Programming Instructions |
Programmieranleitung

English | Deutsch

TIM-PN

with Siemens S7-1500 Controller
mit Siemens S7-1500 Steuerung

HBM

Hottinger Baldwin Messtechnik GmbH
Im Tiefen See 45
D-64293 Darmstadt
Tel. +49 6151 803-0
Fax +49 6151 803-9100
info@hbm.com
www.hbm.com

Mat.:
DVS: A05016_01_X00_00 HBM: public
07.2018

Version: 1.0

Programming Instructions |
Programmieranleitung

English | Deutsch

TIM-PN

with Siemens S7-1500 Controller

HBM

1 Reference documents

Ref.	Title	Vers.	Path
[1]	Hans Berger, Siemens (ed.): Automation with SIMATIC S7-1500	2014	ISBN 978-3-89578-403-3
[2]	Montageanleitung_TIM-PN.pdf	n.a.	https://www.hbm.com/fileadmin/mediapool/hbmdoc/technical/a4341.pdf (14.09.2017)
[3]	profinet_step7_v14_function_manual_de-DE_de-DE.pdf	2014	https://www.siemens.de/Digital- Factory/download/EventDocs/profinet_step7_v14_function_manual_de-DE_de-DE.pdf (15.09.2017)
[4]	Von_PROFIBUS_DP_nach_PROFINET_IO_de-DE.pdf	2010	https://support.industry.siemens.com/cs/attachments/19289930/Von_PROFIBUS_DP_nach_PROFINET_IO_de-DE.pdf?download=true (20.09.17)

2 List of abbreviations

API	Application Process Identifier
AR	Application Relation
CPU	Central Processing Unit
DB	Data module
DP	Distributed Peripherals
LPO	Load Process Output
IE	Industrial Ethernet
OB	Organization module
PIO	Peripherals Image of Outputs
PII	Peripherals Image of Inputs
PD	Programming Device
PLC	Programmable Logic Controller
PN	Profinet
SCL	Structured Control Language
SFB	System block
PLC	Programmable Logic Controller
USI	User Structure Identifier

3 Description

This document describes the procedure for reading TIM-PN-specific PROFINET IO diagnosis information, cyclically reading measured values and control values, reading and setting parameter sets, and cyclically modifying the control byte via Siemens PLC. An S7-1500 is used to demonstrate.

Limitation: This application example does not describe system diagnosis and error events of the control unit. There are examples from Siemens with additional links related to this topic for further diagnostic tasks in the user program:

https://support.industry.siemens.com/cs/document/98210758/diagnose-im-anwenderprogramm-mit-s7-1500?dti=0&lc=de-WW (14.09.2017)

https://support.industry.siemens.com/cs/document/24000238/profinet-io-%E2%80%93- diagnoseverarbeitung-im-anwenderprogramm?dti=0&lc=de-ww (22.09.2017)

4 Construction

4.1 Components used

Components	Article number	Notice
CPU 1516-3 PN/DP	6ES7 516-3AN01-0AB0	It is also possible to use any other S7-1500 CPU
TIM-PN		

4.2 Hardware structure

The control unit and TIM-PN are wired directly together with each other as a Profinet subnet (one possibility is shown in *Fig. 4.2*). The PD/PC is connected to the free port of the PLC. The web service of TIM-PN is also connected to the PC via a second network card. If this is not possible, a switch can also be used. Make certain in this case that the web server and Profinet of the TIM-PN are not wired via a switch (*see Fig. 4.1*).

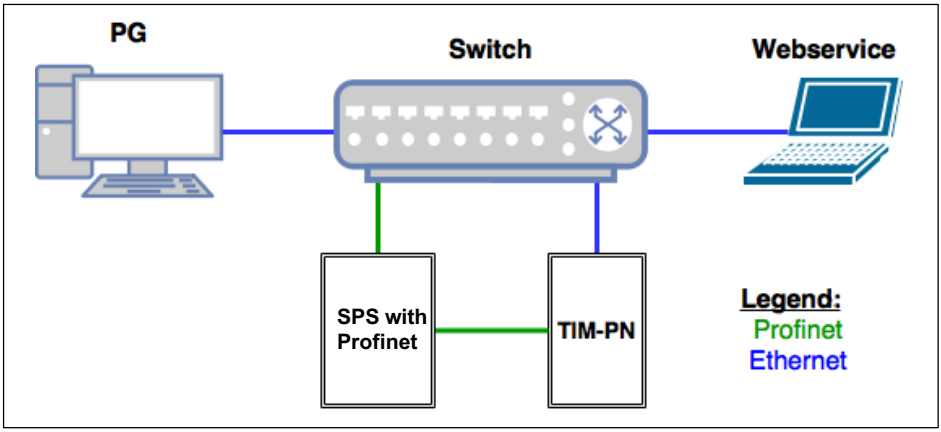

Fig. 4.1 Wiring variant with switch

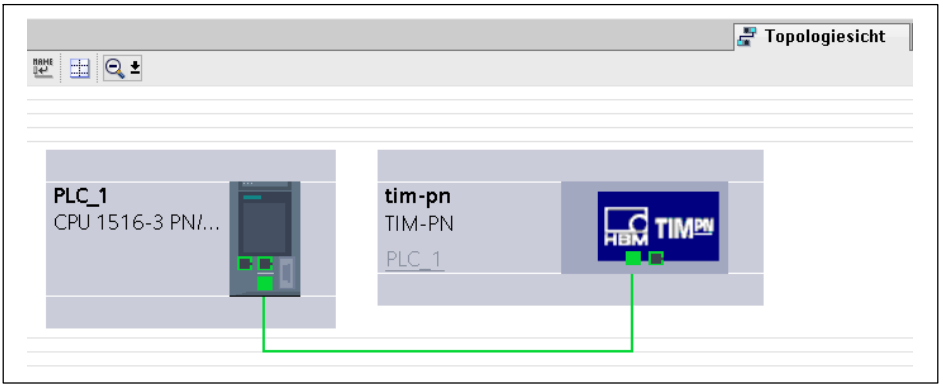

Fig. 4.2 Profinet topology with S7-1500 and TIM-PN

4.3 Addressing

Components	IP address	Netmask
PLC_1: X1	PN: 192.168.1.23	255.255.255.0
tim-pn: X6	PN: 192.168.1.13	255.255.255.0
PLC_1: X2	IE: 192.168.0.1	255.255.255.0
PD	IE: 192.168.0,241	255.255.255.0
tim-pn: X5	IE: 192.168.1.2	255.255.255.0
PC	IE: 192.168.1.254	255.255.255.0

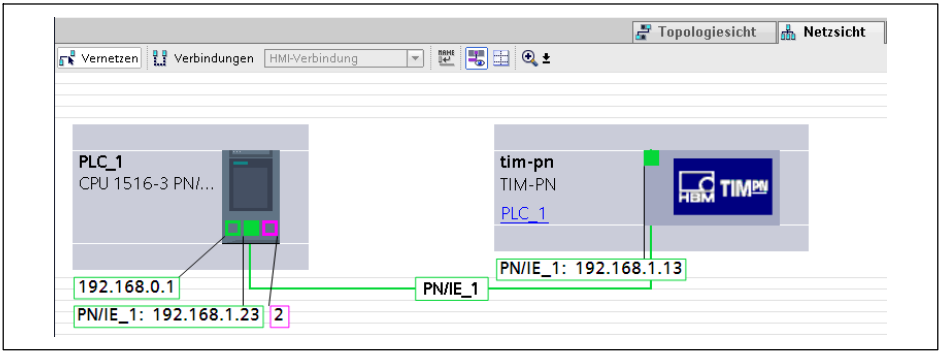

Fig. 4.3 Networking with IP addresses

4.4 Device overview

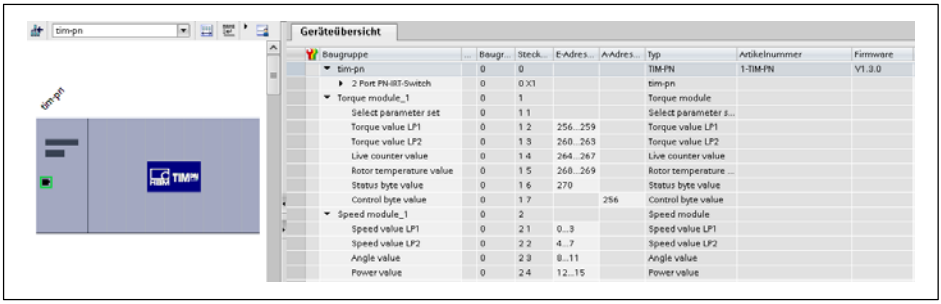

Fig. 4.4 Device overview of TIM-PN

4.5 Variables table

Standard variables table		
Angle_value	DInt	%MD18
Angle_value_changed	Real	%MD118
Angle_value_peripherie	DInt	%ID8
Control_byte_value	Byte	%QB256
Control_byte_value_eingeben	Byte	%MB256
Life_counter_value	DInt	%MD26
Life_counter_value_changed	DInt	%MD126
Life_countervalue_peripherie	DInt	%ID264
Power_value	DInt	%MD22
Power_value_changed	Real	%MD122
Power_value_peripherie	DInt	%ID12
Rotor_temperature_value	Word	%MW30
Rotor_temperature_value_changed	Real	%MD130
Rotor_temperature_value_peripherie	Word	%IW269
Set_Shunt_ON	Bool	%M256.3
Set_Zero_Angle	Bool	%M256.2
Set_Zero_Torque	Bool	%M256.1
Speed_value_LP1	DInt	%MD10

Standard variables table		
Speed_value_LP1_changed	Real	%MD110
Speed_value_LP1_peripherie	DInt	%ID0
Speed_value_LP2	DInt	%MD14
Speed_value_LP2_changed	Real	%MD114
Speed_value_LP2_peripherie	DInt	%ID4
Status_byte_value_peripherie	Byte	%IB270
Status_Request_Shunt_On	Bool	%M16.3
Status_Shunt_On	Bool	%M16.2
Status_Zero_Setting_AoR_active	Bool	%M16.1
Status_Zero_Setting_Torque_active	Bool	%M16.0
Torque_value_LP1	DInt	%MD1
Torque_value_LP1_changed	Real	%MD102
Torque_value_LP1_peripherie	DInt	%ID256
Torque_value_LP2	DInt	%MD6
Torque_value_LP2_changed	Real	%MD106
Torque_value_LP2_peripherie	DInt	%ID260

5 Required function blocks

"RDREC" and "WRREC" are asynchronously working instructions, i.e. processing extends over multiple calls. To start the transfer of data sets, set REQ = 1. The status of the task is indicated by the output parameter BUSY and the middle two bytes of output parameter STATUS. (See Siemens help/information system)

Information

The interfaces of the instructions are identical to the standard "PROFIBUS Guideline, PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3"

The following modules are also provided for migration of S7-300/400 programs:

- RD_REC
- WR_REC

5.1 SFB52 "RDREC" (Read Record)

With this instruction you read the data set with the number INDEX from the component addressed with the ID (hardware ID). Specify with MLEN the maximum number of bytes you would like to read. The length chosen for the RECORD target region should therefore be at least MLEN bytes long. The TRUE value of output parameter VALID indicates that the data set has been successfully transferred to the RECORD target region. In this case the output parameter LEN contains the length of the data that was read in bytes. If an error occurred while the data set was being transferred, this is indicated by output parameter ERROR. In this case the output parameter STATUS contains the error information.

