SIEB & MEYER

Drive System SD2x

TIA SD2 SERVOLINK 4 Drive Protocol

Description of the bus system application for the handling of the data exchange in SD2x

D-00000245.3
2024-06-06

Trademarks

All product, font and company names mentioned in this manual may be trademarks or registered trademarks of their respective companies.

SIEB & MEYER Worldwide

For questions regarding our products and technical problems please contact us.

SIEB & MEYER AG
Auf dem Schmaarkamp 21
21339 Lueneburg
Germany

Phone: +49 4131 203 0
Fax: +49 4131 203 2000
info@sieb-meyer.de
http://www.sieb-meyer.com

SIEB & MEYER Shenzhen Trading Co. Ltd.
Room A208 2/F,
Internet Innovation and Creation Services Base Building (2),
No.126, Wanxia road, Shekou, Nanshan district,
Shenzhen City, 518067
P.R. China

Phone: +86 755 2681 1417 / +86 755 2681 2487
Fax: +86 755 2681 2967
info@sieb-meyer.cn
http://www.sieb-meyer.cn

SIEB & MEYER Asia Co. Ltd.
5 Fl, No. 578, Sec. 1
Min-Sheng N. Road
Kwei-Shan Hsiang
Guishan Dist., Taoyuan City 33393
Taiwan

Phone: +886 3 311 5560
Fax: +886 3 322 1224
info@sieb-meyer.tw

1 License Information

The SIEB & MEYER example programs described in this document are subject to the following license conditions.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Any deviations from these conditions require written permission from the copyright holder in advance.

Disclaimer

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS, AUTHORS, AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2 Description of the Program

This document describes the handling of the cyclic process data and the service channel between PLC and SD2x via a gateway as well as the access to the safety functions of SD2x as extension.

Notes for the described program examples:

- The program examples described in this document are used for the different bus systems.
- The program sequence is controlled by the commands in the organization block OB1 (Main).
- In order to keep the program examples simple, we did not include the programming of the error evaluation. If required, the user must program these additional evaluations himself.

2.1 System Components

The PLC program has been created using the engineering tool TIA Portal by the company Siemens. All function blocks and functions necessary for the communication are included in the program.

In order to use the program for the communication between PLC and the SD2x drives by SIEB & MEYER, you need the following hardware and software components:

- TIA Portal V15.1
- Siemens PLC S7-1500
  - CPU 1511 C-1 PN
  - AI5 / AQ2_1
  - DI16 / DQ16_1
  - CP1542-5_1
- SIEB & MEYER SERVOLINK 4 gateway
  - Gateway 0362151(A) for PROFIBUS DP
    (Up to 12 drives can be connected to the gateway.)
  - Gateway 0362157(A) for PROFINET IO
    (Up to 24 drives can be connected to gateway 0362157 and up to 16 drives to gateway 0362157A.)
- SIEB & MEYER SD2x

The used PLC communicate via the according fieldbus system with the gateway. The gateway forwards the information to the SD2x drives via .

2.2 Available Example Programs

The following example program is available:

- TIA_SD2_GATEWAY for PROFIBUS DP and PROFINET IO

2.3 Installation

1. At first create a working directory for the application programs. Example:C:\SM_AG \workspace

2. Copy the ZAP file of the desired program (e.g. “TIA_SD2_GATEWAY_20190418_0825.zap15_1”) into the created directory.
3. Start the TIA Portal user interface and switch to the project view.
4. Select the menu “Project → Retrieve …”.
5. Go to the created directory and select the ZAP file. Click “Open” to confirm your selection. A window appears, in which you can select the target directory.
6. Select the desired project directory (e.g. C:\SM_AG\workspace) and click “Select folder”. Then, the ZAP file is retrieved.

✓ The project is saved in a subdirectory of the working directory (e.g.. C:\SM_AG\workspace\TIA_SD2_GATEWAY). Afterwards, the project will be available in the user interface as well.

3 Program Content

The PLC program example contains the following program blocks as SCL sources.

3.1 Organization Block

Symbolic name	Number	Description
Main	OB1	Main program

3.2 Function Blocks

3.2.1 General Function Blocks

Symbolic name	Number	Description
FB_Init_FBs	FB6	Initializes the other function blocks
FB_SwapDINT	FB10	Swaps the byte order of a 4 byte integer value
FB_SwapINT	FB11	Swaps the byte order of a 2 byte integer value

3.2.2 Function Blocks to Control the Drive

Symbolic name	Number	Description
FB_InitData	FB5	Initializes the fieldbus data
FB_ReadData	FB7	Analyzes the cyclic fieldbus data for the drive structure
FB_WriteData	FB12	Compiles the fieldbus data from drive structure

3.2.3 Function Blocks of the Service Channel

Symbolic name	Number	Description
FB_ReadObject	FB8	Reads an object from the drive
FB_SetArrayIndex	FB9	Sets the array index counter in the drive
FB_WriteObject	FB13	Writes on an object in the drive

3.2.4 Funktion Blocks for the SC Objects 467 to 470

Symbolic name	Number	Description
FB_SC_CheckObject	FB1	Checks the input data of the object
FB_SC_FillObject	FB2	Fills the object with the given data
FB_SC_HandleObject	FB4	Controls the object accesses
FB_SC_Handle	FB3	Controls the access to a drive

3.2.5 Function Blocks of the TIA Portal System

Function block	Number	Description
TON	IEC_-Timer_0_DB	Generate switch-on delay; used for time monitoring

3.3 Functions

Symbolic name	Number	Description
FC_GetStatus	FC1	Determines the device state by means of the drive status word

3.4 Instance Data Blocks

Symbolic name	Number	Description
FB_SC_Handle_DB	DB7	Instance data block for FB_SC_Handle
FB_InitFBs_DB	DB5	Instance data block for FB_Init_FBs
FB_ReadData_DB	DB8	Instance data block for FB_ReadData
FB_WriteData_DB	DB9	Instance data block for FB_WriteData

3.5 Global Data Blocks

Symbolic name	Number	Description
DB_DRIVESTATE	DB1	List of constants for the drive status according to the drive protocol DS402
DB_ERRORCODE	DB2	List of Constants for Error Codes
DB_GLOBALDATA	DB4	Global used variables
DB_SC_PROCESSDATA	DB3	Input data and output data for FB_SC_HandleObject

3.6 User-defined Data Types

Symbolic name	Description
TS_DRIVE_STATUSWORD	Data structure of the drive status word
TS_DRIVE_DATA_IN	Data structure of the received data from the fieldbus system
TS_DRIVE_DATA_OUT	Data structure of the transmitted data to the fieldbus system
TS_DRIVE_DATA	Data structure of the drive data
TS_DRIVE_ACTUALVALUES	Data structure of the actual values
TS_DRIVE_SETPOINTS	Data structure of the setpoints
TS_OBJECTDATA	Data structure for the object access
TS_HANDLE_DATA	Data structure for the control of a function block
TS_OBJECTDATA_SC	Structure of the data array for the SC objects
TS_SC_PROCESSDATA	Data structure for the access to the SC objects

4 Program Control

This chapter provides information on the individual blocks and their functions.

4.1 Program Organization Block

4.1.1 Main

The program is controlled via the organization block Main. Here, the data from the fieldbus is read, allocated to the drives, processed and written to the fieldbus again. The individual program steps are processed by the function blocks.

Before the proper sequential control comes into operation, the data must be initialized ones when the program is started or after a fault.

4.1.1.1 Assignment of the Fieldbus Data

In the sequential program, the fieldbus data defined in the PLC variables can be assigned to the program variables.

