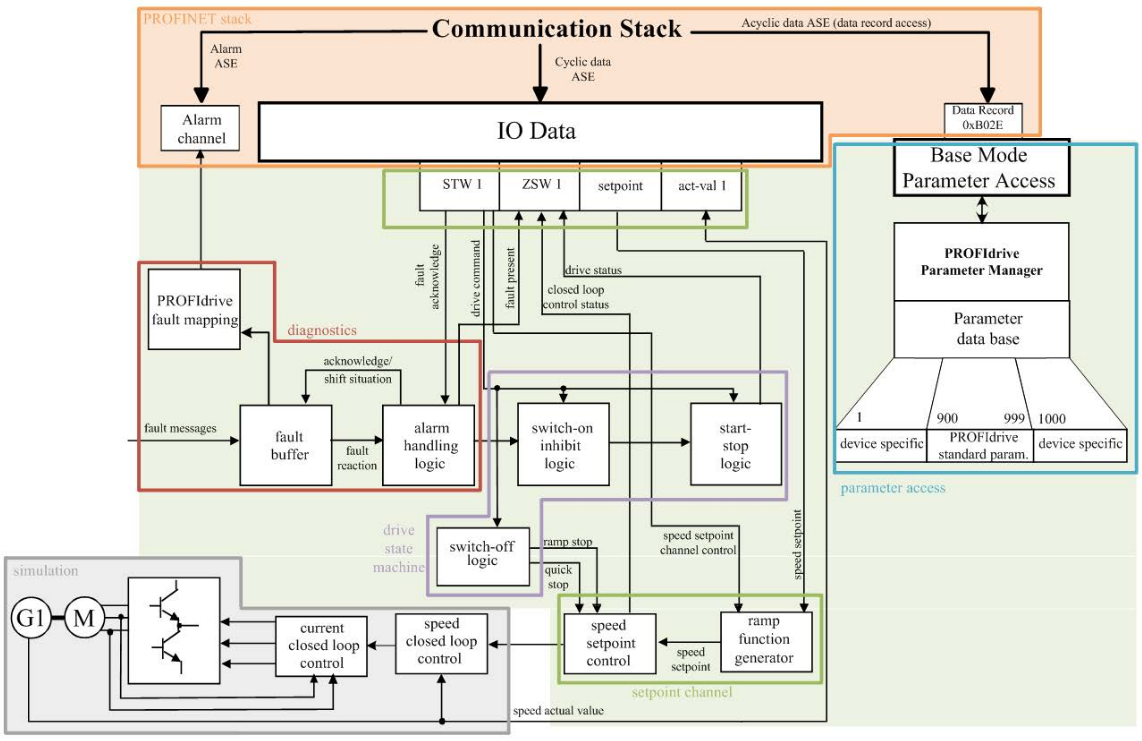
https://www.profibus.com/download/profidrive-profinet-profile-tester/

The PROFIdrive Implementation Guide can serve as a planning aid for a PROFIdrive implementation.

https://kb.hilscher.com/display/PROFIDRIVE/PROFIdrive+Implementation+Guide

3 PROFIdrive model of the application example

Figure 3-1 Overview of PROFIdrive application class AC1 "Speed-controlled drive" Example from /1/ Figure 136



3.1 PROFINET device model according to PROFIdrive (see /1/ section 8.4)

The general PROFINET device model breaks down a device into modules and submodules as well as slots and subslots. Slots and subslots should be considered placeholders, i.e. modules can be plugged into free slots or submodules into free subslots. The structure is hierarchical, i.e. a device contains one or more slots, a module contains one or more subslots.

Slot 0 is the Device Access Point (DAP) and describes the bus interface with its properties. The actual device functionality (e.g. inputs/outputs) is modeled starting with slot 1 in subslots.

Table 3-1 Logical structure of the PROFIdrive P-Device in the application example:

Slot 0 (API = 0)		Slot 1 (API = 0x3A00 PROFIdrive)			
Subslot 0		Subslot 0	Subslot 1	Subslot 2	
			Module Access Point (MAP)	Standard Telegram 1 (submodule ID = PROFIdrive telegram number)	
			Contains parameters Access Point and alarm channel		
P-Device		Drive Object 1			

3.2 Cyclic data (IO AR)

Standard telegrams 1 and 2 are implemented according to PROFIdrive standard /1/. Standard telegram 2 is implemented without a sign-of-life, since the application example does not yet support any isochronous operation.

Table 3-2 Standard telegram 1 from /1/ section 6.3.4.3.2

IO data number	Setpoint	Actual value
1	STW1	ZSW1
2	NSOLL_A	NIST_A

Standard telegram 1 has a size of 4 octets (bytes). STW1, NSOLL_A, ZSW1, NIST_A are 16-bit-data (Word). Input Data and Output Data are each 4 bytes over the same subslot 2.

STW1 is implemented according to "Speed control mode" /1/ section 6.3.2.2. The optional function Jog1/Jog2 is not implemented. No device-specific control bits are implemented.

ZSW1 is implemented in "Speed control mode" as in /1/ section 6.3.2.5. No device-specific status bits are implemented.

NSOLL_A and NIST_A have the data type N2 (0x4000 ≙ 100%, see /1/ section 5.3.2). NIST_A returns the speed of the simulated control system (PT1 element).

Table 3-3 Standard telegram 2 from /1/ section 6.3.4.3.3

IO data number	Setpoint	Actual value
1	STW1	ZSW1
2	NSOLL_B	NIST_B
3		
4	STW2	ZSW2

The standard telegram 2 has a size of 8 octets (bytes). STW1, STW2, ZSW1, ZSW2 are 16-bit data. NSOLL_B, NIST_B are 32-bit data. Input Data and Output Data are each 8 bytes over the same subslot 2.

STW1 and ZSW1 are implemented as in standard telegram 1.

NSOLL_B and NIST_B have the data type N4 (0x40000000 ≙ 100%, see /1/ section 5.3.2). NIST_B returns the speed of the simulated control system (PT1 element).

STW2 and ZSW2 are implemented without a sign of life and without manufacturer-specific data. They are therefore currently fully unused.