If an error occurs and the second byte in the STATUS contains a C0 (read constrain conflict), the requested length in the Read Request could be too small. A larger value should be chosen for MLEN. The length in the Read Request can also normally be much greater than the actual length of the diagnostic data. Then only as much as is actually present will be returned.

Information

If a DPV1 slave is projected with a GSD file (GSD rev. 3 or later) and the DP interface of the DP master is set to "S7-compatible", no data sets with "RDREC" may be read from the I/O modules in the user program. In this case the DP master addresses the wrong slot (projected slot + 3).
Remedy: Change the interface of the DP master to "DPV1".

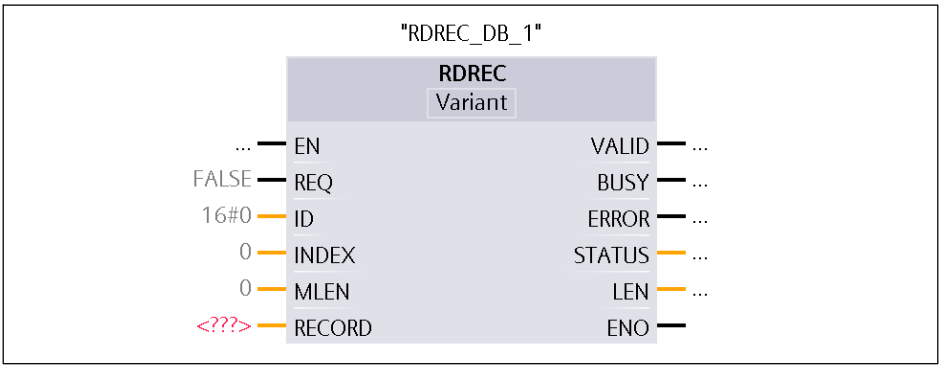

Fig. 5.1 Siemens function block ->SFB52

5.2 SFB53 "WRREC" (Write Record)

With this instruction you transfer the data set RECORD to the component addressed with the ID (hardware ID). With LEN you specify the length of the data set being transferred in bytes. The RECORD source region should therefore be chosen so it is at least LEN bytes long. The TRUE value of output parameter DONE indicates that the data set has been successfully transferred. If an error occurred while the data set was being transferred, this is indicated by output parameter ERROR. In this case the output parameter STATUS contains the error information.

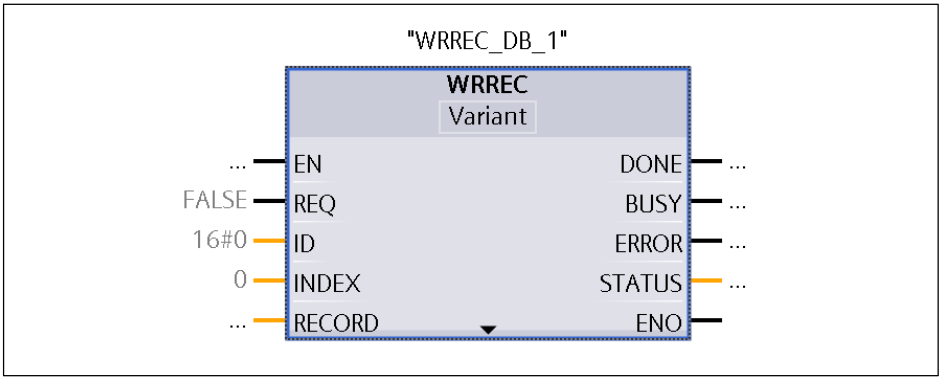

Fig. 5.2 Siemens function block ->SFB53

6 Loading and processing measured values

6.1 Program sequence

The program is processed cyclically in OB1 according to the LPO process. This requires the PLC to have been successfully started and the CPU to be in RUN. A cycle can be divided into five points:

- Start of cycle time monitoring
- Load all inputs into the PII
- Serial processing of all inputs
- Write events to the PIO and to the flag and data regions
- Write PIO to the output modules
- Communication with other systems and programming devices

Measured values are read and processed in OB1. To do this, first the TIM-PN input quantities are assigned to the global variables in the standard variables table.

In the standard configuration of the device overview, the "Module" column is integrated as "Torque modul_1". It has the following measured values as input quantities: "Torque value LP1", "Torque value LP2", "Live counter value" and "Rotor temperature value". The last of these is only a "placeholder" and is not present in the hardware. For measured values: "Speed value LP1", "Speed value LP2", "Angle value" and "Power value", it is necessary to integrate Speed-Modul_1" from the hardware catalog. As a precondition, the rotor and stator must support this functionality.

6.2 Source code in SCL

Main [OB1]

```
//NETWORK 1: Torque value LP1
          "Torque_value_LP1_peripherie" //load
  T       "Torque_value_LP1"                    //transfer
  CAD                                           //switch accumulators
  DTR                                           //converts a
                                                  32-bit integer

                                                //into a floating-
                                                  point number
  L       1000.0                                //scale
  /R                                            //division
  T       "Torque_value_LP1_changed"            //transfer

//NETWORK 2: Torque value LP2
  L       "Torque_value_LP2_peripherie" //load
  T       "Torque_value_LP2"                    //transfer
  CAD                                           //switch accumulators
  DTR                                           //converts a
                                                  32-bit integer

                                                //into a floating-
                                                  point number
  L       1000.0                                //scale
  /R                                            //division
  T       "Torque_value_LP2_changed"            //transfer

//NETWORK 3: Speed value LP1
  L       "Speed_value_LP1_peripherie"
  T       "Speed_value_LP1"
  CAD
  DTR
  L       100.0
  /R
  T       "Speed_value_LP1_changed"

//NETWORK 4: Speed value LP2
  L       "Speed_value_LP2_peripherie"
  T       "Speed_value_LP2"
  CAD
  DTR
  L       100.0
  /R
  T       "Speed_value_LP2_changed"

UNTESTED

//NETWORK 5: Angle value
  L       "Angle_value_peripherie"
```

```
 T       "Angle_value"
 CAD
 DTR
 T       "Angle_value_changed"

//NETWORK 6: Power value
 L       "Power_value_peripherie"
 T       "Power_value"
 CAD
 DTR
 T       "Power_value_changed"

//NETWORK 7: Life counter
 L       "Life_countervalue_peripherie"
 T       "Life_counter_value"
 CAD
 DTR
 T       "Life_counter_value_changed"
         CAD;
 DTR;
 T       "Life_counter_value_changed";
```

7 Reading and setting a parameter set

7.1 Program sequence

The section of the program for reading and writing the parameter set is in the time-controlled organization module "Cyclic interrupt (OB30)". This is recommended because SFB52 and SFB53 work asynchronously. The data for reading and writing in this example is saved in data modules DB3 (write) and DB4 (read). The diagram below (*Fig. 7.1*) illustrates the process.

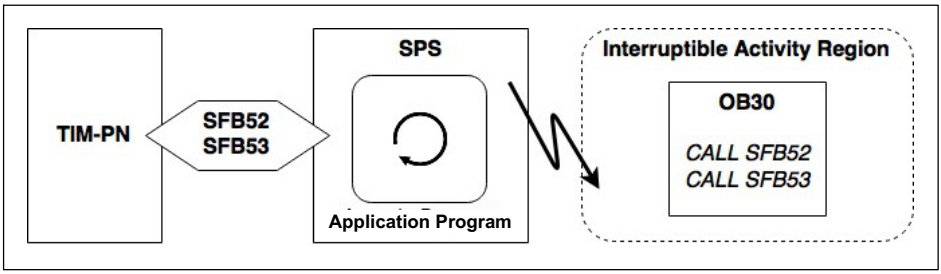

Fig. 7.1 Program sequence when reading and writing the parameter set

7.2 Source code in SCL

Created with Siemens STEP7 and TIA-Portal version 13 SP2.

DB_Cyclic_read_parameter [DB4]		
Name	**Data type**	**Start value**
Static		
Start_read_param	Bool	
Busy	Bool	
Error	Bool	
Status	Dword	
Index	Dint	
parameter_read	Byte	16#0
len	Uint	
valid	Bool	

DB_Cyclic_write_parameter [DB3]		
Name	**Data type**	**Start value**
Static		
Parameter_byte_set	Byte	16#0
Start_write_param	Bool	
Busy	Bool	
Done	Bool	
Error	Bool	
Status	DWord	
Index	DInt	

Cyclic interrupt [OB30]

```
// Parameter set read
"RDREC_DB"(REQ := "DB_Cyclic_read_parameter".Start_read_param,
     ID := "tim-pn~Torque_module_1~Select_parameter_set",
     INDEX := 1,
     MLEN := 1,
     VALID => "DB_Cyclic_read_parameter".valid,
     BUSY => "DB_Cyclic_read_parameter".Busy,
     ERROR => "DB_Cyclic_read_parameter".Error,
     STATUS => "DB_Cyclic_read_parameter".Status,
```

```
    LEN => "DB_Cyclic_read_parameter".len,
    RECORD := "DB_Cyclic_read_parameter".parameter_read);

// Parameter set write
"WRREC_DB"(REQ := "DB_Cyclic_write_parameter".Start_write_param,
    ID := "tim-pn~Torque_module_1~Select_parameter_set",
    INDEX := 1,
    LEN := 1,
    DONE => "DB_Cyclic_write_parameter".Done,
    BUSY => "DB_Cyclic_write_parameter".Busy,
    ERROR => "DB_Cyclic_write_parameter".Error,
    STATUS => "DB_Cyclic_write_parameter".Status,
    RECORD := "DB_Cyclic_write_parameter".Parameter_byte_set);
```

8 Status byte read

8.1 Program sequence

The general program sequence of OB1 is described in *section 6.1*.

The status byte is transferred cyclically within OB1 of function "TIM-PN Status Byte R [FC2]". The bits returned to the output of the function block as Boolean variables are analyzed in this function.

8.2 Description of the status bit

The status byte is listed in the "Submodule" device overview in the "Module" column: "Torque modul_1" "Status byte value". The assignment as variable is made in the standard variable table.