Example for the data of drive 0:

```
// assigning the drive number
"DB GLOBALDATA".DriveNumber := 0;
// reading the drive actual values
"DB GLOBALDATA".SD Drive["DB GLOBALDATA".DriveNumber].InData := "InData 0";
. . .
// output of drive reference values
"Outdata_0" := "DB_GLOBALDATA".SD_Drive["DB_GLOBALDATA".DriveNumber].OutData;
```

4.1.1.2 Evaluation of the Digital Input I_Speed_0

The input I_Speed_0 specifies the speed setpoint. The according procedure is described in chapter 8.2.1 “Motor Control”, page 47.

4.1.1.3 Extension of the Block Main for Several Drives

If more than one drive is used, the program block labeled with “Drive 0” must be copied and adapted. The following adaptations are required for each new program block:

- increase variable DrNo by 1
- define additional inputs and outputs for the drive control
- adapt setpoints for speed and current

Example

DrNo	Drive	SD2x address	TIA hardware configuration
0	Single	0	Servo drive 16 in / 16 out
1	Single	1	Servo drive 16 in / 16 out
2	Double A	2	Servo drive 16 in / 16 out
3	Double B		Servo drive 16 in / 16 out

DrNo	Drive	SD2x address	TIA hardware configuration
4	Double A	3	Servo drive 16 in / 16 out
5	Double B		Servo drive 16 in / 16 out
6	Single	4	Servo drive 16 in / 16 out
⋮	⋮	⋮	⋮

Note

Make sure the each double-axis drive starts with an even DrNo.

4.1.1.4 Safety Control-Funktionen

This program part is used to forward the data to the safety control functions of the drives via the function block FB_SC_Handle.

Note

If you do not require the safety control functions for the safety functions SFM and SLOF in the SD2x drives, you can delete this program part as well as the function blocks required for the safety functions.

For full data transmission and evaluation the PLC must always address both safety control functions of the drive (SC0 and SC1). For this purpose the data are processed step-by-step for each drive:

0	Initialization of the function blocks
1	Write setpoint “limit speed zero”, SC0
2	Write setpoint “limit speed zero”, SC1
3	Write setpoint “safe limited output frequency”, SC0
4	Write setpoint “safe limited output frequency”, SC1
5	Read actual value “safe limited output frequency”, SC0
6	Read actual value “safe limited output frequency”, SC1
7	Read status, SC0
8	Read status, SC1
9	Status evaluation
10	Customized program sequence

Afterwards the list items 5 to 10 are repeated. By changing the transfer parameters to FB_SC_Handle you can send other commands to the safety control functions.