3.3 Parameter channel (BMP protocol via Record Data ASE)

Only the mandatory PAP data record 0xB02E for the "Base Mode Parameter Access - Local" is implemented. The optional data record 0xB02F for the "Base Mode Parameter Access - Global" is not implemented (is also not recommended). Other data records are not defined by PROFIdrive and have not been implemented.

Three parameter channels are implemented. This allows 2 IO controllers and 1 IO supervisor (device access) to be accessed simultaneously via separate ARs. The scope of 2 IO controllers and 1 IO supervisor is defined by the PROFINET-IO application example on which the PROFIdrive application example is based.

Table 3-4 List of implemented parameters:

PNU	Parameter description	Wrt.	Impl.	Vali.
100 -200	Examples of manufacturer-specific parameters for the configuration of the application (Parameters for ramp generator, threshold values, gain factors of the control system)	rw	V	L
300	Exemplary parameter for selecting the source for the telegram selection (PROFINET configuration or PROFIdrive parameter PNU00922)	rw	V	L
700 -819	Exemplary manufacturer-specific parameters for tests with the PROFIdrive PROFINET Profile tester	rw	V	L
900	Setpoint telegram (DO IO DATA)	ro	O	L
907	Actual value telegram (DO IO DATA)	ro	O	L
922	Telegram selection	rw	M	L
944	Fault message counter	ro	M	L
945	Fault code	ro	O	L
947	Fault number	ro	M	L
948	Fault time	ro	O	L
949	Fault value	ro	O	L
950	Scaling of the fault buffer	ro	O	L
951	Fault number list with text	ro	O	L
952	Fault situation counter	rw	O	L
953	Warning parameters	ro	O	L
964	Drive Unit identification data block	ro	M	G
965	Profile identification number	ro	M	L
974	Base Mode Parameter Access identification	ro	O	G
975	DO identification	ro	O	L
980	Number list of defined parameter	ro	M	L
60000	Velocity reference value	ro	M	L
61000	NameOfStation	ro	O	G
61001	IpOfStation	ro	O	G
61002	MacOfStation	ro	O	G
61003	DefaultGatewayOfStation	ro	O	G
61004	SubnetMaskOfStation	ro	O	G

Wrt.: ro – read only, rw – read-/writeable; Impl.: M – mandatory, O – optional, V – vendor specific; Vali.: L – local, G - Global

3.4 Alarms (Alarm ASE)

All alarms are optional according to PROFIdrive. For example, alarms are generated as errors (faults) or warnings. The faults/warnings are set and reset in the application example for the demonstration via parameter PNU00700-PNU00702.

3.5 State machine

The general state machine is implemented according to /1/ section 6.3.3.1.

3.6 Synchronization (Isochronous Mode Application ASE)

Synchronization is not a mandatory component of application class 1 (AC1) and is therefore not implemented in the application example.

3.7 Application

The control system (controller, inverter, motor, encoder) is simulated as a PT1 element.

4 Implementation of the application example in DevKit ERTEC 200P-2

4.1 Overview

The PROFIdrive_AC1 example is implemented according to PROFIdrive V4.2 /1/.

The PROFIdrive_AC1 example requires the installation of the firmware stack for the Evaluation Kit (see /2/ and /3/). It is based on the application example "App1_STANDARD", which is located under `<InstallPath>\pnio_src\application\App1_STANDARD` (see /2/ section 2.4 "Application examples").

To use the PROFIdrive_AC1 example with the Evaluation Kit, the following procedure is recommended:

1. Complete commissioning of the Evaluation Kit as described in /3/ section 4 "Quickstart Guide".
2. Copy the PROFIdrive source files
   `<SRC_PROFIdrive_AC1_example.zip>\pnio_src\application\App41_PROFIdrive_AC1`
   to
   `<InstallPath>\pnio_src\application\App41_PROFIdrive_AC1`
3. The selection of the application example is made in the header file
   `usrapp_cfg.h in`
   `<InstallPath>\pnio_src\application\App_common`
   with the following entry:
   `#define  EXAMPL_DEV_CONFIG_VERSION  41`
4. Create the project as new and load the PROFIdrive_AC1 example into the Evaluation Kit

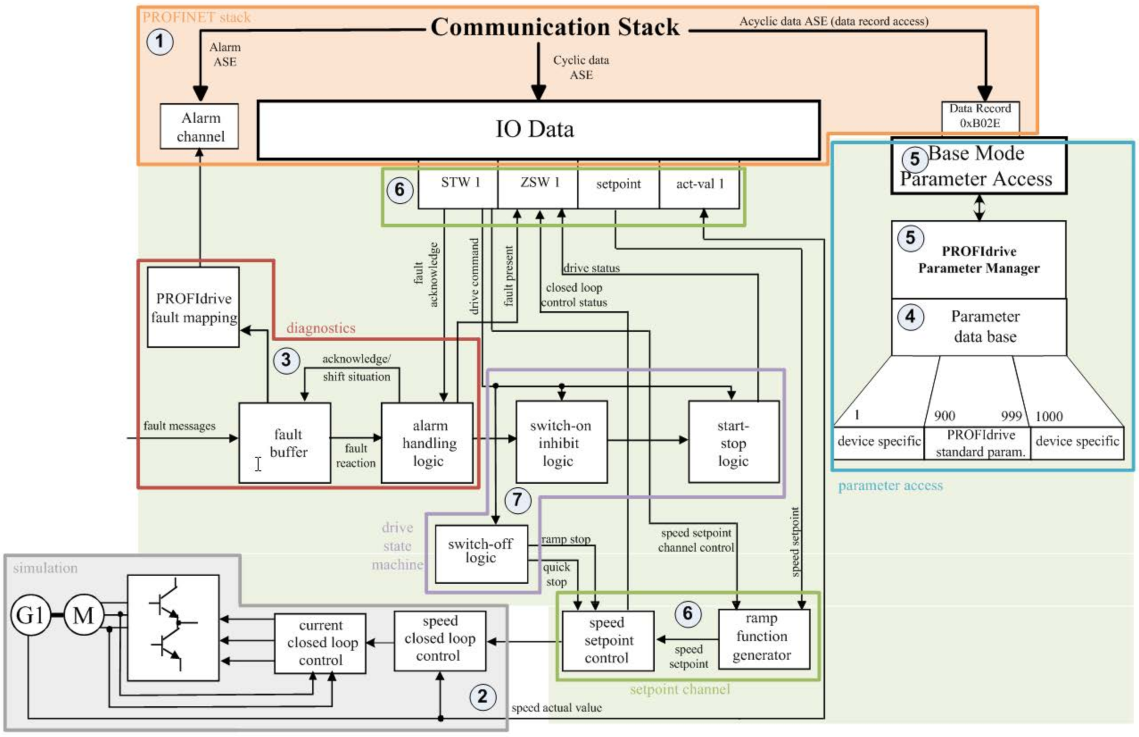
The modified GSDML file for the example is located in `<SRC_PROFIdrive_AC1_example.zip>\GSDML` for the configuration of a PROFINET project.