Name in GSDML file	Data type	Description
Status byte value	Unsigned8	Bit 0
		0 No action
		1 Zero setting torque active
		Bit 1
		0 No action
		1 Zero setting angle of rotation active
		Bit 2
		0 Shunt is off
		1 Shunt is on
		Bit 3 … 3
		reserved

Tab. 8.1 Description of the status bit

8.3 Source code in SCL and AWL

Created with Siemens STEP7 and TIA-Portal version 13 SP2.

TIM-PN status byte R [FC2]		
Name	**Data type**	**Comments**
Input		
Status_Byte	Byte	
Output		
Zero_setting_torque_active	Bool	
Zero_setting_angle_of_rotation_active	Bool	
Request_shunt_on	Bool	
Shunt_on	Bool	
InOut		
Temp		
Constant		
Return		
TIM-PN Status Byte R	Void	

```
#Zero_setting_torque_active := BYTE_TO_BOOL(#Status_Byte AND
B#2#0000_0001);

#Zero_setting_angle_of_rotation_active := BYTE_TO_BOOL(SHR(IN :=
(#Status_Byte AND B#2#0000_0010), N := 1));

#Shunt_on := BYTE_TO_BOOL(SHR(IN := (#Status_Byte AND B#2#0000_0100), N
:= 2));

#Request_shunt_on := BYTE_TO_BOOL(SHR(IN := (#Status_Byte AND
B#2#0001_0000), N := 4));
```

Main [OB1]

```
//Network 9: Status-Byte
  L       "Status_byte_value_peripherie"
  T       "Status_byte_value_changed"

  CALL    "TIM-PN Status Byte R"
    Status_Byte             :="Status_byte_value_peripherie"
    Zero_setting_torque_
    active                  :="Status_Zero_Setting_Torque_active"
    Zero_setting_angle_of_rotation_
    active                  :="Status_Zero_Setting_AoR_active"    Request
    _shunt_on               :="Status_Request_Shunt_On"
    Shunt_on                :="Status_Shunt_On"
```

9 Control byte write

9.1 Program sequence

The general program sequence of OB1 is described in *section 3.1*.

The control byte is set cyclically within OB1 of function "TIM-PN Control Byte W [FC1]". In this function the control bits of the output byte are manipulated depending on the Boolean input variables of the function block.

9.2 Description of the control bit

The control byte is listed in the "Submodule" device overview in the "Module" column: "Torque modul_1" "Control byte value". The assignment as variable is made in the standard variable table.

Name in GSDML file	Data type	Description
Status byte value	Unsigned8	Bit 0
		0 No action
		1 Request torque zero setting
		Bit 1
		0 No action
		1 Request angle of rotation zero setting
		Bit 2
		0 Request shunt off
		1 Request shunt on

Tab. 9.1 Description of the control bit

9.3 Source code in SCL and AWL

TIM-PN Control Byte W [FC1]		
Name	**Data type**	**Comments**
Input		
Set_Shunt_On	Bool	
Set_Torque_Zero	Bool	
Set_Angle_Zero	Bool	
Output		
Control_Byte	Byte	
InOut		
Temp		
Constant		
Return		
TIM-PN Control Byte W	Void	

```
#Control_Byte := 0;

IF #Set_Shunt_On = TRUE
THEN
  #Control_Byte := #Control_Byte OR B#16#04;
END_IF;

IF #Set_Angle_Zero = TRUE
THEN
  #Control_Byte := #Control_Byte OR B#16#02;
END_IF;

IF #Set_Torque_Zero = TRUE
THEN
  #Control_Byte := #Control_Byte OR B#16#01;
END_IF;
```

Main [OB1]

```
//Network 10: Control Byte
CALL     "TIM-PN Control Byte W"
  Set_Shunt_On           :="Set_Shunt_ON"
  Set_Torque_Zero        :="Set_Zero_Torque"
  Set_Angle_Zero         :="Set_Zero_Angle"
  Control_Byte           :="Control_byte_value"
```

10 PROFINET Diagnosis using the S7 User program

Detailed diagnosis by reading data sets from the TIM-PN devices.

- OB82 (diagnostic alarm) with SFB54 "RALARM" (Receive Alarm): Organization module 82 is called for asynchronous error related to the diagnostic alarm. Errors that cannot be assigned to the program sequence based on the time when they occur are referred to as "asynchronous errors". (cf. [1], section 5.9, page 238f)

  System block 54 also reads alarm information from the triggering components and modules. It is called within an alarm organization module, for example OB82. RALARM is processed synchronously. This means that the requested data (TINFO and AINFO) will be available immediately after the call in the output parameters (cf. [1], section 5.7.7, page 222ff). RALARM is not required for the TIM-PN diagnostic data sets – it is simply an addition here.

  The flag to start RDREC is set here.

- Wake-up alarm (for S7-1500 ⍰ 20 OBs) OB30 to OB38 and others beginning at OB123: Event class

  "Cyclic Interrupt" is an event that is triggered at periodic (configurable) intervals. The program is processed in the wake-up alarm OB independently of processing of the cyclic program (e.g. OB1). (see [1], section 5.7, page 215ff)

  The asynchronously running function block RDREC is executed here. This is recommended because execution in the main program, alarm program or error program can lead to delays in the cycle processing time. (cf. [1], section 5.5, page 184)

Information

The processing time of a wake-up alarm must be considerably less than its time grid. Otherwise OB80 (time error) is called. If OB80 is not present, the error is ignored.

10.1 Addressing the TIM-PN diagnostic data set (PROFINET-IO)

System block "RDREC" is used to read a data set with the number INDEX from the module and save it in the RECORD data region. The INDEX for calling the diagnostic data of TIM-PN is defined below:

0xF80C: 0xF00C:	Returns all diagnostic data of the device Returns all diagnostic data for an API (Application Process Identifier)
0xE00C: 0xC00C:	Returns all diagnostic data for an AR (Application Relation) Returns all diagnostic data for a Slot
0x800C:	Returns all diagnostic data for a SubSlot.

10.2 Structure of diagnostic data sets

The (*Fig. 10.1*) shows the dataset structure used for TIM-PN diagnostic messages. Channel-dependent diagnostic data is transferred in the USI (bytes 19 and 20) in format I (W#16#8000). This corresponds to the structure that is used (*see Fig. 10.2: Source code of the diagnostic structure*).

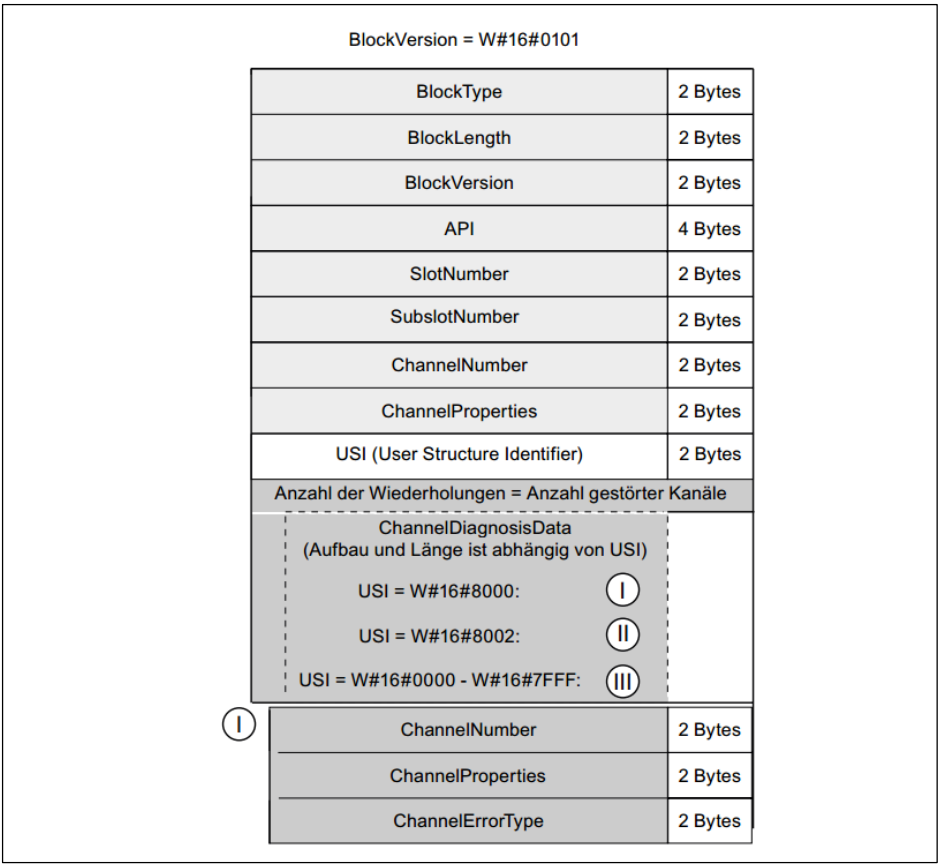

Fig. 10.1 Structure of the diagnosis record

```
/** \brief This structure describes the structure of the PROFINET Channel Diagnosis data.
           For a detailed description of the structure parameters see the specification IEC61158 Part5/6
struct SDAI_PN_PROFINET_CHANNEL_DIAGNOSIS
{
  U16   UserStructureIdent;
  U16   ChannelNumber;
  U16   ChannelProperties;   /**< This parameter is handled completely by the stack */
  U16   ChannelError;
};
```

Fig. 10.2 Source code of the diagnostic structure

10.3 Program sequence

Control begins after the initialization step with cyclical processing of the program – in this case "Main (OB1)". The diagnostic alarm (OB82) is called acyclically, for example if a diagnostic message is rejected with TIM-PN. It is indicated internally in OB82 that "enable read" (REQ = 1) is set for reading diagnostic information of TIM-PN. SFB52, which runs asynchronously, is called in the time-controlled organization module "Cyclic interrupt (OB30)". After the diagnosis record has been read successfully, the diagnostic messages are extracted and in this example saved in a ring buffer. The diagram below (*Fig. 10.3*) illustrates the program sequence:

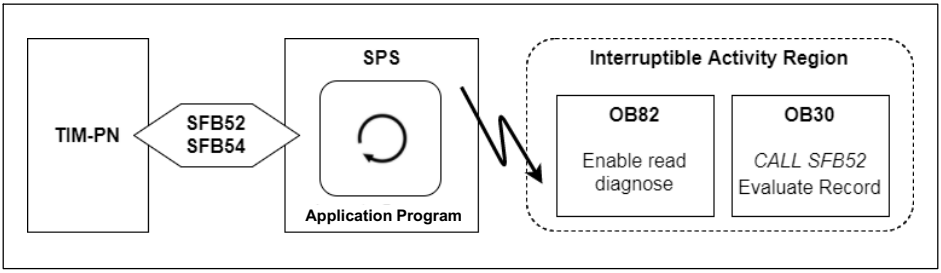

Fig. 10.3 Program sequence for reading diagnostic data

The exact meaning of each byte and more extensive diagnostic possibilities for the records (data sets in PROFINET IO), as well as the "ChannelDiagnosisData" (ChannelNumber, ChannelProperties and ChannelErrorTypes) are described in detail in the corresponding Siemens manual. (cf. [4], section 5.5 Blocks of diagnosis and configuration data sets). Further examples and explanations can be found there.