Transmission Process

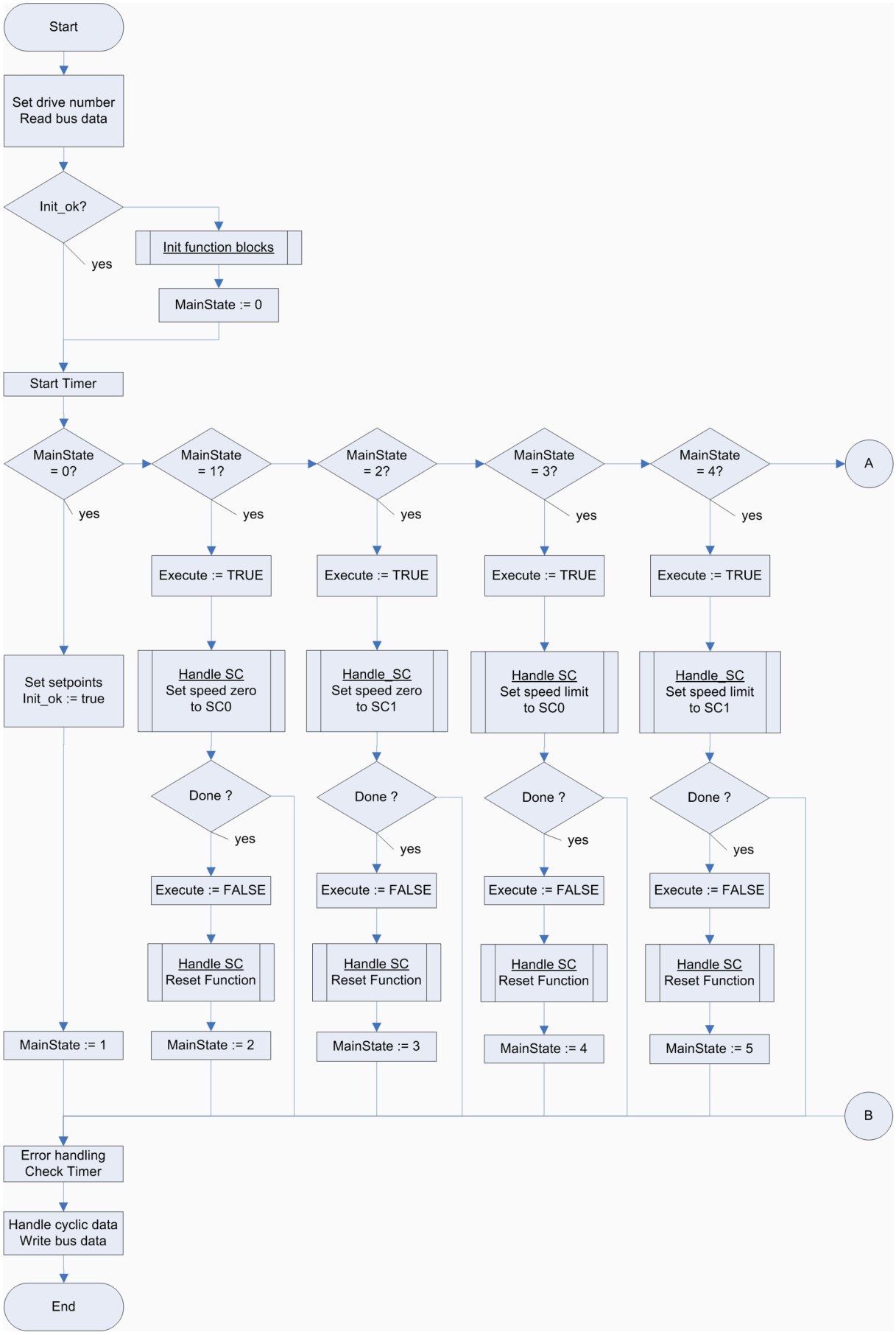


Fig. 1: Safety control functions: send data

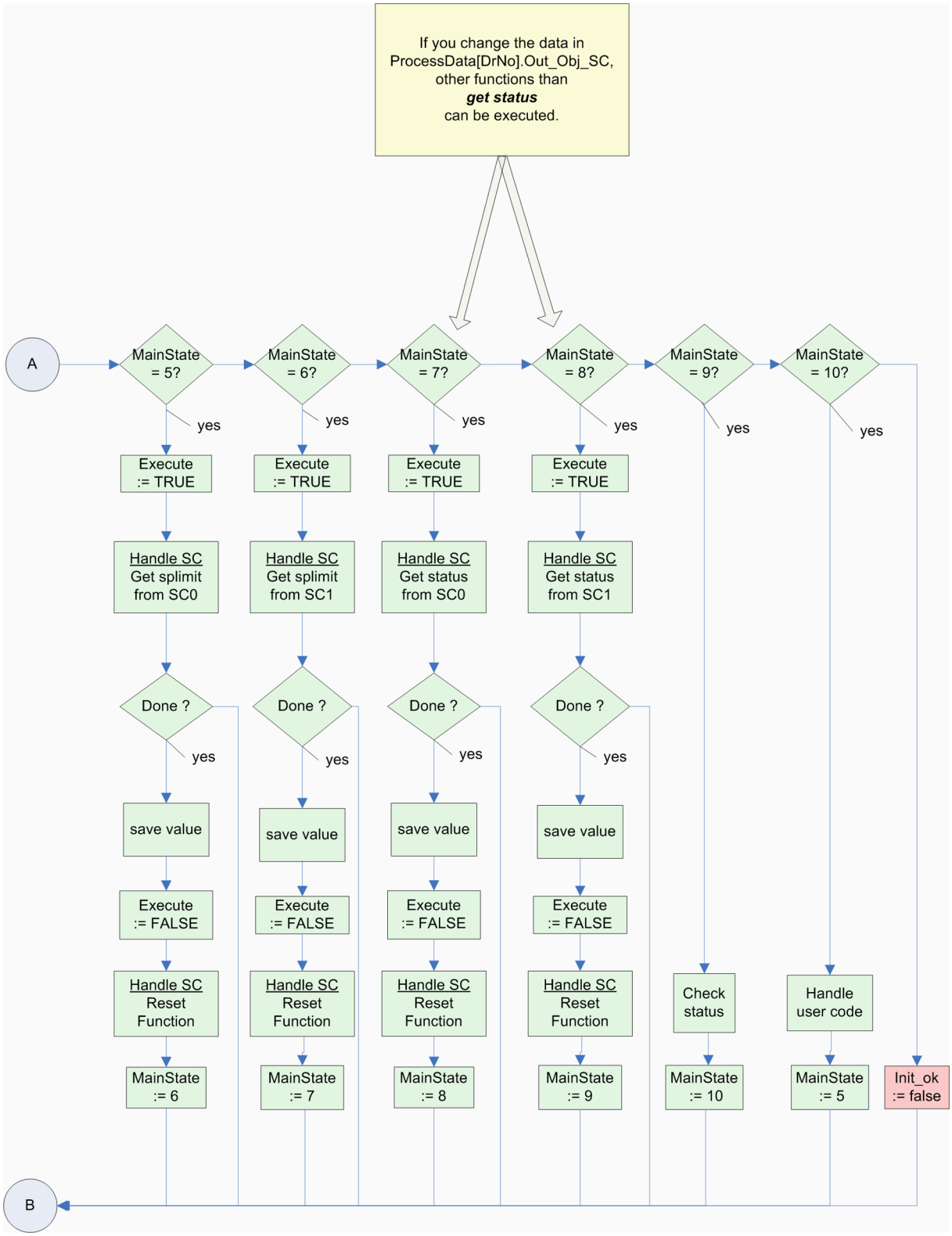


Fig. 2: Safety control functions: read data

Tip

You can define default values for “speed zero” and “safe limited output frequency” in the SD2x parameter sets. These values are sent to the safety control function when the SD2x firmware is started. If you use the default values, the PLC can skip writing the values (list items 1 to 4). Instead the PLC can directly read the actual values from the safety control functions when they are required.

0	Initialization of the function blocks
5	Read actual value “safe limited output frequency”, SC0
6	Read actual value “safe limited output frequency”, SC1
7	Read status, SC0
8	Read status, SC1
9	Status evaluation
10	Customized program sequence

4.2 Function Blocks

4.2.1 Sequential Control for Several Function Blocks

Some function blocks must be called repeatedly with each PLC cycle until the function is completely executed. This applies particularly to the function blocks for reading and writing of the SD2x objects. For sequential control of these function blocks the variables of the user-defined data type TS_HANDLE_DATA are used. They are specified in the input and output data structures. Before actually using the function blocks, they should be called once with EXECUTE = FALSE in the function block FB_Init_FBs.

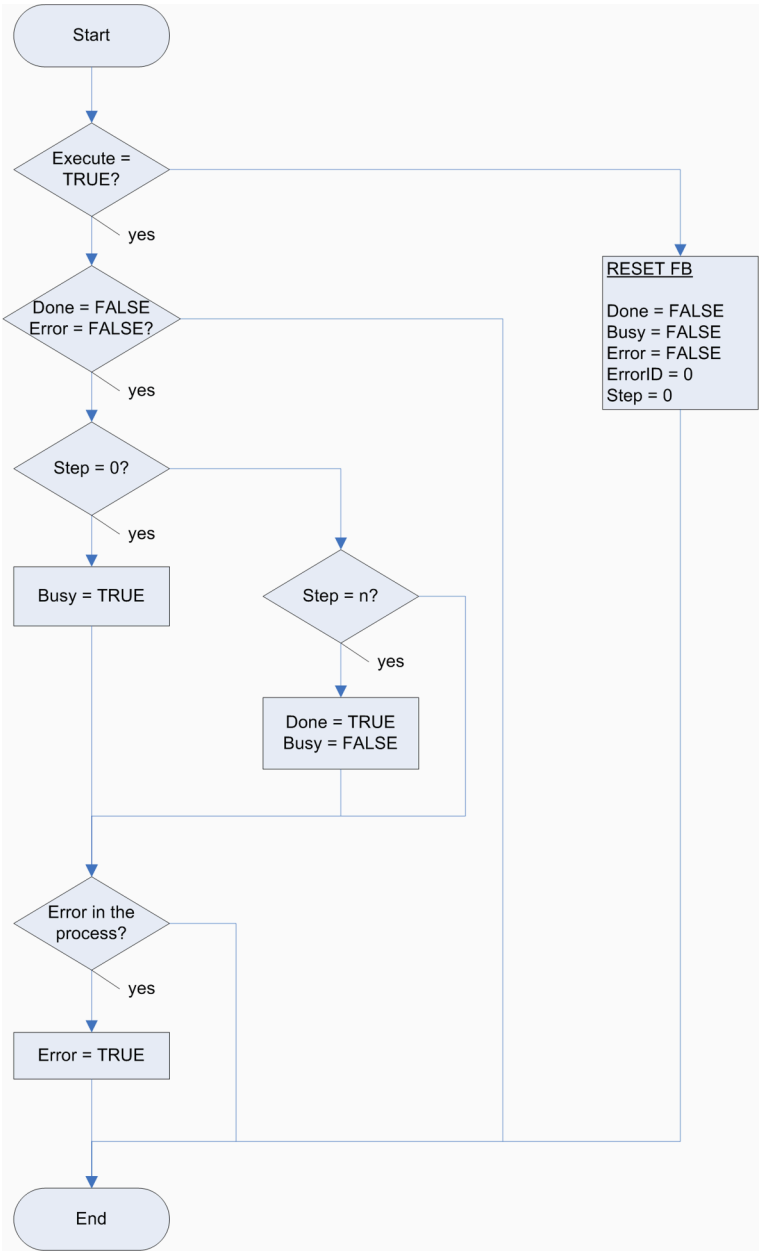


Fig. 3: Sequential control of the function blocks

4.2.2 Function block FB_Init_FBs

4.2.2.1 General Description

The function block FB_Init_FBs initializes the other function blocks. After the initialization is completed successfully init_ok of the according drive is set to TRUE.