4.2 Files for App41_PROFIdrive_AC1

Table 4-1 Quick overview of the files of the application example:

File	Brief description	Fig. No.
usriod_main_pdrvac1.c	Main program of the PROFINET example Start of the PROFINET-IO stack, initialization of the PROFIdrive application, initialization and start of the 1ms timer for the PROFIdrive application, main loop with start of functions via keyboard of PROFINET functions (for example, firmware updates)	1
iodapi_event_pdrvac1.c	PROFINET interface with messages from PROFINET events to the application Contains functions which the PROFINET IO stack calls when events such as connection/disconnection, alarm reception, etc. occur and thereby communicates to the application.	1
PnUsr_Api_pdrvac1.c	PROFINET subroutines for the user example. If necessary, the functions contained can be used as a function library in the customer application.	1
pdrv_application.c pdrv_application.h	PROFIdrive application 1ms time slice with call of the other PROFIdrive modules, simulation of the control system, terminal outputs for PROFIdrive events, implementation of the application parameters PNU100-PNU819	2
pdrv_diagnostics.c pdrv_diagnostics.h	PROFIdrive diagnostics fault and warning handling	3
pdrv_parameter.inc	Macro file with list of implemented PROFIdrive parameters	4
pdrv_pardatabase.c pdrv_pardatabase.h	PROFIdrive parameter database Creates the parameter database from pdrv_parameter.inc	4
pdrv_parmanager.c pdrv_parmanager.h	PROFIdrive Parameter Manager Realizes the PROFIdrive Base Mode Parameter Access	5
pdrv_setpointchannel.c pdrv_setpointchannel.h	PROFIdrive setpoint channel Implements access to the signals of standard telegrams 1 & 2; selection of the setpoint source based on the general state, ramp generator	6
pdrv_statemachine.c pdrv_statemachine.h	PROFIdrive general state machine	7
pdrv_types.h	Definitions of PROFIdrive data types	

Figure 4-1 Mapping of the source files to the functionality (see table)



4.3 usriod_main_pdrvac1.c

The module is created based on `<InstallPath>\pnio_src\application\App1_STANDARD\usriod_main.c`. It includes the startup of the PROFINET IO stack, the initialization of the PROFIdrive application, the initialization and start of the 1ms timer for the PROFIdrive application and main loop with the start of PROFINET functions (e.g. firmware updates) via keyboard.

In addition, the functionalities of the PNU61000-PNU61004 parameters are implemented with parameter read routines `uPdrv_RfPnu61000() bis uPdrv_RfPnu61002()`.

4.4 iodapi_event_pdrvac1.c

The module is created based on
`<InstallPath>\pnio_src\application\App1_STANDARD\iodapi_event.c`
. It contains functions which the PROFINET IO stack calls up when events such as connection/disconnection, alarm reception, etc. occur and thereby communicates to the application.

In the `PNIO_cbf_data_write()` function, the actual values ZSW1, ZSW2, NIST_A, NIST_B are transferred from the PROFIdrive setpoint channel to the PROFINET stack. This handles the different byte order (Endianess, ERTEC Little Endian, PROFINET Big Endian).

In the `PNIO_cbf_data_read()` function, the setpoints STW1, STW2, NSOLL_A, NSOLL_B are transferred from the PROFINET stack to the PROFIdrive setpoint channel. This handles the different byte order (Endianess, ERTEC Little Endian, PROFINET Big Endian).

In the `PNIO_cbf_ar_connect_ind()` function, the establishment of the PROFINET-AR connection is communicated to the PROFIdrive Parameter Manager, which then reserves a parameter channel for this connection.

In the `PNIO_cbf_ar_disconn_ind()` function, the PROFINET-AR disconnection is communicated to the PROFIdrive Parameter Manager, which then closes the parameter channel for this connection and releases the parameter channel.

In the `PNIO_cbf_ar_indata_ind()` function, the initial reception of setpoints from the PROFINET stack is reported to the PROFIdrive setpoint channel.

In the `PnUsr_cbf_rec_read()` function, the read request of the PROFIdrive Base Mode Parameter Access Record is passed to the PROFIdrive Parameter Manager, which processes the read request accordingly and supplies the contents of the record.

In the `PnUsr_cbf_rec_write()` function, the write request PROFIdrive Base Mode Parameter Access Record is passed to the PROFIdrive Parameter Manager, which processes the write request accordingly and returns a corresponding response.

The functions of the parameters PNU00300 "Source telegram selection", PNU00922 "Telegram selection", PNU00900 "Setpoint telegram" and PNU00907 "Actual value telegram" are implemented. The source for the telegram selection is set in parameter PNU00300. For PNU00300 = 0 (default value), the telegram selection according to the PROFINET configuration (Expected Configuration Data) is performed in the `PNIO_cbf_data_write()` function. For PNU00300 = 1, the telegram selection is made via the PNU00922 parameter in the parameter write routine `uPdrv_WfPnu00922()`.

4.5 pdrv_application.c

The module is an example of the drive application. It consists of the PdrvApp_main() function, which is called cyclically every 1ms by the operating system. This function cyclically calls the PROFIdrive functions such as PROFIdrive general state machine, ramp generator, threshold value calculations, setpoint value channel, control system simulation, actual value transfer and Parameter Manager. For demonstration purposes, calculated data are output and can be displayed in a terminal program via the serial interface of the Evaluation Kit.