The following section (*see section 10.4*) describes only the manufacturer-specific diagnostic messages of TIM-PN.

10.4 Description of the error codes for the Channel Error Types

Errorrcode	Mapping to module	Channel-error type	Error text
TIM_Parameter_Error	Head module	16	Parameter Error Diagnosis: This is a TIM Parameter Error
Measurement Failure	Head module	256	Measurement Failure Diagnosis: This is a Measurement Failure
Torque_Channei1_Failure	Torque module	257	Torque Channel Failure 1 Diagnosis: This is a Torque ChannelFailure 1
Torque_Channei2_Failure	Torque module	258	Torque Failure Channel 2 Diagnosis: This is a Torque ChannelFailure 2
Speed_Channei1_Failure	Speed module	259	Speed Channel Failure 1 Diagnosis: This is a Speed ChannelFailure 1
Speed_Channei2_Failure	Speed module	260	Speed Channel Failure 2 Diagnosis: This is a Speed ChannelFailure 2
Angle_Channei_Failure	Speed module	261	Angle Channel Failure Diagnosis: This is an Angle ChannelFailure
Power_Channel_Failure	Speed module	262	Power Channel Failure Diagnosis: This is a Power ChannelFailure
Rotor_Temperature_Failure	Torque module	263	Temperature Channel Failure Diagnosis: This is a Temperature Channel Failure

Errorrcode	Mapping to module	Channel-error type	Error text
Stator_Error	Head module	264	Stator Error Diagnosis: This is a Stator Error
Rotor Error	Head module	265	Rotor Error Diagnosis: This is a Rotor Error
TIM-Failure	Head module	266	TIM Failure Diagnosis: This is a TIM Failure
Parameter_Selection_Failure	Head module	267	Parameter Selection Failure Diagnosis: This is a Parameter Selection Failure

Tab. 10.1 Error code diagnosis mapping (cf. [2], page 86f)

10.5 Source code in SCL

Created with Siemens STEP7 and TIA-Portal version 13 SP2.

DB_Cyclic_read_diagnostic [DB5]		
Name	**Data type**	**Start value**
Static		
Start_read_diag	Bool	
Busy	Bool	
Error	Bool	
Status	Dword	
Index	Dint	
len	Uint	
valid	Bool	
parameter_read	Array[1..128] of Word	
parameter_read[1]	Word	16#0
…	…	…
parameter_read[128]	Word Array[0..15] of	16#0
ChannelDiagnoseArray	Struct	
ChannelDiagnoseArray[0]	Struct	
ChannelNumber	Word	16#0
ChannelProperties	Word	16#0
ChannelErrorType	Word	16#0
…	…	
ChannelDiagnoseArray[15]	Struct	
ChannelNumber	Word	16#0
ChannelProperties	Word	16#0
ChannelErrorType	Word	16#0
IndexCDA	Int	0

Diagnostic error interrupt [OB82]

```
//For additional diagnoses information call RALARM...

//Set flag for start reading diagnose record
"DB_Cyclic_read_diagnostic".Start_read_diag := true;
```

Cyclic interrupt [OB30]		
Name	**Data type**	**Comments**
Input		
Initial_Call	Bool	Initial call of this OB
Event_Count	Int	Events discarded
Temp	Int	loop counter variable
i usedLength	Int	while loop break condition
blockLength	Int	length of the block
blockVersion	Word	version of the block
usi	Word	user structure identifier
blockLengthIndex	Int	index of the next blockLength entry
blockVersionIndex	Int	index of the next blockVersion entry
usiIndex	Int	index of the next USI entry
Constant		
positionBlockLength = 2	Int	Offset for BlockLength index at start of the block
positionBlockVersion = 3	Int	Offset for BlockVersion index at start of the block
positionUSI = 10	Int	Offset for USI index at start of the block
upperLimitCBA = 15	Int	upper limit of the circular buffer index

```
// Read Diagnose
"RDREC_DB_diag"(REQ:="DB_Cyclic_read_diagnostic".Start_read_diag,
    ID:="tim-pn~Head",
    INDEX:=W#16#F80C,
    MLEN:=256,
    VALID=>"DB_Cyclic_read_diagnostic".valid,
    BUSY=>"DB_Cyclic_read_diagnostic".Busy,
    ERROR=>"DB_Cyclic_read_diagnostic".Error,
    STATUS=>"DB_Cyclic_read_diagnostic".Status,
    LEN=>"DB_Cyclic_read_diagnostic".len,
    RECORD:= "DB_Cyclic_read_diagnostic".parameter_read);

//evaluate received channel diagnose Record
IF "DB_Cyclic_read_diagnostic".valid = TRUE
THEN
    //set start values
    #usedLength := 0;
    #blockLengthIndex := #positionBlockLength;
    #blockVersionIndex := #positionBlockVersion;
```

```
    #usiIndex := #positionUSI;

    //evaluate as long as blocks available or exit if there is a wrong
    block version
    WHILE (#usedLength < UINT_TO_INT("DB_Cyclic_read_diagnostic".len))
    DO
       #blockLength :=
WORD_TO_INT("DB_Cyclic_read_diagnostic".parameter_read[#blockLengthIndex
]);      //Byte 3 and 4 => Block length
       #blockVersion :=
"DB_Cyclic_read_diagnostic".parameter_read[#blockVersionIndex];      //B
yte 5 and 6 => Block version

       IF #blockVersion = W#16#0101    //check version for data mapping
    THEN
         #usedLength := #usedLength + (4 + #blockLength);      //16+4
(header) are fix because of version
         //32     :=     0      +      (4 + 28)       //will
be the offset for the second block
         //58     :=     32     +      (4 + 22)

         #usi := "DB_Cyclic_read_diagnostic".parameter_read[#usiIndex];
         //read
USI
         IF #usi = W#16#8000     //check

ChannelDiagnoseData format
         THEN
           //start read diagnose 6 diagnose bytes per channel
           FOR #i := (#usiIndex + 1) TO (#usedLength / 2) BY 3   //divi-
           de
by two because of word array
           DO
             //check upper limit of circular buffer index
             IF "DB_Cyclic_read_diagnostic".IndexCDA > #upperLimitCBA
             THEN
                "DB_Cyclic_read_diagnostic".IndexCDA := 0;
             END_IF;

               //ChannelNumber
"DB_Cyclic_read_diagnostic".ChannelDiagnoseArray["DB_Cyclic_read_diagnos-
tic".IndexCDA]. ChannelNumber :=
"DB_Cyclic_read_diagnostic".parameter_read[#i];

             //ChannelProperties
"DB_Cyclic_read_diagnostic".ChannelDiagnoseArray["DB_Cyclic_read_diagnos-
tic".IndexCDA]. ChannelProperties :=
"DB_Cyclic_read_diagnostic".parameter_read[#i + 1];

             //ChannelErrorType
```

```
"DB_Cyclic_read_diagnostic".ChannelDiagnoseArray["DB_Cyclic_read_diagnos-
tic".IndexCDA]. ChannelErrorType :=
"DB_Cyclic_read_diagnostic".parameter_read[#i + 2];

              //increment circular buffer index
              "DB_Cyclic_read_diagnostic".IndexCDA :=
"DB_Cyclic_read_diagnostic".IndexCDA + 1;
           END_FOR;
         END_IF;
      ELSE

         //in case of wrong version, we have to exit the loop because we
         cannot calculate the index of the next block
         EXIT;
      END_IF;
      //calculate index of the next block
      #blockLengthIndex := #positionBlockLength + (#usedLength / 2);
      #blockVersionIndex := #positionBlockVersion + (#usedLength / 2);
      #usiIndex := #positionUSI + (#usedLength / 2);
   END_WHILE;
   //reset start flag for acyclic read of RDREC (will be set in OB82)
   "DB_Cyclic_read_diagnostic".Start_read_param := false;
END_IF;
```

Programming Instructions |
Programmieranleitung

English | Deutsch

TIM-PN

mit Siemens S7-1500 Steuerung

HBM

1 Referenzdokumente

Ref.	Titel	Vers.	Pfad
[1]	Hans Berger, Siemens (Hg.): Automatisieren mit SIMATIC S7-1500	2014	ISBN 978-3-89578-403-3
[2]	Montageanleitung_TIM-PN.pdf	n.a.	https://www.hbm.com/fileadmin/mediapool/hbmdoc/technical/a4341.pdf (14.09.2017)
[3]	profinet_step7_v14_function_manual_de-DE_de-DE.pdf	2014	https://www.siemens.de/Digital- Factory/download/EventDocs/profinet_step7_v14_function_manual_de-DE_de-DE.pdf (15.09.2017)
[4]	Von_PROFIBUS_DP_nach_PROFINET_IO_de-DE.pdf	2010	https://support.industry.siemens.com/cs/attachments/19289930/Von_PROFIBUS_DP_nach_PROFINET_IO_de-DE.pdf?download=true (20.09.17)

2 Abkürzungsverzeichnis

API	Application Process Identifier
AR	Application Relation
CPU	Central Processing Unit
DB	Datenbaustein
DP	Dezentrale Peripherie
EVA	Einlesen Verarbeiten Ausgeben
IE	Industrial Ethernet
OB	Organisationsbaustein
PAA	Peripherie Abbild der Ausgänge
PAE	Peripherie Abbild der Eingänge
PG	Programmiergerät
PLC	Programmable Logic Controller (dt. SPS)
PN	Profinet
SCL	Structured Control Language
SFB	Systembaustein
SPS	Speicherprogrammierbare Steuerung
USI	User Structure Identifier

3 Beschreibung

Dieses Dokument beschreibt das Vorgehen zum Auslesen der TIM-PN spezifischen PROFINET IO Diagnose-Informationen, das zyklische auslesen der Mess- und Steuerwerte, das Auslesen und Setzen des Parametersets und das zyklische modifizieren des Steuerbytes via Siemens SPS. Zur Demonstration wird eine S7-1500 verwendet.