4.2.2.2 Calling the Function Block

```
"FB_Init_FBs"(DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.2.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number

4.2.3 Function block FB_InitData

4.2.3.1 General Description

Using the function block FB_InitData the command bits of the cyclic protocol are initialized and deposited in the drive structure. The reference values are initialized with the value zero.

Afterwards the initializing data must be sent to the drive once.

4.2.3.2 Calling the Function Block

```
#FB_InitData(DrNo := #DrNo);
```

4.2.3.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number

4.2.4 Function block FB_ReadData

4.2.4.1 General Description

The function block FB_ReadData processes the cyclic data of the SD2x drive.

The data of the SD2x drive is filtered by the function block so that individual information blocks are created. The information is deposited in the drive structure. Thus it is globally available in the whole application program.

At the output of the block an error information is available.

This function block must be called before each processing of the cyclic data.

4.2.4.2 Calling the Function Block

```
"FB_ReadData_DB"(DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.4.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In/out	INT	Drive number
ErrorID	Out	WORD	Error code

4.2.5 Function block FB_WriteData

4.2.5.1 General Description

The function block FB_WriteData compiles a setpoint telegram from the data in the drive structure and writes the new cyclic parameters and control bits into the drive structure. Then the new data is transmitted to the drive. At the output of the block an error information is upcoming.

This function block must be called after each processing of the cyclic data.

4.2.5.2 Calling the Function Block

```
"FB_WriteData_DB"(DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.5.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number
ErrorID	Out	WORD	Error code

4.2.6 Function block FB_ReadObject

4.2.6.1 General Description

The function block FB_ReadObject allows read access to the object data of the drive via the service channel. Afterwards, the result is available in the global data structure ObjectRdData. The manual “Drive System SD2 – SERVOLINK 4 Bus System Description” provides information on the functional sequence of the access.

4.2.6.2 Calling the Function Block

```
#FB_ReadObject(DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.6.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number

4.2.7 Function block FB_WriteObject

4.2.7.1 General Description

The function block FB_WriteObject allows write access to the object data of the drive via the service channel. This set values must be entered into the global data structure ObjectWrData. The document “Drive System SD2 – SERVOLINK 4 Bus System Description”provides information on the functional sequence of the access.

4.2.7.2 Calling the Function Block

```
#FB_WriteObject(DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.7.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number

4.2.8 Function blockFB_SetArrayIndex

4.2.8.1 General Description

The function block FB_SetArrayIndex is used to set the subindex of an array object. Afterwards, during writing of the object, the subindex is increased automatically. The manual “Drive System SD2 – SERVOLINK 4 Bus System Description” provides information on the functional sequence of the access.

In order to set the array index to the first element, the calling parameter must have the value 4. This value must be entered into the global data structure ObjectAiData.

4.2.8.2 Calling the Function Block

```
#FB_SetArrayIndex (DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.8.3 Inputs and Outputs

Parameter	In / Out	Type	Description
Drive	In	INT	Drive number

4.2.9 Function block FB_SC_Handle

4.2.9.1 General Description

The function block FB_SC_Handle is used to write the commands into the SC functions of the drive via the function blocks FB_SC_FillObject and FB_SC_HandleObject. By calling the function block repeatedly you can execute commands step-by-step, one after the other.

4.2.9.2 Calling the Function Block

```
// handle communication for safety control 0, setting speed zero
"DB GLOBALDATA".SC HandleData["DB GLOBALDATA".DriveNumber].Execute := TRUE;
"FB SC Handle DB"(DrNo:= "DB GLOBALDATA".DriveNumber,
   SC:="DB SC PROCESSDATA".SC0,
   SubIndex:="DB SC PROCESSDATA".PAR SPEED ZERO,
   RdWr:="DB SC PROCESSDATA".WRITE PAR,
   Pattern:=#bPattern,
```

```
   Par:="DB_GLOBALDATA".SD_Drive["DB_GLOBALDATA".DriveNumber].Setpoints.
SpeedZero);
IF "DB GLOBALDATA".SC HandleData["DB GLOBALDATA".DriveNumber].Done THEN
   // reset function block
   "DB_GLOBALDATA".SC_HandleData["DB_GLOBALDATA".DriveNumber].Execute :=
 FALSE;
   "FB SC Handle DB"();
   "DB GLOBALDATA".MainState["DB GLOBALDATA".DriveNumber] := 2;
   "DB GLOBALDATA".Pattern["DB GLOBALDATA".DriveNumber] :=
   "DB GLOBALDATA".Pattern["DB GLOBALDATA".DriveNumber] + 1;
   "IEC Timer 0 DB".TON(IN := FALSE, PT := T#0s);
END_IF;
```

4.2.9.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number
SC	In	BYTE	Number of the safety control function
SubIndex	In	BYTE	Command number
RdWr	In	BYTE	Read or write command
Pattern	In	BYTE	Check pattern
Par	In	DWORD	Default value

4.2.9.4 Sequence of the Function Block

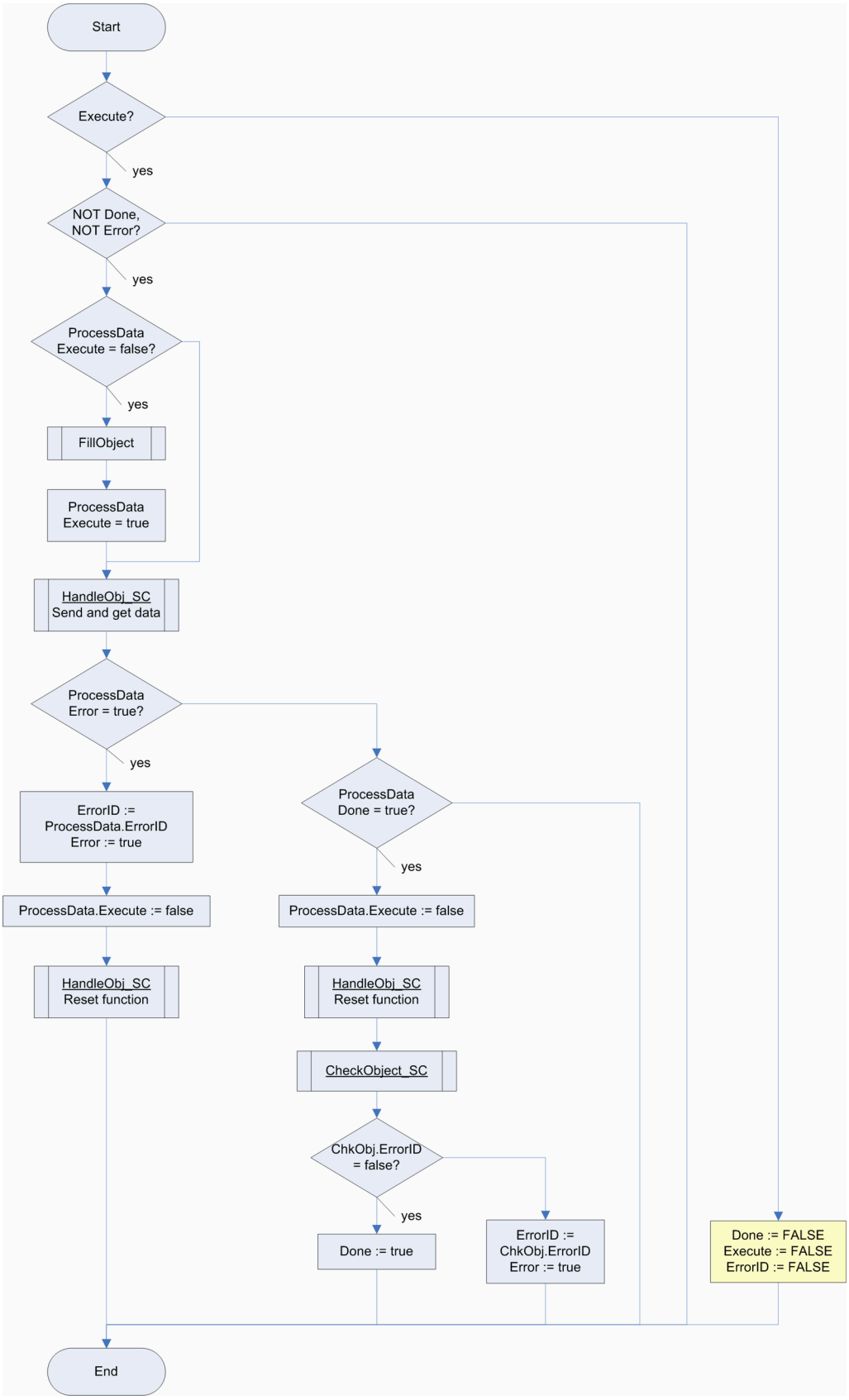


Fig. 4: FB_SC_Handle

4.2.10 Function block FB_SC_FillObject

4.2.10.1 General Description

The function block FB_SC_FillObject is used to allocate the input parameters of the block to the output structure for sending. In addition, the block calls the checksum calculation and writes the result into the structure.