The module also implements application-dependent PROFIdrive functions such as standstill detection `bPdrvApp_IsAxisStandstill()`, transition acknowledgment for the PROFIdrive general state machine `bPdrvApp_IsTransitionCondition()` and a 1s timer `uPdrvApp_GetTimer1s()`.

Furthermore, the parameter read routines and parameter write routines for the application parameters and for the test parameters are implemented for testing with the profile tester.

4.6 pdrv_diagnostics.c

The module implements PROFIdrive diagnostic mechanisms (see /1/ section 6.3.8 "Diagnostics"). Both the PROFIdrive "Complete Fault Buffer" (see /1/ Table 114 "Fault buffer parameters") as well as the PROFIdrive warnings (see /1/ section 6.3.8.2 "Warning mechanism") are implemented.

Table 4-2 List of implemented parameters from "Fault Buffer" and "Warning mechanism":

PNU	Parameter description	Wrt.	Impl.	Vali.
944	Fault message counter	ro	M	L
945	Fault code	ro	O	L
947	Fault number	ro	M	L
948	Fault time	ro	O	L
949	Fault value	ro	O	L
950	Scaling of the fault buffer	ro	O	L
951	Fault number list with text	ro	O	L
952	Fault situation counter	rw	O	L
953	Warning parameters	ro	O	L

Wrt.: ro – read only, rw – read-/writeable; Impl.: M – mandatory, O – Optional; Vali.: L – local, G - Global

The `PdrvDiag_SetFaultMsg()` function is used to set a fault and entry in the PROFIdrive fault buffer. A corresponding message is also generated via the standard PROFINET alarm mechanism. The `PdrvDiag_AckFaultSit()` function acknowledges a fault and is called with a positive edge of the STW1 bit 7 and when the PNU00952=0 parameter is reset (see /1/ Figure 61 "Fault acknowledgment for the fault buffer mechanism").

The `bPdrvDiag_IsFaultOff1()`, `bPdrvDiag_IsFaultOff2()`, `bPdrvDiag_IsFaultOff3()` functions return whether a fault is active with the corresponding fault reaction "Power down", "Coast stop" oder "Quick stop".

The `PdrvDiag_SetWarning()` function is used to set and reset a warning. A corresponding message is also generated via the standard PROFINET alarm mechanism.

Furthermore, the parameter read routines and parameter write routines are implemented for the parameters listed above.

4.7 pdrv_pardatabase.c

This module implements the parameter database, in which the parameters defined in `pdrv_parameter.inc` file are created in C structures using macros.

The `ptPdrvPar_GetParObj()` The function searches for a parameter object in the parameter database and returns a pointer to the parameter object if it is successful.

In addition to the properties listed in /1/ section 6.2.1.3 "Parameter description", a parameter object contains 3 different access functions. The parameter read routine is called when the parameter access is read and fills the response buffer with the requested current parameter values. The write routine is called during write access and gets the written values from the job buffer and enters these into the memory associated with the parameter. The text routine is called when an additional text is requested (see /1/ section 6.2.1.4 "Text") and returns a pointer to the requested text.

Furthermore, the parameter read routines and parameter write routines are implemented for various parameters.

4.8 pdrv_parmanager.c

This module implements the PROFIdrive Base Mode Parameter Access (see /1/ Figure 131).

The `bPdrvPar_EstablishCxn()` function is called when a PROFINET connection is established. This checks whether a PROFIdrive parameter channel is available for the PROFINET connection and reserves a PROFIdrive parameter channel for further use with this PROFINET connection. The PROFIdrive parameter channel is reset. If the same PROFINET connection is closed, the `bPdrvPar_DisconnCxn()` function is called and the PROFIdrive parameter channel is also closed and released for later use. Three PROFIdrive parameter channels are available simultaneously (configurable in `PARCXN_NR` constant). The restriction to 3 PROFIdrive parameter channels is based on the restriction of the PROFINET Evaluation Kit EK200P-2 V4.4 used. This allows a maximum of 3 simultaneous PROFINET connections to be maintained, of which there are a 2 controller connections maximum as well as one device access connection.
Note: With firmware V4.5 of the EK 200P-2 Evaluation Kit, up to 5 simultaneous PROFINET connections can be maintained, of which there are 4 controller connections maximum as well as one device access connection.

When receiving record requests for the Base Mode-Parameter Access Local Record (0xB02E), the `uPdrvPar_WriteReqCxn()` or `uPdrvPar_ReadReqCxn()` functions are correspondingly called, which generate the response to this record request according to the state of the associated PROFIdrive parameter channel (see /1/ Table 33). With a valid Write.req, the PROFIdrive parameter request is transferred to the request buffer for further processing. With a valid Read.req, the response is taken from the response buffer.

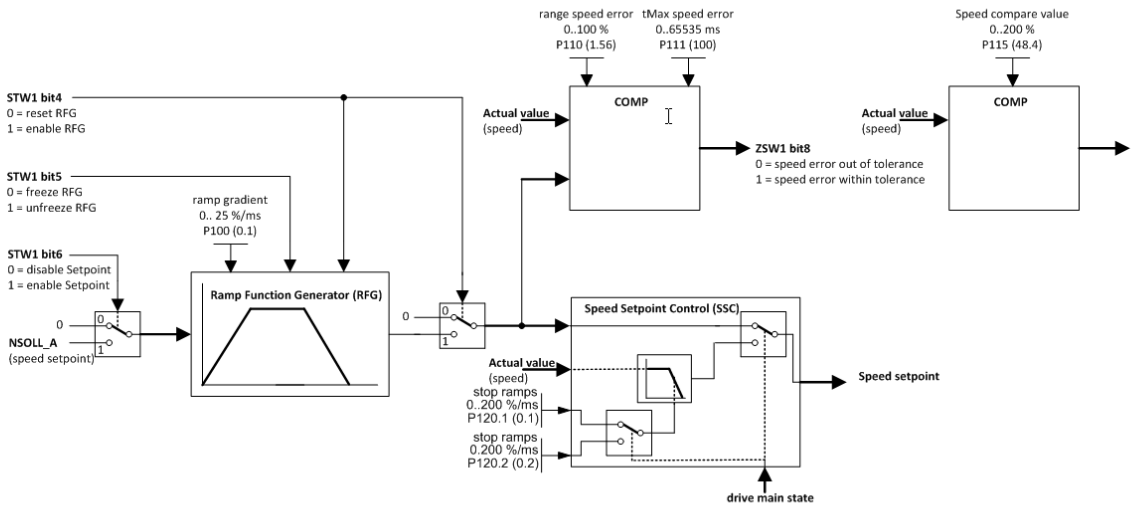
The `PdrvPar_ProcessReq()` function is called cyclically. With each call, the function checks if there is a parameter request to be processed in one of the parameter channels, the parameter request is processed and the result stored in the response buffer. If there are parameters to be processed in several parameter channels, the parameter requests are processed sequentially according to the round robin method.