Abgrenzung: Dieses Anwendungsbeispiel beschreibt nicht die Systemdiagnose und Fehlerereignisse der Steuerung. Für weitere Diagnosen im Anwenderprogramm gibt es Beispiele von Siemens mit weiteren Links bezüglich dieser Thematik:

https://support.industry.siemens.com/cs/document/98210758/diagnose-im-anwenderprogramm-mit-s7-1500?dti=0&lc=de-WW (14.09.2017)

https://support.industry.siemens.com/cs/document/24000238/profinet-io-%E2%80%93- diagnoseverarbeitung-im-anwenderprogramm?dti=0&lc=de-ww (22.09.2017)

4 Aufbau

4.1 Verwendete Komponenten

Komponente	Artikelnummer	Hinweis
CPU 1516-3 PN/DP	6ES7 516-3AN01-0AB0	Alternativ kann auch jede andere S7-1500 CPU verwendet werden
TIM-PN		

4.2 Hardwareaufbau

Die Steuerung und das TIM-PN werden direkt miteinander als Profinet-Subnetz verschaltet (eine Möglichkeit zeigt *Abb. 4.2.* Das PG/PC ist an dem freien Port der SPS angeschlossen. Der Webservice des TIM-PN wird ebenfalls an den PC über eine zweite Netzwerkkarte angeschlossen. Falls dies nicht möglich ist, kann auch ein Switch verwendet werden – hierbei ist darauf zu achten, dass der Webserver und das Profinet des TIM-PN nicht über einen Switch verschaltete sind (*siehe Abb. 4.1*).

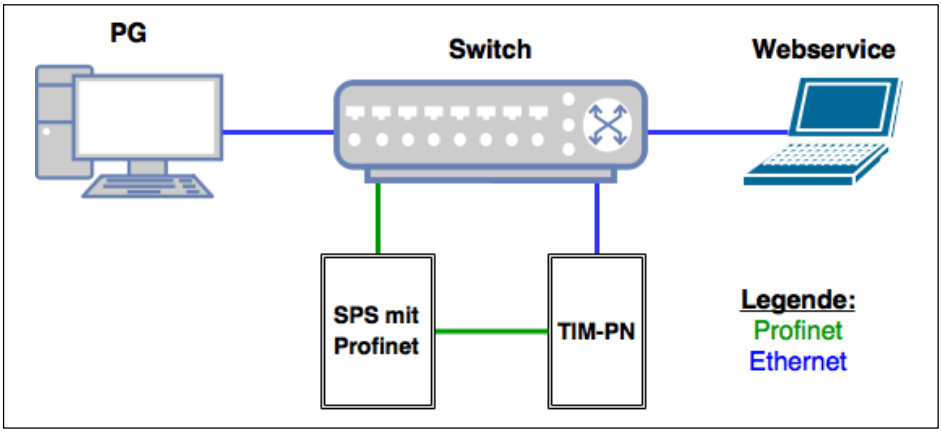

Abb. 4.1 Verschaltungsvariante mit Switch

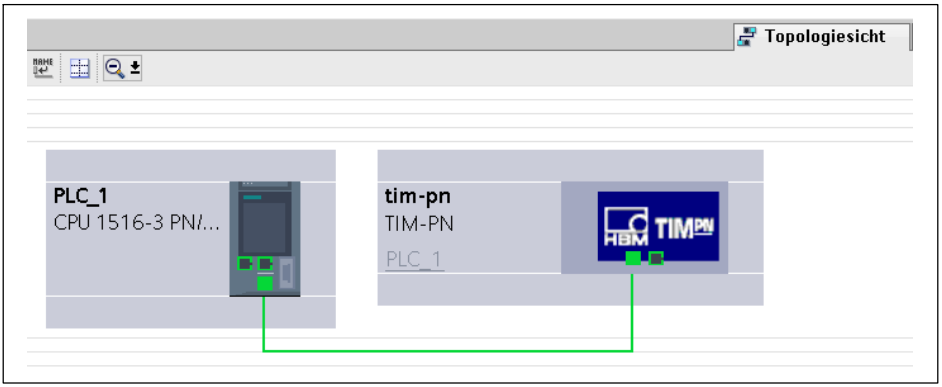

Abb. 4.2 Profinet-Topologie mit S7-1500 und TIM-PN

4.3 Adressierung

Komponente	IP-Adresse	Netzmaske
PLC_1: X1	PN: 192.168.1.23	255.255.255.0
tim-pn: X6	PN: 192.168.1.13	255.255.255.0
PLC_1: X2	IE: 192.168.0.1	255.255.255.0
PG	IE: 192.168.0.241	255.255.255.0
tim-pn: X5	IE: 192.168.1.2	255.255.255.0
PC	IE: 192.168.1.254	255.255.255.0

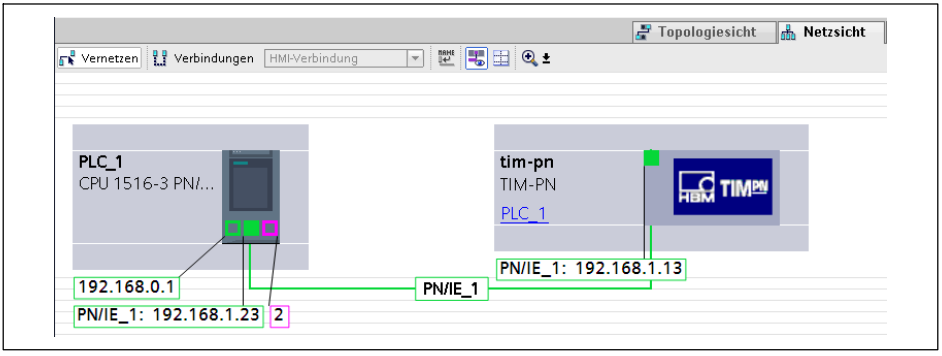

Abb. 4.3 Vernetzung mit IP-Adressen

4.4 Geräteübersicht

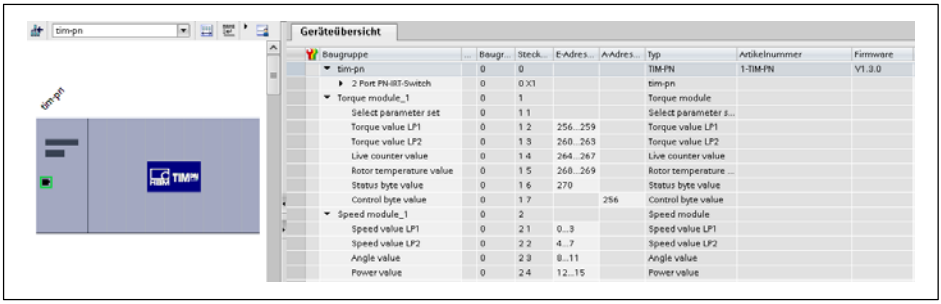

Abb. 4.4 Geräteübersicht des TIM-PN

4.5 Variablentabelle

Standardvariablentabelle		
Angle_value	DInt	%MD18
Angle_value_changed	Real	%MD118
Angle_value_peripherie	DInt	%ID8
Control_byte_value	Byte	%QB256
Control_byte_value_eingeben	Byte	%MB256
Life_counter_value	DInt	%MD26
Life_counter_value_changed	DInt	%MD126
Life_countervalue_peripherie	DInt	%ID264
Power_value	DInt	%MD22
Power_value_changed	Real	%MD122
Power_value_peripherie	DInt	%ID12
Rotor_temperature_value	Word	%MW30
Rotor_temperature_value_changed	Real	%MD130
Rotor_temperature_value_peripherie	Word	%IW269
Set_Shunt_ON	Bool	%M256.3
Set_Zero_Angle	Bool	%M256.2
Set_Zero_Torque	Bool	%M256.1
Speed_value_LP1	DInt	%MD10

Standardvariablentabelle		
Speed_value_LP1_changed	Real	%MD110
Speed_value_LP1_peripherie	DInt	%ID0
Speed_value_LP2	DInt	%MD14
Speed_value_LP2_changed	Real	%MD114
Speed_value_LP2_peripherie	DInt	%ID4
Status_byte_value_peripherie	Byte	%IB270
Status_Request_Shunt_On	Bool	%M16.3
Status_Shunt_On	Bool	%M16.2
Status_Zero_Setting_AoR_active	Bool	%M16.1
Status_Zero_Setting_Torque_active	Bool	%M16.0
Torque_value_LP1	DInt	%MD1
Torque_value_LP1_changed	Real	%MD102
Torque_value_LP1_peripherie	DInt	%ID256
Torque_value_LP2	DInt	%MD6
Torque_value_LP2_changed	Real	%MD106
Torque_value_LP2_peripherie	DInt	%ID260

5 Benötigte Funktionsbausteine

„RDREC“ und „WRREC“ sind asynchron arbeitende Anweisungen, d. h. die Bearbeitung erstreckt sich über mehrere Aufrufe. Sie starten die Datensatzübertragung, indem REQ = 1 gesetzt wird. Über den Ausgangsparameter BUSY und die mittleren zwei Bytes des Ausgangsparameters STATUS wird der Zustand des Auftrags angezeigt. (Vgl. Siemens Hilfe/Informationssystem)

Information

Die Schnittstellen der Anweisungen sind identisch zu der Norm "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3"

Für die Migration von S7-300/400-Programmen gibt es noch folgende Bausteine:

- RD_REC
- WR_REC

5.1 SFB52 „RDREC“ (Read Record)

Mit dieser Anweisung lesen Sie den Datensatz mit der Nummer INDEX von der mittels ID adressierten Komponente (Hardware-Kennung). Mit MLEN geben Sie vor, wie viele Bytes Sie maximal lesen möchten. Den Zielbereich RECORD sollten Sie daher mindestens MLEN Bytes lang wählen. Der Wert TRUE des Ausgangsparameters VALID zeigt an, dass der Datensatz erfolgreich in den Zielbereich RECORD übertragen wurde. In diesem Fall enthält der Ausgangsparameter LEN die Länge der gelesenen Daten in Bytes. Falls bei der Datensatzübertragung ein Fehler auftrat, wird dies über den Ausgangsparameter ERROR angezeigt. Der Ausgangsparameter STATUS enthält in diesem Fall die Fehlerinformation.

Falls ein Fehler auftritt und im STATUS das zweite Byte ein C0 (read constrain conflict) beinhaltet, dann könnte die angeforderte Länge im Read-Request zu klein sein. MLEN sollte größer gewählt werden. Die Länge im Read Request kann normalerweise auch viel größer als die tatsächliche Länge der Diagnose-

daten sein. Es wird dann nur so viel geliefert wie auch tatsächlich vorhanden ist.

Information

Wenn ein DPV1-Slave über GSD-Datei projektiert ist (GSD ab Rev. 3) und die DP-Schnittstelle des DP-Masters auf "S7-kompatibel" eingestellt ist, dürfen im Anwenderprogramm keine Datensätze mit "RDREC" von den E/A-Baugruppen gelesen werden. Der DP-Master adressiert in diesem Fall den falschen Steckplatz (projektierter Steckplatz + 3).
Abhilfe: Schnittstelle des DP-Masters auf "DPV1" umstellen.