Note

For further information on the parameters refer to the manual “Drive System SD2 – Safety Functions SFM / SLOF” (chapter “Fieldbus Communication”).

4.2.10.2 Calling the Function Block

```
#FB SC FillObject (DrNo := #DrNo,
   SC := #SC,
   SubIndex := #SubIndex,
   RdWr := #RdWr,
   Data := #Par,
   Pattern := #Pattern);
```

4.2.10.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number
SC	In	BYTE	Number of the safety control function
SubIndex	In	BYTE	Command number
RdWr	In	BYTE	Read or write function
Data	In	DWORD	Default value
Pattern	In	BYTE	Check pattern

4.2.11 Function block FB_SC_HandleObject

4.2.11.1 General Description

The function block FB_SC_HandleObject is used to write the safety parameters into the safety control functions via the objects 467 to 468 as well as to read the safety parameters from the safety control functions via the objects 469 the 470.

- object 467: SAFE_SPEED_MONITOR_DEMAND_DATA_OBJECT
- object 468: SAFE_SPEED_MONITOR_DEMAND_CTRL_OBJECT
- object 469: SAFE_SPEED_MONITOR_REPLY_DATA_OBJECT
- object 470: SAFE_SPEED_MONITOR_REPLY_STATUS_OBJECT

The following commands are send one after the other:

- Write object: 4 bytes (setpoint value) in object 467(via FB_WriteObject)
- Write object: 4 bytes (control data) in object 468 (via FB_WriteObject)
- Read object: 4 bytes (actual value) from object 469 (via FB_ReadObject)
- Read object: 4 bytes (status data) from object 470 (via FB_ReadObject)

By means of the two write commands the PLC sends the prompt telegram to the according safety control function of the drive. The required data are to find in the global process data Out_SC_Obj (TS_SC_PROCESSDATA). With the two read commands the PLC reads the response telegram of the according safety control function. The read data are entered into the global process data In_SC_Obj (TS_SC_PROCESSDATA) and can be evaluated.

4.2.11.2 Calling the Function Block

```
#FB_SC_HandleObject (DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.11.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number

4.2.11.4 Sequence of the Function Block

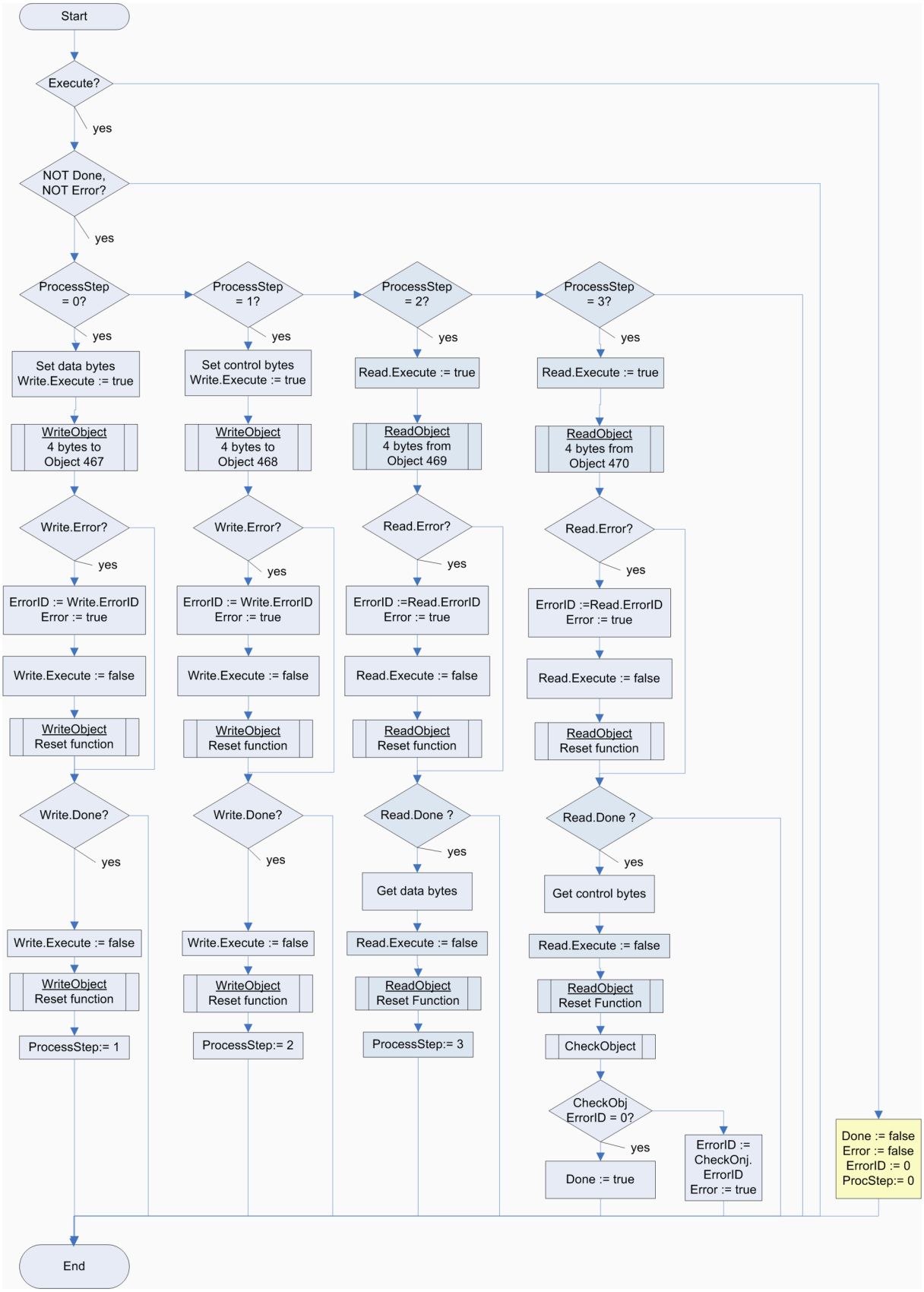


Fig. 5: FB_SC_HandleObject

4.2.12 Function block FB_SC_CheckObject

4.2.12.1 General Description

The function block FB_SC_CheckObject is used to check the addresses, the check patterns and the check sums of the input and output data of the telegram exchange.

For this purpose the check sum of the input data is calculated and compared with the transmitted checksum. If an error occurs, the corresponding error code is returned. Afterwards, the address is checked. The received address must match the transmitted address. A mismatch causes an error and the corresponding error code is returned. At last , the check patterns are compared. For this purpose the output pattern is incremented by 1 and compared to the input pattern. If an error occurs, the corresponding error code is returned.

4.2.12.2 Calling the Function Block