The different byte order (Endianess, ERTEC Little Endian, PROFINET Big Endian) is handled when processing the parameter requests.

4.9 pdrv_setpointchannel.c

This module implements the setpoint channel. The cyclical setpoints of the standard telegrams are stored here for further use and are made available with get functions.

Figure 4-2 Ramp generator from /1/ Figure 29 (no jog functionality)



The ramp generator (see /1/ Figure 29) is implemented without jogging functionality in the `nPdrvSpc_CalcRfg()` function. The `nPdrvSpc_CalcSsc()` function implements speed setpoint control which, depending on the state of the general state machine, either passes through the ramp generator setpoint unchanged or realizes an OFF ramp with normal or rapid stop speed.

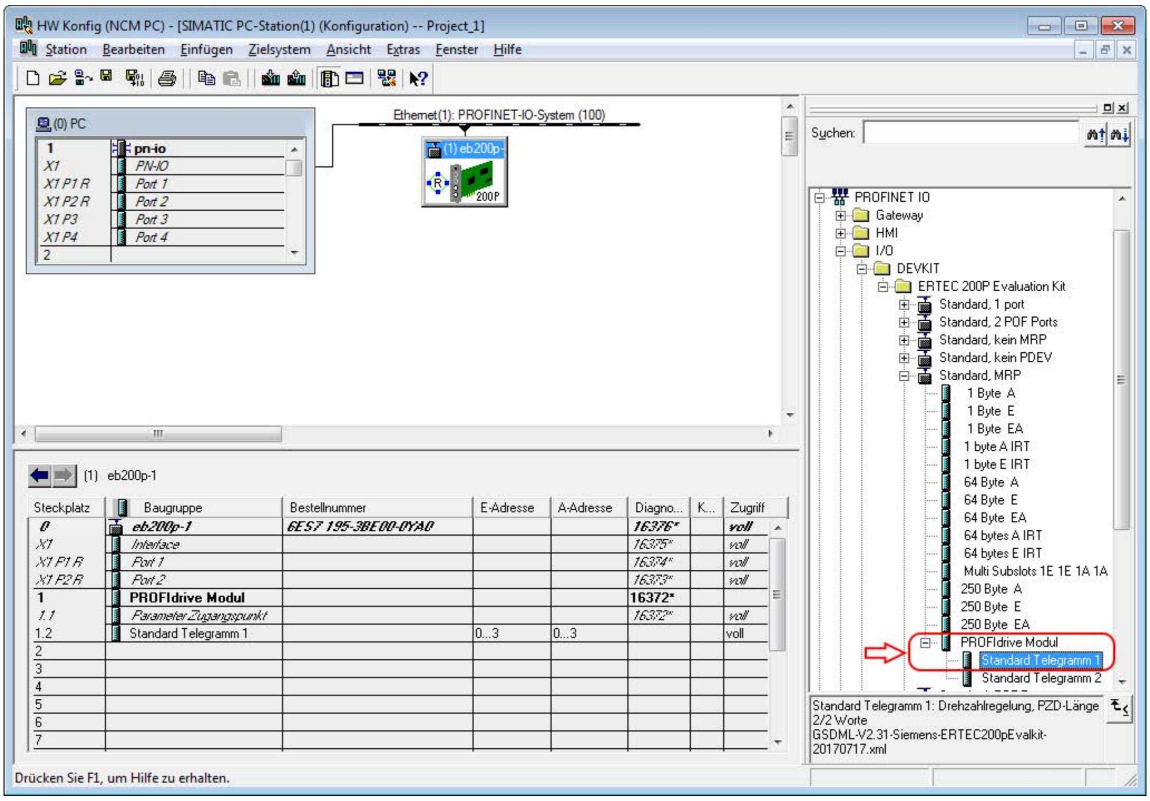
4.10 pdrv_statemachine.c

The `ePdrvSma_AxisGeneralStateMachine()` function implements the general state machine that is described in /1/ Figure 27 with the state graphs. The current status of the general state machine can be queried with the `ePdrvSma_GetAxisMainState()` function.

4.11 GSDML file

The PROFIdrive module was added to the GSDML file of the EK-ERTEC 200P-2 PN IO V4.4.0 Evaluation Kit. The PROFIdrive module thus appears in the same category as the other IO modules of the Evaluation Kit.

Figure 4-3 Example for hardware configuration in the program NCM PC (Profile Tester)



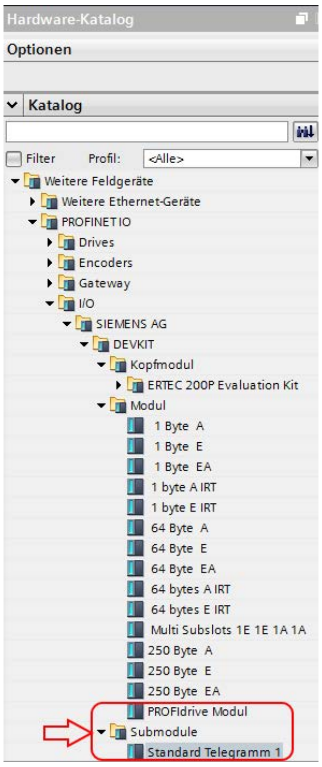
(1) eb200p-1

Steckplatz	Baugruppe	Bestellnummer	E-Adresse	A-Adresse	Diagno...	K...	Zugriff
0	***eb200p-1***	***6ES7 195-3BE00-0YA0***			***16376****		***voll***
X1	*Interface*				*16375**		*voll*
X1 P1 R	*Port 1*				*16374**		*voll*
X1 P2 R	*Port 2*				*16373**		*voll*
1	**PROFIdrive Modul**				**16372***		
1.1	*Parameter Zugangspunkt*				*16372**		*voll*
1.2	Standard Telegramm 1		0...3	0...3			voll
2							
3							
4							
5							
6							
7							

Standard Telegramm 1: Drehzahlregelung, PZD-Länge 2/2 Worte
GSDML-V2.31-Siemens-ERTEC200pEvalkit-20170717.xml

Drücken Sie F1, um Hilfe zu erhalten.