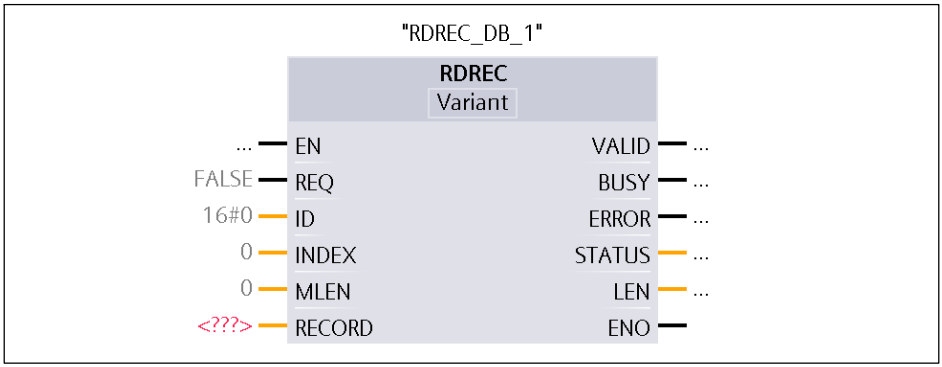

Abb. 5.1 Siemens Funktionsbaustein ->SFB52

5.2 SFB53 „WRREC“ (Write Record)

Mit der Anweisung übertragen Sie den Datensatz RECORD zu der mittels ID adressierten Komponente (Hardware-Kennung). Mit LEN geben Sie die Länge des zu übertragenden Datensatzes in Bytes vor. Den Quellbereich RECORD sollten Sie daher mindestens LEN Bytes lang wählen. Der Wert TRUE des Ausgangsparameters DONE zeigt an, dass der Datensatz erfolgreich übertragen wurde. Falls bei der Datensatzübertragung ein Fehler auftrat, wird dies über den Ausgangsparameter ERROR angezeigt. Der Ausgangsparameter STATUS enthält in diesem Fall die Fehlerinformation.

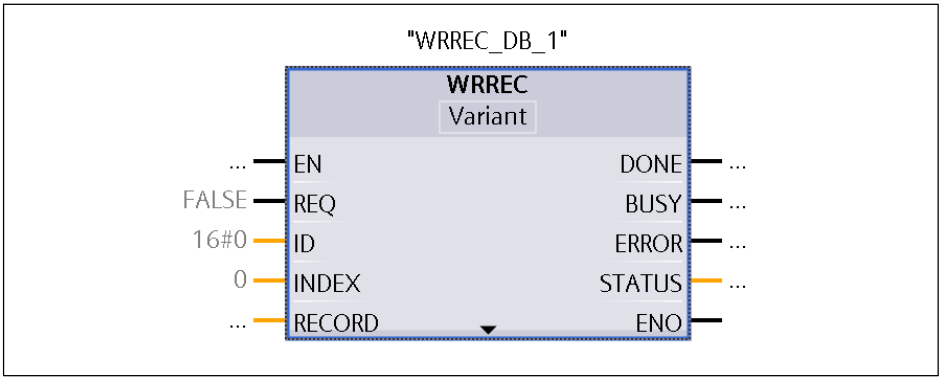

Abb. 5.2 Siemens Funktionsbaustein ->SFB53

6 Auslesen und bearbeiten von Messwerten

6.1 Programmablauf

Im OB1 erfolgt die Programmbearbeitung zyklisch nach dem EVA-Prinzip. Dies setzt voraus, dass die SPS

erfolgreich angelaufen ist und sich die CPU im RUN befindet. Ein Zyklus lässt sich in fünf Punkte gliedern:

- Start der Zykluszeitüberwachung
- Einlesen aller Eingänge in das PAE
- Serielle Bearbeitung aller Eingänge
- Ergebnisse in das PAA und in die Merker- und Datenbereiche schreiben
- Schreiben des PAA in die Ausgangsbaugruppen
- Kommunikation mit anderen Systemen und Programmiergeräten

Das Auslesen und bearbeiten von den Messwerten erfolgt im OB1. Hierzu werden zunächst in der Standard-Variablentabelle die Eingangsgrößen des TIM-PN den globalen Variablen zugeordnet.

In der Geräteübersicht, Spalte „Baugruppe“ ist standardmäßig das „Torque modul_1“ eingebunden. Dieses besitzt die Messwerte als Eingangsgrößen: „Torque value LP1“, „Torque value LP2“, „Live counter value“ und „Rotor temperature value“. Letzteres ist nur ein „Platzhalter“ und ist nicht in Hardware vorhanden. Für die Messwerte: „Speed value LP1“, „Speed value LP2“, „Angle value“ und „Power value” muss das „Speed-Modul_1” aus dem Hardware-Katalog eingebunden werden. Vorausgesetzt, dass der Rotor und der Stator diese Funktionalität unterstützen.

6.2 Quellcode in SCL

Main [OB1]

```
//NETWORK 1: Torque value LP1
          "Torque_value_LP1_peripherie" //load
  T       "Torque_value_LP1"                    //transfer
  CAD                                           //switch accumulators
  DTR                                           //converts a
                                                  32-bit integer

                                                //into a floating-
                                                  point number
  L       1000.0                                //scale
  /R                                            //division
  T       "Torque_value_LP1_changed"            //transfer

//NETWORK 2: Torque value LP2
  L       "Torque_value_LP2_peripherie" //load
  T       "Torque_value_LP2"                    //transfer
  CAD                                           //switch accumulators
  DTR                                           //converts a
                                                  32-bit integer

                                                //into a floating-
                                                  point number
  L       1000.0                                //scale
  /R                                            //division
  T       "Torque_value_LP2_changed"            //transfer

//NETWORK 3: Speed value LP1
  L       "Speed_value_LP1_peripherie"
  T       "Speed_value_LP1"
  CAD
  DTR
  L       100.0
  /R
  T       "Speed_value_LP1_changed"

//NETWORK 4: Speed value LP2
  L       "Speed_value_LP2_peripherie"
  T       "Speed_value_LP2"
  CAD
  DTR
  L       100.0
  /R
  T       "Speed_value_LP2_changed"
```

UNTESTED

```
//NETWORK 5: Angle value
  L       "Angle_value_peripherie"
```

```
  T       "Angle_value"
  CAD
  DTR
  T       "Angle_value_changed"

//NETWORK 6: Power value
  L       "Power_value_peripherie"
  T       "Power_value"
  CAD
  DTR
  T       "Power_value_changed"

//NETWORK 7: Lifecounter
  L       "Life_countervalue_peripherie"
  T       "Life_counter_value"
  CAD
  DTR
  T       "Life_counter_value_changed"
          CAD;
  DTR;
  T       "Life_counter_value_changed";
```

7 Parameterset lesen und setzen

7.1 Programmablauf

Der Programmteil zum Lesen und Schreiben des Parametersatz erfolgt in dem zeitgesteuerten Organisationsbaustein „Cyclic interrupt (OB30)“. Dies ist Aufgrund des asynchron arbeitenden SFB52 und SFB53 empfehlenswert. Die Daten für das Lesen und Schreiben werden in diesem Beispiel in den Datenbausteinen, DB3 (schreiben) und DB4 (lesen), gespeichert. Die nachfolgende Abbildung (*Abb. 7.1*) veranschaulicht den Ablauf.

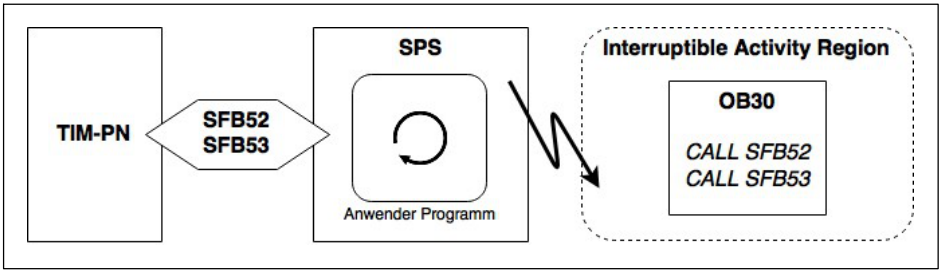

Abb. 7.1 Programmablauf beim Lesen und Schreiben des Parametersatz

7.2 Quellcode in SCL

Erstellt mit Siemens STEP7 und TIA-Portal Version 13 SP2.

DB_Cyclic_read_parameter [DB4]		
Name	**Datentyp**	**Startwert**
Static		
Start_read_param	Bool	
Busy	Bool	
Error	Bool	
Status	Dword	
Index	Dint	
parameter_read	Byte	16#0
len	Uint	
valid	Bool	

DB_Cyclic_write_parameter [DB3]		
Name	**Datentyp**	**Startwert**
Static		
Parameter_byte_set	Byte	16#0
Start_write_param	Bool	
Busy	Bool	
Done	Bool	
Error	Bool	
Status	DWord	
Index	DInt	

Cyclic interrupt [OB30]

```
// Parametersatz read
"RDREC_DB"(REQ := "DB_Cyclic_read_parameter".Start_read_param,
     ID := "tim-pn~Torque_module_1~Select_parameter_set",
     INDEX := 1,
     MLEN := 1,
     VALID => "DB_Cyclic_read_parameter".valid,
     BUSY => "DB_Cyclic_read_parameter".Busy,
     ERROR => "DB_Cyclic_read_parameter".Error,
     STATUS => "DB_Cyclic_read_parameter".Status,
```

```
    LEN => "DB_Cyclic_read_parameter".len,
    RECORD := "DB_Cyclic_read_parameter".parameter_read);
// Parametersatz write
"WRREC_DB"(REQ := "DB_Cyclic_write_parameter".Start_write_param,
    ID := "tim-pn~Torque_module_1~Select_parameter_set",
    INDEX := 1,
    LEN := 1,
    DONE => "DB_Cyclic_write_parameter".Done,
    BUSY => "DB_Cyclic_write_parameter".Busy,
    ERROR => "DB_Cyclic_write_parameter".Error,
    STATUS => "DB_Cyclic_write_parameter".Status,
    RECORD := "DB_Cyclic_write_parameter".Parameter_byte_set);
```

8 Statusbyte lesen

8.1 Programmablauf

Der allgemeine Programmablauf des OB1 ist in *Kapitel 6.1* beschrieben.

Das Statusbyte wird zyklisch innerhalb des OB1 der Funktion „TIM-PN Status Byte R [FC2]“ übergeben. In dieser Funktion erfolgt die Auswertung der Bits, welche als Boolesche-Variablen an den Ausgang des Funktionsbausteins zurückgegeben werden.

8.2 Beschreibung der Statusbits

Das Statusbyte ist in der Geräteübersicht „Submodul“ in der Spalte „Baugruppe“ gelistet: „Torque modul_1“ „Status byte value“. Die Zuordnung als Variable erfolgt in der Standard-Variablentabelle.