```
#FB_SC_CheckObject (DrNo := "DB_GLOBALDATA".DriveNumber);
```

4.2.12.3 Inputs and Outputs

Parameter	In / Out	Type	Description
DrNo	In	INT	Drive number
ErrorID	Out	WORD	Error code

4.2.13 Function block FB_SwapINT

4.2.13.1 General Description

Using the function block FB_SwapINT the byte order of a 2 byte integer value is swapped. Byte swapping is necessary to allow for the formats big-endian and little-endian used by the devices.

Byte order before:	Byte1, Byte2
Byte order after:	Byte2, Byte1

4.2.13.2 Calling the Function Block

```
#FB_SwapINT(value :="DB_GLOBALDATA".SD_Drive[#DrNo].OutData.ServiceIndex);
```

4.2.13.3 Inputs and Outputs

Parameter	In / Out	Type	Description
value	In / Out	INT	Integer value

4.2.14 Function block FB_SwapDINT

4.2.14.1 General Description

Using the function block FB_SwapDINT the byte order of a 4 byte integer value is swapped. Byte swapping is necessary to allow for the formats big-endian and little-endian used by the devices.

Byte order before:	Byte1, Byte2, Byte3, Byte4
Byte order after:	Byte4, Byte3, Byte2, Byte1

4.2.14.2 Calling the Function Block

```
#FB_SwapDINT (value :="DB_GLOBALDATA".SD_Drive[#DrNo].OutData.ServiceValue);
```

4.2.14.3 Inputs and Outputs

Parameter	In / Out	Type	Description
value	In / Out	DINT	Double integer value

4.3 Functions

4.3.1 FunctionFC_GetStatus

4.3.1.1 General Description

The function FC_GetStatus determines the device status by means of the status word of the drive according to the finite state automaton (see manual “Drive System SD2 - Device Control”). For this purpose the status word is passed to the function. The function returns an enumerated value of DB_DRIVESTATE.

4.3.1.2 Calling the Function Block