Figure 4-4 PROFIdrive example in the hardware catalog of the TIA Portal



5 Adaptation of the application example

5.1 Identification data

In addition to the changes to the PROFINET stack described in /3/ section 7.2.1, the following identification data must be adapted in the application example.

The `PDRV_MODULE_ID_MAP` module ID must be configured in `pdrv_application.h` (manufacturer-specific).

The identification data must be configured in the `pdrv_application.h` file (for more on this, see parameters PNU00964, PNU00975 in /1/ Table 118 and Table 120)

Table 5-1Identification data

Constant	Description
`PDRV_ID_MANUFACTURER`	Manufacturer code according to PNO
`PDRV_ID_DUTYPE`	Drive Unit Type (manufacturer-specific)
`PDRV_ID_DOTYPE`	Drive Object Type (manufacturer-specific)
`PDRV_ID_FWVERSION`	Firmware version
`PDRV_ID_FWDATE_Y`	Year of the firmware date
`PDRV_ID_FWDATA_DM`	Day and month of the firmware date

5.2 Diagnostics

The faults and warnings implemented as examples should be removed and the faults and warnings occurring in the new application should be implemented.

This involves

- Removing the entries `FAULT_USER0` bis `FAULT_USER9` and `WARNING_USER0 to WARNING_USER9` in the `pdrv_diagnostics.*` file
- Removing the parameters PNU00700 to PNU00702 in the `pdrv_parameter.inc` file and the associated implementations (tags, functions) in `pdrv_application.c` (example for testing with the profile tester)
- New warning must be assigned to a fault class of the PROFINET standard alarm functionality and entered in the `m_tWarningAttribs[]` table. For more than 16 warnings, the PNU00954 to PNU00960 parameters must be implemented.
- New faults are stored in the `m_tFaultAttribs[]` table.
  The following must be assigned to a fault for this:
  * A fault code (PNU00945)
  * A fault number (PNU00947)
  * A fault reaction (OFF1 ≙ Motor OFF, OFF2 ≙ Pulse lock, OFF3 ≙ Rapid stop, none)
  * A fault class of the PROFINET standard alarm functionality
  * A fault text (16 characters, PNU00951)
  .
- The `PdrvDiag_SetFaultMsg()` and `PdrvDiag_SetWarning()` must correspondingly called in new application

It is required to check whether a new application requires a consistency check during the diagnostic processing. In the example application, no consistency protection is required since all software components are processed sequentially with the same priority.

5.3 Parameters

The parameters implemented as examples must be removed and the parameters occurring in the new application must be implemented.

This involves

- Removing the parameters PNU00100 to PNU00200 in the pdrv_parameter.inc file and the associated implementations (tags, functions) in pdrv_application.c (parameters of the example application)
- Removing the parameters PNU00700 to PNU00819 in the pdrv_parameter.inc file and the associated implementations (tags, functions) in pdrv_application.c (parameters for testing with the profile tester)
- Removing the parameter PNU00300 in the pdrv_parameter.inc file and selecting one of the two variants for the telegram selection in iodapi_event_pdrvac1.c If possible, the telegram should be selected according to the PROFINET configuration. In this case, the PNU00922 parameter only displays the configured telegram.
- The parameters PNU00900 to PNU00999 in the pdrv_parameter.inc file must be adapted to the changed conditions.
- The application-specific parameters must be implemented for the new application.

Each parameter requires a read routine pfnRead(). Writable parameters also require a write routine pfnWrite(). If a parameter contains additional texts, a text routine pfnText() is required.

Note: It is recommended to generate the parameter database automatically from a separate parameter database. The macro method selected in the example application makes it difficult to find syntax errors and formal errors in the parameter database, which is implemented in the pdrv_pardatabase.c file.

The example application did not implement retentive storage for parameters and therefore also does not implement an explicit reset to the factory state.

The following additional setting options for parameter processing are provided:

Table 5-2Additional setting options

Constant	Description
`PDRV_PAR_BLOCKSIZE`	Data block size of a parameter request/response (see /1/ section 6.2.3.2)
`PARCXN_NR`	Number of parameter channels (simultaneous possible parameter connections). The PROFINET stack must be adapted accordingly if there is a change.

5.4 Application

The example application in the `pdrv_application.c` file consists of a very simple simulation of the full control system (actuator, motor, measured value acquisition) using a PT1 element. This simulation is to be replaced by your own application.

The application must call the following PROFIdrive functions in the correct order with correct parameters:

- General state machine `ePdrvSma_AxisGeneralStateMachine()`
- Ramp generator `nPdrvSpc_CalcRfg()`
- Setpoint channel `nPdrvSpc_CalcSsc()`
- Tolerance calculation for ZSW1 bit 8 `bPdrvSpc_CalcSpeedWithinTolerance()`
- Comparison for ZSW1 bit 10 `bPdrvSpc_CalcSpeedReached()`
- Actual value transfer to standard telegram 1 `PdrvSpc_SetNistA()`
- Parameter manager `PdrvPar_ProcessReq()`

Typically, with the exception of the parameter manager, the functions are executed in a constant bus cycle with the same priority as the actual drive application. The parameter manager is often executed with a lower priority and not necessarily with a constant bus cycle. In the present example, all functions with the same priority are processed every 1 ms, which is why consistency checks were omitted in the example. If an application is implemented with different execution priorities, appropriate measures (interrupt locks, alternating buffers) must be implemented at appropriate points for consistency.