Name in GSDML Datei	Datatype	Description
Status byte value	Unsigned8	Bit 0
		0 No action
		1 Zero setting torque active
		Bit 1
		0 No action
		1 Zero setting angle of rotation active
		Bit 2
		0 Shunt is off
		1 Shunt is on
		Bit 3 … 3
		reserved

Tab. 8.1 Beschreibung der Statusbits

8.3 Quellcode in SCL und AWL

Erstellt mit Siemens STEP7 und TIA-Portal Version 13 SP2.

TIM-PN Status Byte R [FC2]		
Name	**Datentyp**	**Kommentar**
Input		
Status_Byte	Byte	
Output		
Zero_setting_torque_active	Bool	
Zero_setting_angle_of_rotation_active	Bool	
Request_shunt_on	Bool	
Shunt_on	Bool	
InOut		
Temp		
Constant		
Return		
TIM-PN Status Byte R	Void	

```
#Zero_setting_torque_active := BYTE_TO_BOOL(#Status_Byte AND
B#2#0000_0001);

#Zero_setting_angle_of_rotation_active := BYTE_TO_BOOL(SHR(IN :=
(#Status_Byte AND B#2#0000_0010), N := 1));

#Shunt_on := BYTE_TO_BOOL(SHR(IN := (#Status_Byte AND B#2#0000_0100), N
:= 2));

#Request_shunt_on := BYTE_TO_BOOL(SHR(IN := (#Status_Byte AND
B#2#0001_0000), N := 4));
```

Main [OB1]

```
//Netzwerk 9: Status-Byte
  L       "Status_byte_value_peripherie"
  T       "Status_byte_value_changed"

  CALL    "TIM-PN Status Byte R"
    Status_Byte             :="Status_byte_value_peripherie"
    Zero_setting_torque_
    active                  :="Status_Zero_Setting_Torque_active"
    Zero_setting_angle_of_rotation_
    active                  :="Status_Zero_Setting_AoR_active"    Request
    _shunt_on               :="Status_Request_Shunt_On"
    Shunt_on                :="Status_Shunt_On"
```

9 Steuerbyte schreiben

9.1 Programmablauf

Der allgemeine Programmablauf des OB1 ist in *Kapitel 3.1* beschrieben.

Das Steuerbyte wird zyklisch innerhalb des OB1 von der Funktion „TIM-PN Control Byte W [FC1]“ gesetzt. In dieser Funktion werden die Steuerbits des Ausgangsbytes manipuliert in Abhängigkeit von den Booleschen-Eingangsvariablen des Funktionsbausteins.

9.2 Beschreibung der Steuerbits

Das Steuerbyte ist in der Geräteübersicht „Submodul“ in der Spalte „Baugruppe“ gelistet: „Torque modul_1“ „Control byte value“. Die Zuordnung als Variable erfolgt in der Standard-Variablentabelle.

Name in GSDML Datei	Datatype	Description
Status byte value	Unsigned8	Bit 0
		0 No action
		1 Request torque zero setting
		Bit 1
		0 No action
		1 Request angle of rotation zero setting
		Bit 2
		0 Request shunt off
		1 Request shunt on

Tab. 9.1 Beschreibung der Steuerbits

9.3 Quellcode in SCL und AWL

TIM-PN Control Byte W [FC1]		
Name	**Datentyp**	**Kommentar**
Input		
Set_Shunt_On	Bool	
Set_Torque_Zero	Bool	
Set_Angle_Zero	Bool	
Output		
Control_Byte	Byte	
InOut		
Temp		
Constant		
Return		
TIM-PN Control Byte W	Void	

```
#Control_Byte := 0;

IF #Set_Shunt_On = TRUE
THEN
  #Control_Byte := #Control_Byte OR B#16#04;
END_IF;

IF #Set_Angle_Zero = TRUE
THEN
  #Control_Byte := #Control_Byte OR B#16#02;
END_IF;

IF #Set_Torque_Zero = TRUE
THEN
  #Control_Byte := #Control_Byte OR B#16#01;
END_IF;
```

Main [OB1]

```
//Netzwerk 10: Controle-Byte
CALL     "TIM-PN Control Byte W"
  Set_Shunt_On           :="Set_Shunt_ON"
  Set_Torque_Zero        :="Set_Zero_Torque"
  Set_Angle_Zero         :="Set_Zero_Angle"
  Control_Byte           :="Control_byte_value"
```

10 PROFINET-Diagnose mittels S7-Anwenderprogramm

Detaildiagnose durch Auslesen der Datensätze von den TIM-PN Geräten.

- OB82 (Diagnosealarm) mit SFB54 „RALARM“ (Receive Alarm): Der Organisationsbaustein 82 wird bei asynchronen Fehlern bezüglich des Diagnosealarms aufgerufen. Fehler die nicht in ihrem zeitlichen Auftreten dem Programmablauf zugeordnet werden können, werden als „Asynchrone Fehler“ bezeichnet. (vgl. [1], Kapitel 5.9, Seite 238f)

  Der Systembaustein 54 liest zusätzlich Alarminformationen von den auslösenden Baugruppen oder Modulen. Der Aufruf erfolgt innerhalb eines Alarmorganisationsbausteins wie beispielsweise OB82. Die Bearbeitung von RALARM erfolgt synchron. Dies bedeutet, dass die angeforderten Daten (TINFO und AINFO) unmittelbar nach dem Aufruf an den Ausgangsparametern zur Verfügung stehen (vgl. [1], Kapitel 5.7.7, Seite 222ff). RALARM wird für die TIM-PN Diagnosedatensätze nicht benötigt – es dient hier lediglich als Ergänzung.

  Hier wird das Flag zum Starten des RDREC gesetzt.

- Weckalarme (bei S7-1500 ¿ 20 OBs) OB30 bis OB38 und noch weitere ab OB123: Ereignisklasse

  „Cyclic Interrupt“ ist ein Ereignis, welches in periodischen Zeitabständen (konfigurierbar) ausgelöst wird. Die Abarbeitung des Programms im Weckalarm-OB erfolgt unabhängig von der Bearbeitung des zyklischen Programms (z.B. OB1). (vgl. [1], Kapitel 5.7, ab Seite 215)

  Hier wird der asynchron laufende Funktionsbaustein RDREC ausgeführt. Dies ist empfehlenswert, da die Ausführung im Haupt-, Alarm- oder Fehlerprogramm zu Verzögerungen in der Zyklusbearbeitungszeit führen kann. (vgl. [1], Kapitel 5.5, Seite 184)

Information

Die Bearbeitungszeit eines Weckalarms muss deutlich kleiner sein als dessen zeitliches Raster, sonst wird der OB80 (Zeitfehler) aufgerufen. Ist der OB80 nicht vorhanden, wird der Fehler ignoriert.

10.1 Adressierung des TIM-PN Diagnosedatensatzes (PROFINET-IO)

Mit Hilfe des Systembausteins „RDREC“ wird ein Datensatz mit der Nummer INDEX von der Baugruppe gelesen und im Datenbereich RECORD abgelegt. Der INDEX um die Diagnosedaten des TIM-PN abzurufen ist nachfolgend definiert:

0xF80C: 0xF00C:	Liefert alle Diagnosedaten des Geräts Liefert alle Diagnosedaten für ein API (Application Process Identifier)
0xE00C: 0xC00C:	Liefert alle Diagnosedaten für eine AR (Application Relation) Liefert alle Diagnosedaten für einen Slot (Steckplatz)
0x800C:	Liefert alle Diagnosedaten für einen SubSlot (Substeckplatz).

10.2 Struktur der Diagnosedatensätze

Die nachfolgende Abbildung (*Abb. 10.1*) zeigt die verwendete Datensatzstruktur für die TIM-PN Diagnosemeldungen. Die kanalabhängigen Diagnosedaten werden im USI (Byte 19 und 20) im Format I (W#16#8000) übertragen. Dieses entspricht der verwendeten Struktur (*siehe Abb. 10.2: Quellcode der Diagnosestruktur*).

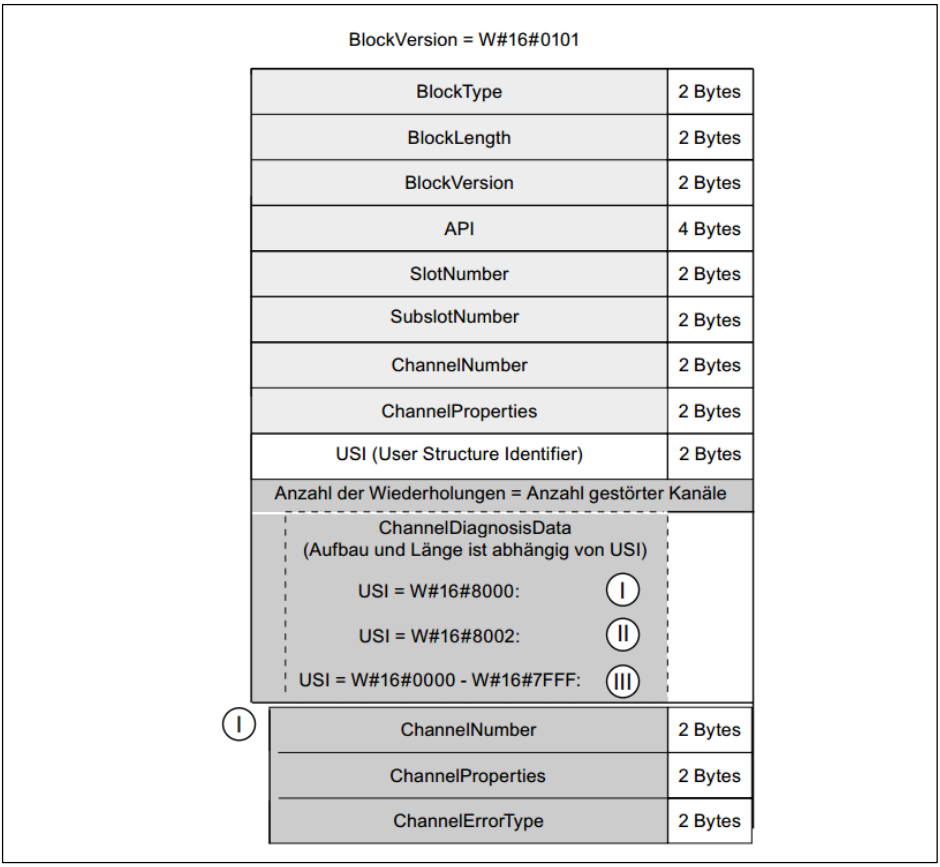

Abb. 10.1 Aufbau des Diagnose-Records

```
/** \brief This structure describes the structure of the PROFINET Channel Diagnosis data.
           For a detailed description of the structure parameters see the specification IEC61158 Part5/6
struct SDAI_PN_PROFINET_CHANNEL_DIAGNOSIS
{
  U16   UserStructureIdent;
  U16   ChannelNumber;
  U16   ChannelProperties;   /**< This parameter is handled completely by the stack */
  U16   ChannelError;
};
```