```
"DB GLOBALDATA".SD Drive[#DrNo].ActValues.Status :=
   "FC GetStatus"(Status :=
   "DB_GLOBALDATA".SD_Drive[#DrNo].InData.StatusWord);
```

4.3.1.3 Inputs and Outputs

Parameter	In / Out	Type	Description
Status	In	TS_DRIVE_STATUSWORD	Status word
ReturnValue	Return	WORD	Device status

4.4 Global Data Blocks

4.4.1 Data Block DB_DRIVESTATE

The global data block DB_DRIVESTATE contains a list of the drive states according to the drive protocol DS402.

Status	Value	Meaning
STATE_Start	0	Start, initialization
State_NotReadyToSwitchOn	1	Not ready to be switched on
STATE_SwitchOnDisabled	2	Switch on disabled
STATE_ReadyToSwitchOn	3	Ready to be switched on
State_SwitchedOn	4	Switched on
State_OperationEnabled	5	Operation enabled
STATE_QuickStopActive	6	Quick stop
State_FaultReactActive	7	Fault reaction
State_Fault	8	Fault occurred

4.4.2 Data Block DB_ERRORCODE

The global data block DB_ERRORCODE contains the error codes that are returned by the function blocks.

The error codes have the following meanings:

Error code	Value	Meaning
ERR_None	0	No error
ERR_WrongState	1	The drive is in the wrong device status for the up-coming command.
ERR_Parameter	2	Parameter error
ERR_Pattern	3	Check pattern wrong
ERR_Checksum	4	Checksum wrong
ERR_Address	5	Address wrong
ERR_SERVOLINK_Fault	6	SERVOLINK error
ERR_SERVOLINK_Slot	7	SERVOLINK slot error
ERR_TimeOut	8	Timeout

Note

The error variables may contain more error codes. These error codes come from the return codes in the blocks of the service channel. They are listed in the manual “Drive System SD2 – SERVOLINK 4 Bus System Description”.

4.4.3 Data Block DB_GLOBALDATA

The global data block DB_GLOBALDATA contains the program-relevant data that must be adapted to the particular project.

Variable	Type	Meaning
SD_Drive	ARRAY[0..15] OF "TS_DRIVE_DATA"	Structure of the drive data
ObjectRdData	ARRAY[0..15] OF "TS_OBJECTDATA"	Input/output data from function block
ObjectWrData	ARRAY[0..15] OF "TS_OBJECTDATA"	Input/output data from function block
ObjectAiData	ARRAY[0..15] OF "TS_OBJECTDATA"	Input/output data from function block
ReadValue	ARRAY[0..15] OF DINT	Read return value

Variable	Type	Meaning
ProcessData_SC	ARRAY[0..15] OF "TS_SC_PROCESSDATA"	Process data of safety objects
HandleData_SC	ARRAY[0..15] OF "TS_HANDLE_DATA"	Control parameters for the function block
MainState	ARRAY[0..15] OF INT	Step pointer in the program Main
ErrorID	ARRAY[0..15] OF WORD	Global error code
StandStill	ARRAY[0..15] OF BOOL	Status: speed zero
SpeedLimit	ARRAY[0..15] OF BOOL	Status: speed limit
Pattern	ARRAY[0..15] OF INT	Pattern field (incremented after each command transmission)
DriveNumber	INT	Drive number
ChkDataIn	ARRAY[0..7] OF BYTE	Array of the input data to calculate the checksum
tChecksumIn	BYTE	Checksum of the input data
ChkDataOut	ARRAY[0..7] OF BYTE	Array of the output data to calculate the checksum
tChecksumOut	BYTE	Checksum of the output data
oldstate	ARRAY[0..3] OF BYTE	Previous status value
ErrCounter	ARRAY[0..3] OF INT	Error counter
testNo	ARRAY[0..9] OF INT	Debug variable
testIn	ARRAY[0..9] OF BYTE	Debug variable
testOut	ARRAY[0..9] OF BYTE	Debug variable
testi	INT	Debug variable

4.4.4 Data Block DB_SC_PROCESSDATA

The global data block DB_SC_PROCESSDATA contains constants for the data array of the safety telegram.

Parameter	Value	Meaning
SC0	0x00	Address value for the safety control function 0
SC1	0x80	Address value for the safety control function 1
PAR_REPEAT	0	Subindex for "repeat"
PAR_SPEED_ZERO	1	Subindex for "speed zero"
PAR_SPEED_LIMIT	2	Subindex for "safe limited output frequency"
PAR_STATUS	3	Subindex for "status"
PAR_VERSION	4	Subindex for "version"
PAR_COBID	5	Subindex for "COB-ID"
PAR_ACTIV_LIMIT	6	Subindex for "activate limit"
PAR_VERSIONIFO	7	Subindex for "version info"
READ_PAR	0x00	Identifier for the function "read"
WRITE_PAR	0x80	Identifier for the function "write"

4.5 PLC Variables

4.5.1 Standard Variables

The standard variable table contains the variables that are assigned to inputs or outputs of the PLC system.

Name	Data type	Address	Meaning
I_SwOn_0	BOOL	%I10.0	Input “Switch on”
I_EnOp_0	BOOL	%I10.1	Input “Enable operation”
I_QStop_0	BOOL	%I10.2	Input “Quick stop”
I_ResFault_0	BOOL	%I10.3	Input “Fault reset”
I_ResFault_SC	BOOL	%I10.4	Input “Fault reset safety”
I_Speed_0	BYTE	%IB11	Inputs “Speed setpoint”
InData_0	TS_DRIVE_DATA_IN	%I124.0	Fieldbus input data drive 0
InData_1	TS_DRIVE_DATA_IN	%I140.0	Fieldbus input data drive 1
OutData_0	TS_DRIVE_DATA_OUT	%Q126.0	Fieldbus output data drive 0
OutData_1	TS_DRIVE_DATA_OUT	%Q142.0	Fieldbus output data drive 1

4.6 User-defined Data Types

The user-defined data types describe the data structure of each drive. These are the input and output values of the fieldbus data and the prepared setpoints and actual values.

The data can be accessed via the global variables.

The structure of TS_DRIVE_DATA is based on the standards by the PLCopen group. The structures of the fieldbus parameters are based on the CANopen protocol. A description of the telegram structure can be found in the manuals “Drive System SD2 – CAN Bus Connection” and “Drive System SD2 – Device Control”.

4.6.1 User-defined Data Type TS_DRIVE_STATUSWORD

The user-defined data type TS_DRIVE_STATUSWORD contains the data structure of the drive status word.

Name	Type	Meaning
Status word byte 0..1		
ReadyToSwitchOn	BOOL	Ready to be switched on
SwitchedOn	BOOL	Switched on
OperationEnabled	BOOL	Operation enabled
Fault	BOOL	Fault occurred
VoltageEnabled	BOOL	Voltage enabled
QuickStop	BOOL	Quick stop
SwitchOnDisabled	BOOL	Switch on disabled
Warning	BOOL	Warning
smres010	BOOL	Reserved
Remote	BOOL	Remote mode
TargetReached	BOOL	Target reached
InternalLimitActive	BOOL	Internal limit reached
OperationModeSpecific0	BOOL	Dependent on the operating mode
OperationModeSpecific1	BOOL	Dependent on the operating mode
IstwertTelegrammKennung1	BOOL	Code of actual value telegram, set to 1 by the drive
IstwertTelegrammKennung0	BOOL	Code of actual value telegram, set to 0 by the drive

4.6.2 User-defined Data Type TS_DRIVE_DATA_IN

The user-defined data type TS_DRIVE_DATA_IN contains the data structure of the received data of a drive from the fieldbus system.

Name	Type	Meaning
Status word byte 0..1		
StatusWord	TS_DRIVE_STATUSWORD	Current status word
Actual values byte 2..9		
Act_Position	DINT	Actual position
Act_Velocity	INT	Actual velocity
Act_Current	INT	Actual current
Service channel byte 10..15		
ServiceReturn	DINT	Return parameters / error code
res14	BYTE	Reserved
ServiceDoneToggle	BOOL	Return toggle bit
ServiceFault	BOOL	Error
smres152	BOOL	Reserved
smres153	BOOL	Reserved
smres154	BOOL	Reserved
smres155	BOOL	Reserved
smres156	BOOL	Reserved
smres157	BOOL	Reserved

4.6.3 User-defined Data Type TS_DRIVE_DATA_OUT

The user-defined data type TS_DRIVE_DATA_OUT contains the data structure of the transmission data from a drive to the fieldbus system.

Name	Type	Meaning
Control word byte 0..1		
SwitchOn	BOOL	Switch on power unit
EnableVoltage	BOOL	Enable voltage at the power unit
QuickStop	BOOL	Quick stop
EnableOperation	BOOL	Enable operation
Mode0	BOOL	Operation mode bit 0
Mode1	BOOL	Operation mode bit 1