Furthermore, the following functions must be implemented:

- Standstill detection `bPdrvApp_IsAxisStandstill()`
  (used in the general state machine)
- Transition acknowledgment `bPdrvApp_IsTransitionCondition()`
  (used in the general state machine)
- 1s-Timer `uPdrvApp_GetTimer1s()`
  (used with fault buffer PNU00948, optional)

The transition acknowledgment `bPdrvApp_IsTransitionCondition()` is used to delay the state propagation of the general state machine by the application, if the application is not yet ready for the subsequent state. This is in each case for the general state transition from S1 Switch-on inhibit to S2 Ready for switch-on, from S2 Ready for switch-on to S3 Ready for operation, from S3 Ready for operation to S4 Operation (see /1/ Figure 27 "General State Diagram").

5.5 Communication stack

When changing the communication stack, the modules `iodapi_event_pdrvac1.c` and `usriod_main_pdrvac1.c` must be adapted or replaced.

PROFIdrive uses/requires the following functions:

- Communication connection `PNIO_cbf_ar_connect_ind()`
- Communication disconnection `PNIO_cbf_ar_disconn_ind()`
- Detection of the first successful communication exchange `PNIO_cbf_ar_indata_ind()`
- Send cyclic data `PNIO_cbf_data_write()`
- Receive cyclic data `PNIO_cbf_data_read()`
- Read data record `PnUsr_cbf_rec_read()`
- Write data record `PnUsr_cbf_rec_write()`
- Channel diagnostics `uPdrvUsr_ChanDiag()`
- Pulling and plugging a `uPdrv_WfPnu00922()` submodule (when using multiple standard telegrams, optional)
- PROFINET and TCP/IP station data `uPdrv_RfPnu61000()` - `uPdrv_RfPnu61002()` (optional)

In the modules `iodapi_event_pdrvac1.c` and `usriod_main_pdrvac1.c`, the code locations for PROFIdrive can be easily found by searching for the string "Pdrv".

When changing the processor platform or the communication stack, please note that the ERTEC 200P works in the Little Endian format Evaluation Kit and that PROFINET uses Big Endian format.

6 Appendix

6.1 Terms / Abbreviations

API	Application Interface
API	Application Process Identifier
AC	Application Class (PROFIdrive)
ASE	Application Service Element (PROFIdrive)
DAP	Drive Access Point (PROFINET)
DO	Drive Object (PROFIdrive)
DU	Drive Unit (PROFIdrive)
MAP	Module Access Point (PROFIdrive)
PAP	Parameter Access Point (PROFIdrive)
P-Device	Peripheral device (PROFIdrive)
PNU	Parameter NUmber (PROFIdrive)

6.2 References

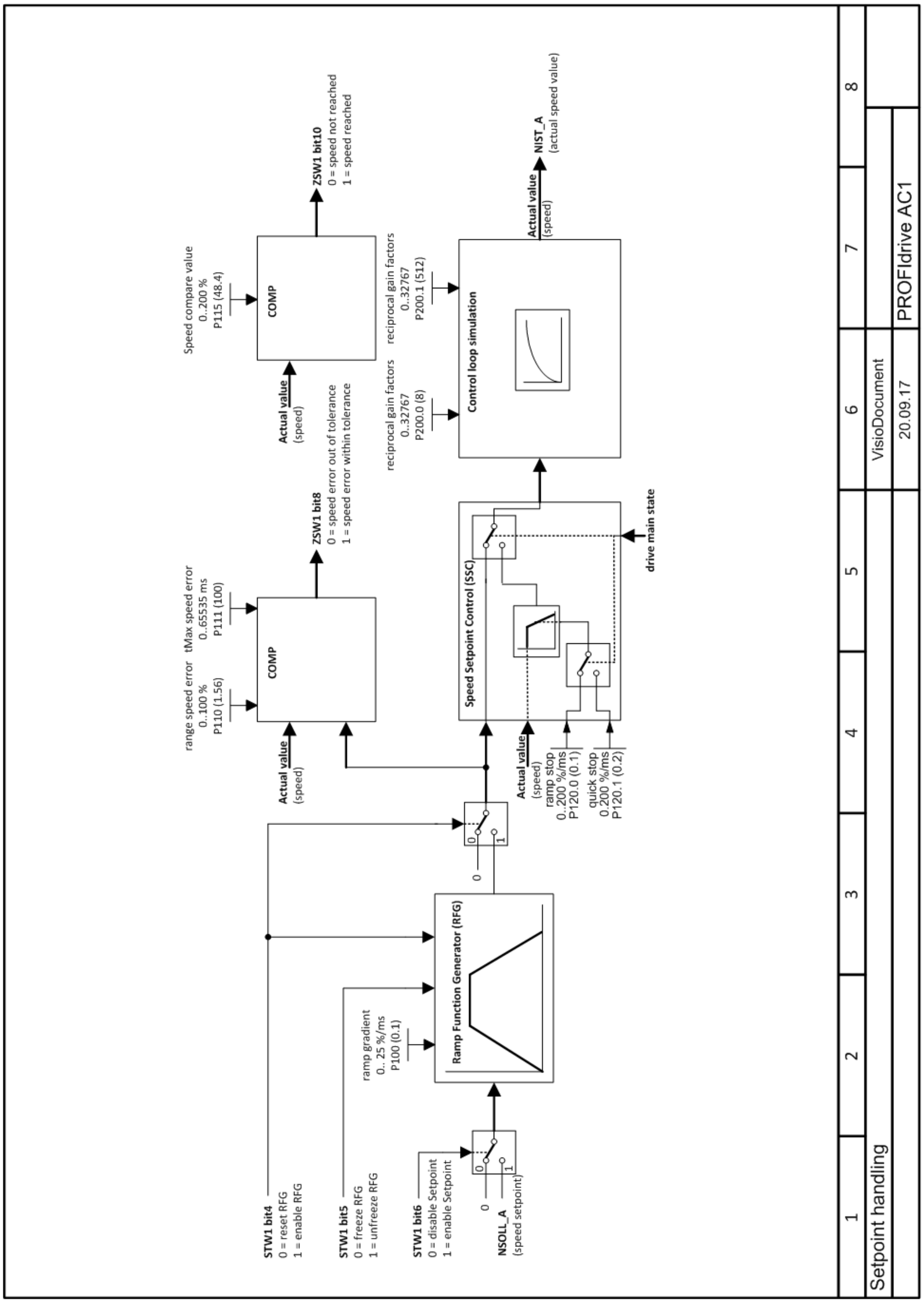
/No./ "Author": "Title",
"Publisher", "Place of publication", "Year", "Page specification"