Abb. 10.2 Quellcode der Diagnosestruktur

10.3 Programmablauf

Die Steuerung beginnt nach dem Initialisierungsschritt mit der zyklischen Programmbearbeitung – hier „Main (OB1)“. Der Diagnosealarm (OB82) wird azyklisch aufgerufen, wenn beispielsweise beim TIM-PN eine Diagnosemeldung geworfen wird. Innerhalb des OB82 wird darauf hin, dass „enable read“ (REQ = 1) für das Auslesen der Diagnoseinformationen des TIM-PN gesetzt. Der Aufruf des asynchron arbeitenden SFB52 erfolgt in dem zeitgesteuerten Organisationsbaustein „Cyclic interrupt (OB30)“. Nach dem erfolgreichen Auslesen des Diagnose-Record, werden die Diagnosemeldungen extrahiert und bei diesem Beispiel in einen Ringspeicher abgelegt. Die nachfolgende Abbildung (*Abb. 10.3*) veranschaulicht den Programmablauf:

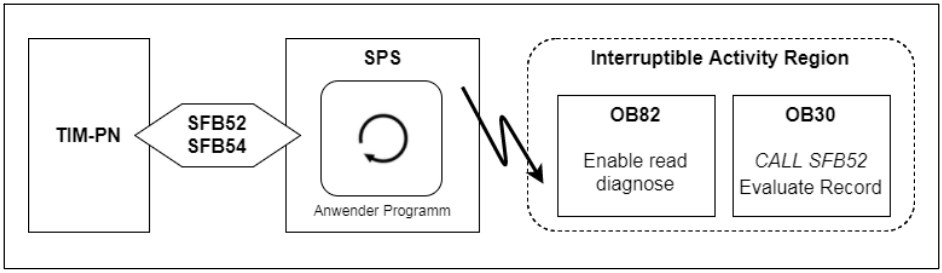

Abb. 10.3 Programmablauf beim Auslesen der Diagnosedaten

Die genaue Bedeutung jedes Bytes und weiterführende Diagnosemöglichkeiten der Records (Datensätze bei PROFINET IO), sowie die „ChannelDiagnosis-Data“ (ChannelNumber, ChannelProperties und ChannelErrorTypes) werden in dem zugehörigen Siemens Handbuch genau beschrieben. (vgl. [4], Kapitel 5.5 Blöcke der Diagnose- und Konfigurationsdatensätze) Dort finden Sie auch weitere Beispiele und Erklärungen.

Der nachfolgende Abschnitt (*siehe Kapitel 10.4*) beschreibt lediglich die Herstellerspezifischen Diagnosemeldungen des TIM-PN.

10.4 Beschreibung der Fehlercodes zu den Channel-Error-Types

Fehlercode	Mapping zu Modul	Kanal-fehler Typ	Fehlertext
TIM_Parameter_Error	Head module	16	Parameter Error Diagnosis: This is a TIM Parameter Error
Measurement Failure	Head module	256	Measurement Failure Diagnosis: This is a Measurement Failure
Torque_Channei1_Failure	Torque module	257	Torque Channel Failure 1 Diagnosis: This is a Torque ChannelFailure 1
Torque_Channei2_Failure	Torque module	258	Torque Failure Channel 2 Diagnosis: This is a Torque ChannelFailure 2
Speed_Channei1_Failure	Speed Module	259	Speed Channel Failure 1 Diagnosis: This is a Speed ChannelFailure 1
Speed_Channei2_Failure	Speed Module	260	Speed Channel Failure 2 Diagnosis: This is a Speed ChannelFailure 2
Angle_Channei_Failure	Speed Module	261	Angle Channel Failure Diagnosis: This is a Angle ChannelFailure
Power_Channel_Failure	Speed Module	262	Power Channel Failure Diagnosis: This is a Power ChannelFailure
Rotor_Temperature_Failure	Torque module	263	Temperature Channel Failure Diagnosis: This is a Temperature Channel Failure

Fehlercode	Mapping zu Modul	Kanal-fehler Typ	Fehlertext
Stator_Error	Head module	264	Stator Error Diagnosis: This is a Stator Error
Rotor Error	Head module	265	Rotor Error Diagnosis: This is a Rotor Error
TIM-Failure	Head module	266	TIM Failure Diagnosis: This is a TIM Failure
Parameter_Selection_Failure	Head module	267	Parameter Selection Failure Diagnosis: This is a Parameter Selection Failure

Tab. 10.1 Fehlercode Diagnose Mapping (vgl. [2], Seite 86f)

10.5 Quellcode in SCL

Erstellt mit Siemens STEP7 und TIA-Portal Version 13 SP2.

DB_Cyclic_read_diagnostic [DB5]		
Name	**Datentyp**	**Startwert**
Static		
Start_read_diag	Bool	
Busy	Bool	
Error	Bool	
Status	Dword	
Index	Dint	
len	Uint	
valid	Bool	
parameter_read	Array[1..128] of Word	
parameter_read[1]	Word	16#0
…	…	…
parameter_read[128]	Word Array[0..15] of	16#0
ChannelDiagnoseArray	Struct	
ChannelDiagnoseArray[0]	Struct	
ChannelNumber	Word	16#0
ChannelProperties	Word	16#0
ChannelErrorType	Word	16#0
…	…	
ChannelDiagnoseArray[15]	Struct	
ChannelNumber	Word	16#0
ChannelProperties	Word	16#0
ChannelErrorType	Word	16#0
IndexCDA	Int	0

Diagnostic error interrupt [OB82]

```
//For additional diagnoses information call RALARM...

//Set flag for start reading diagnose record
"DB_Cyclic_read_diagnostic".Start_read_diag := true;
```

Cyclic interrupt [OB30]		
Name	**Daten-typ**	**Kommentar**
Input		
Initial_Call	Bool	Initial call of this OB
Event_Count	Int	Events discarded
Temp	Int	loop counter variable
i usedLength	Int	while loop break condition
blockLength	Int	length of the block
blockVersion	Word	version of the block
usi	Word	user structure identifier
blockLengthIndex	Int	index of the next blockLength entry
blockVersionIndex	Int	index of the next blockVersion entry
usiIndex	Int	index of the next USI entry
Constant		
positionBlockLength = 2	Int	Offset for BlockLength index at start of the block
positionBlockVersion = 3	Int	Offset for BlockVersion index at start of the block
positionUSI = 10	Int	Offset for USI index at start of the block
upperLimitCBA = 15	Int	upper limit of the circular buffer index

```
// Read Diagnose
"RDREC_DB_diag"(REQ:="DB_Cyclic_read_diagnostic".Start_read_diag,
    ID:="tim-pn~Head",
    INDEX:=W#16#F80C,
    MLEN:=256,
    VALID=>"DB_Cyclic_read_diagnostic".valid,
    BUSY=>"DB_Cyclic_read_diagnostic".Busy,
    ERROR=>"DB_Cyclic_read_diagnostic".Error,
    STATUS=>"DB_Cyclic_read_diagnostic".Status,
    LEN=>"DB_Cyclic_read_diagnostic".len,
    RECORD:= "DB_Cyclic_read_diagnostic".parameter_read);

//evaluate received channel diagnose Record
IF "DB_Cyclic_read_diagnostic".valid = TRUE
THEN
    //set start values
    #usedLength := 0;
    #blockLengthIndex := #positionBlockLength;
    #blockVersionIndex := #positionBlockVersion;
```

```
    #usiIndex := #positionUSI;

    //evaluate as long as blocks available or exit if there is a wrong
    block version
    WHILE (#usedLength < UINT_TO_INT("DB_Cyclic_read_diagnostic".len))
    DO
       #blockLength :=
WORD_TO_INT("DB_Cyclic_read_diagnostic".parameter_read[#blockLengthIndex
]);      //Byte 3 and 4 => Block length
       #blockVersion :=
"DB_Cyclic_read_diagnostic".parameter_read[#blockVersionIndex];      //B
yte 5 and 6 => Block version

       IF #blockVersion = W#16#0101    //check version for data mapping
    THEN
         #usedLength := #usedLength + (4 + #blockLength);      //16+4
(header) are fix because of version
         //32     :=      0       +       (4 + 28)       //will
be the offset for the second block
         //58     :=      32      +       (4 + 22)

         #usi := "DB_Cyclic_read_diagnostic".parameter_read[#usiIndex];
         //read
USI
         IF #usi = W#16#8000     //check

ChannelDiagnoseData format
         THEN
           //start read diagnose 6 diagnose bytes per channel
           FOR #i := (#usiIndex + 1) TO (#usedLength / 2) BY 3   //divi
           de
by two because of word array
           DO
             //check upper limit of circular buffer index
             IF "DB_Cyclic_read_diagnostic".IndexCDA > #upperLimitCBA
             THEN
                "DB_Cyclic_read_diagnostic".IndexCDA := 0;
             END_IF;

               //ChannelNumber
"DB_Cyclic_read_diagnostic".ChannelDiagnoseArray["DB_Cyclic_read_diagnos
tic".IndexCDA]. ChannelNumber :=
"DB_Cyclic_read_diagnostic".parameter_read[#i];

             //ChannelProperties
"DB_Cyclic_read_diagnostic".ChannelDiagnoseArray["DB_Cyclic_read_diagnos
tic".IndexCDA]. ChannelProperties :=
"DB_Cyclic_read_diagnostic".parameter_read[#i + 1];

             //ChannelErrorType
```

```
"DB_Cyclic_read_diagnostic".ChannelDiagnoseArray["DB_Cyclic_read_diagnos
tic".IndexCDA]. ChannelErrorType :=
"DB_Cyclic_read_diagnostic".parameter_read[#i + 2];

              //increment circular buffer index
              "DB_Cyclic_read_diagnostic".IndexCDA :=
"DB_Cyclic_read_diagnostic".IndexCDA + 1;
           END_FOR;
         END_IF;
      ELSE

         //in case of wrong version, we have to exit the loop because we
         cannot calculate the index of the next block
         EXIT;
      END_IF;
      //calculate index of the next block
      #blockLengthIndex := #positionBlockLength + (#usedLength / 2);
      #blockVersionIndex := #positionBlockVersion + (#usedLength / 2);
      #usiIndex := #positionUSI + (#usedLength / 2);
   END_WHILE;
   //reset start flag for acyclic read of RDREC (will be set in OB82)
   "DB_Cyclic_read_diagnostic".Start_read_param := false;
END_IF;
```

www.hbm.com

HBM Test and Measurement
Tel. +49 6151 803-0
Fax +49 6151 803-9100
info@hbm.com

measure and predict with confidence

A05016_01_X00_00 HBM: public