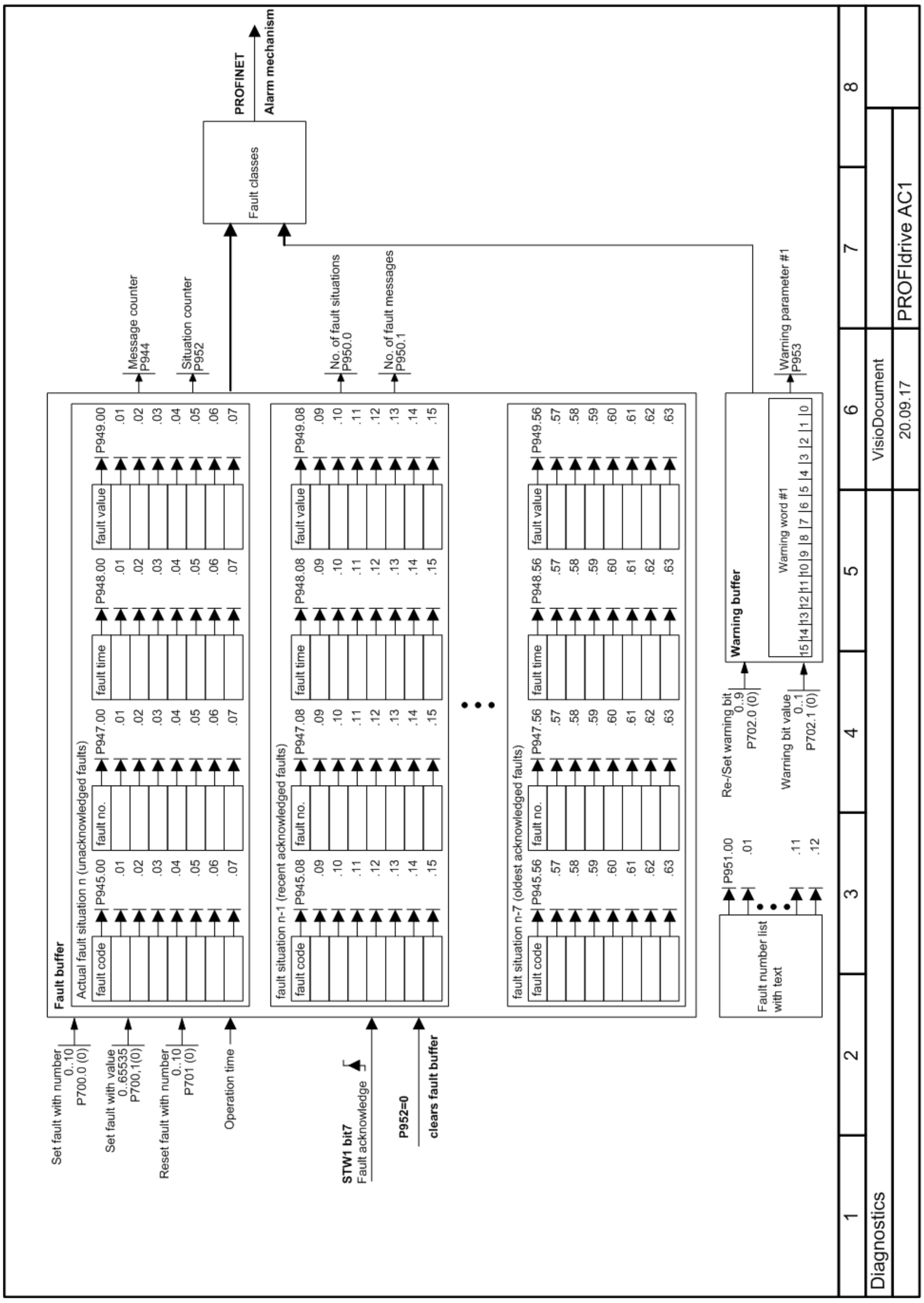
/1/ PROFIBUS User Organization Profile Drive Technology PROFIdrive Version 4.2 October 2015

/2/ Siemens AG: "Interface description PROFINET IO Development Kits V4.4.0" Programming and Operating Manual, A5E33638878-AC, 11/2016

/3/ Siemens AG: "Guidelines for Evaluation Kit ERTEC 200P-2 V4.4.0" Programming and Operating Manual, A5E03855331-AC, 11/2016

6.3 Function charts




Fault buffer
Actual fault situation n (unacknowledged faults)
fault code
P945.00
.01
.02
.03
.04
.05
.06
.07
fault no.
P947.00
fault time
P948.00
fault value
P949.00
fault situation n-1 (recent acknowledged faults)
P945.08
.09
.10
.11
.12
.13
.14
.15
P947.08
P948.08
P949.08
fault situation n-7 (oldest acknowledged faults)
P945.56
.57
.58
.59
.60
.61
.62
.63
P947.56
P948.56
P949.56
Set fault with number
0..10
P700.0 (0)
Set fault with value
0..65535
P700.1(0)
Reset fault with number
0..10
P701 (0)
Operation time
STW1 bit7
Fault acknowledge
P952=0
clears fault buffer
Message counter
P944
Situation counter
P952
No. of fault situations
P950.0
No. of fault messages
P950.1
Fault classes
PROFINET
Alarm mechanism
Fault number list with text
P951.00
.01
.11
.12
Re-/Set warning bit
0..9
P702.0 (0)
Warning bit value
0..1
P702.1 (0)
Warning buffer
Warning word #1
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Warning parameter #1
P953
1
2
3
4
5
6
7
8
Diagnostics
VisioDocument
20.09.17
PROFIdrive AC1