Mode2	BOOL	Operation mode bit 2
FaultReset	BOOL	Error reset
Hold	BOOL	Hold
res011	BOOL	Reserved
res012	BOOL	Reserved
smres013	BOOL	Reserved
smres014	BOOL	Reserved
smres015	BOOL	Reserved
SollwertTelegrammID0	BOOL	Code of setpoint telegram, must be set to 0 by the PLC
SollwertTelegrammID1	BOOL	Code of setpoint telegram, must be set to 1 by the PLC
Setpoints byte 2..7		
res02	INT	Reserved
Spt_Velocity	INT	Velocity setpoint
Spt_Current	INT	Maximum current setpoint
Byte 8		
res08	BYTE	Reserved
Service channel byte 9..15		
ServiceValidToggle	BOOL	New service command
ServiceFunction0	BOOL	Service function
ServiceFunction1	BOOL	

Name	Type	Meaning
ServiceByteIndex0	BOOL	Number of bytes
ServiceByteIndex1	BOOL	
smres095 to smres097	3 * BOOL	Reserved
ServiceIndex	INT	Object number
ServiceValue	DINT	Object value / array index

4.6.4 User-defined Data Type TS_DRIVE_DATA

The user-defined data type TS_DRIVE_DATA contains the data structure of the drive data based on the PLCopen standard.

Name	Type	Meaning
init_ok	BOOL	Initializing status
ActValues	TS_DRIVE_ACTUALVALUES	Actual values of the drive
Setpoints	TS_DRIVE_SETPOINTS	Setpoints of the drive
InData	TS_DRIVE_DATA_IN	Data block (bus system->PLC)
OutData	TS_DRIVE_DATA_OUT	Data block (PLC->bus system)

4.6.5 User-defined Data Type TS_DRIVE_SETPOINTS

The user-defined data type TS_DRIVE_SETPOINTS contains the data structure of the setpoints of the drive.

Name	Type	Meaning
FaultReset	BOOL	TRUE = reset error
SwitchOn	BOOL	TRUE = switch on output stage
EnableOperation	BOOL	TRUE = enable operation
EnableVoltage	BOOL	TRUE = enable voltage
QuickStop	BOOL	FALSE = trigger quick stop
Velocity	INT	Setpoint velocity, max. 0x3FFF
Current	INT	Setpoint current, max. 0x3FFF
SpeedZero	DWORD	Setpoint speed zero [Hz]
SpeedLimit	DWORD	Setpoint speed limit [Hz]

4.6.6 User-defined Data Type TS_DRIVE_ACTUALVALUES

The user-defined data type TS_DRIVE_ACTUALVALUES contains the data structure of the prepared actual values of the drive.

Name	Type	Meaning
Status	WORD	Device status of the drive
StatusWord	TS_DRIVE_STATUSWORD	Status word of the drive
Act_Position	DINT	Actual position
Act_Velocity	INT	Actual velocity
Act_Current	INT	Actual current
StatusSC0	DWORD	Status of safety control function SC0
StatusSC1	DWORD	Status of safety control function SC1
SpeedLimitSC0	DWORD	Current setpoint "speed limit" in SC0
SpeedLimitSC1	DWORD	Current setpoint "speed limit" in SC1

4.6.7 User-defined Data Type TS_OBJECTDATA_SC

The user-defined data type TS_OBJECTDATA_SC describes the structure of the transmitted and received data of the safety objects.

Name	Type	Meaning
Data	DWORD	Value for the functionality
Address	BYTE	Drive number with number of the safety control function
Index	BYTE	Subindex for the functionality with read or write function
Pattern	BYTE	Check pattern
Checksum	BYTE	Check sum
Value0	DINT	Return value from object access
Value1	DINT	Return value from object access

4.6.8 User-defined Data Type TS_SC_PROCESSDATA

The user-defined data type TS_SC_PROCESSDATA describes the structure of the access control to the safety objects.

Name	Type	Meaning
Handle	TS_HANDLE_DATA	Structure for control of the function block
In_SC_Obj	TS_OBJECTDATA_SC	Object data received from SC
Out_SC_Obj	TS_OBJECTDATA_SC	Object data to be sent to SC

4.6.9 User-defined Data Type TS_OBJECTDATA

The user-defined data type TS_OBJECTDATA describes the structure of the input and output data of the function blocks FB_WriteObject, FB_ReadObject and FB_SetArrayIndex.

Name	Type	Meaning
Handle	TS_HANDLE_DATA	Structure for control of the function block
ObjectIndex	INT	Object index
ObjectSubindex	DINT	Object subindex
ObjectValue	DINT	Set value for the object
ObjectNoOfBytes	INT	Number of valid bytes (1..4)
ArrayIndex	DINT	Set value of array index, starts at 0 or 4
ReturnValue	DINT	Error code (2 bytes) or return value (4 bytes)

4.6.10 User-defined Data Type TS_HANDLE_DATA

The user-defined data type TS_HANDLE_DATA describes the structure for the sequential control of a function block.

Name	Type	Meaning
Execute	BOOL	TRUE = enable function execution FALSE = reset function sequence
Busy	BOOL	TRUE = function execution is active
Done	BOOL	TRUE = function successfully executed
Error	BOOL	TRUE = function terminated due to error
ErrorID	INT	Error code

Name	Type	Meaning
Step	INT	Number in sequence

4.7 Checksum Calculation

A checksum is calculated to check the data of the safety objects. For this purpose the individual elements of the structure are written into a byte array. Using this byte array the data are added byte-by-byte and subtracted from the value 0xFF.

$$Check = 0xFF - \sum_{i=0}^{6} Byte\ i$$

Example: Checksum calculation

Reset error in SC1 for drive B, Pattern = AA_h

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Sum	Checksum = 0xFF – sum
0x01	0x00	0x00	0x00	0x81	0x83	0xAA	0xAF	0xFF – 0xAF = **0x50**

For the data to SD2x the checksum is checked in the function block FB_SC_FillObject. For the data from SD2x the checksum is checked in the function block FB_SC_CheckObject.

4.8 System Functions and System Function Blocks

The system function block TON is used for time monitoring.

The blocks of the TIA system functions are described in the according manuals by Siemens.

5 Hardware Configuration

The example project contains the following hardware configuration:

- SPS S7-1500
- PROFINET IO controller – gateway – SD2S
- PROFIBUS DP master – gateway – SD2S

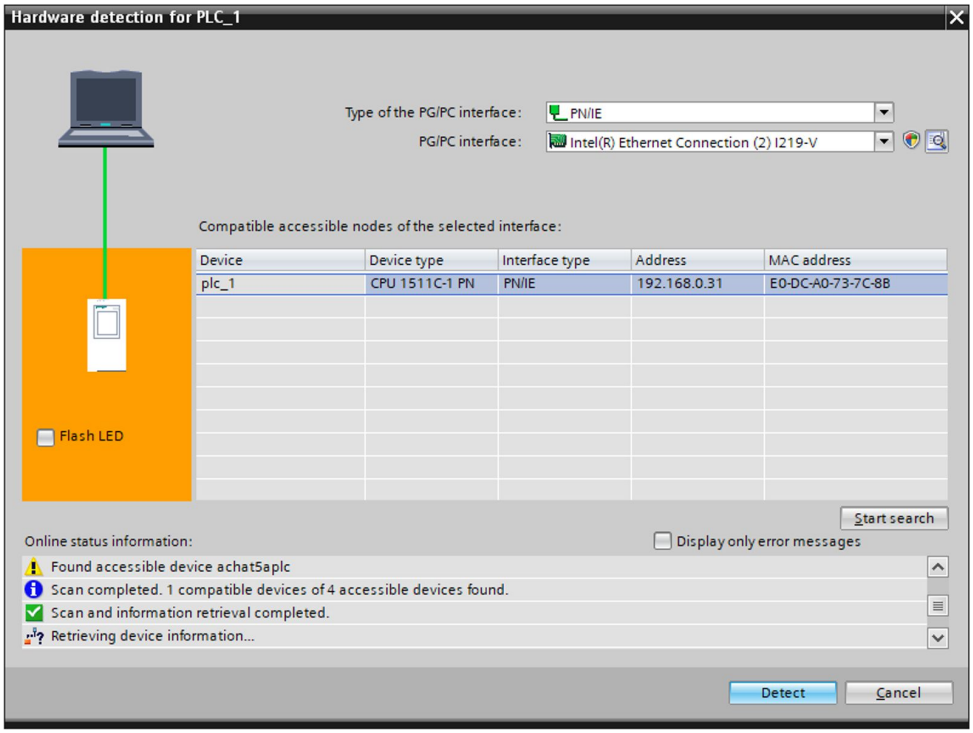
If you require another hardware configuration, proceed as described in the following in TIA Portal:

1. Select the CPU under the entries “Devices & networks” and “Add new device”. (Example: Controllers → Simatic S7-1500 → CPU → Unspecified CPU 1500 → 6ES7 5XX-XXXXX-XXXX → Add)
2. To integrate the CPU, the device data and the gateways proceed as described in the following sections.

5.1 Integrate CPU

The unspecified CPU is displayed.

1. Click “Detect” to open the following search window and detect the configuration of the connected devices.



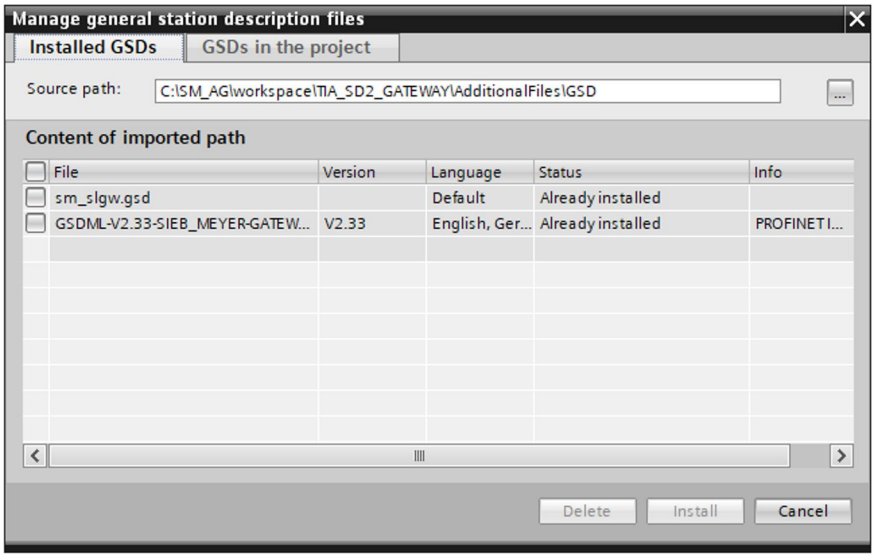
2. Set the interface via the selection lists in the top and click “Start search”. All accessible participants are displayed.
3. Select the desired CPU in the device list and click “Detect”.

✓ Then, the CPU and its components are displayed in the device view.

5.2

You can download the latest GSD files from the SIEB & MEYER website. To integrate the GSD files proceed as follows:

1. Copy the ZIP files into a separate directory and unzip the files.
2. In the menu “Extras” select the point “Manage general station description files (GSD)”. The following selection window appears.



3. In the top, select the source path, in which the GSD files are saved.
4. Select the required files via the check boxes on the left side and click “Install”.

✓ The GSD files are integrated in the project now.

5.3 Integrate a PROFINET IO Gateway 0362157(A)

To integrate a PROFINET IO gateways 0362157(A) proceed as follows:

Note

In TIA Portal V15.1 you cannot add the gateway via the menu “Online → Hardware detection → PROFINET device from network…”. The following error message appears:

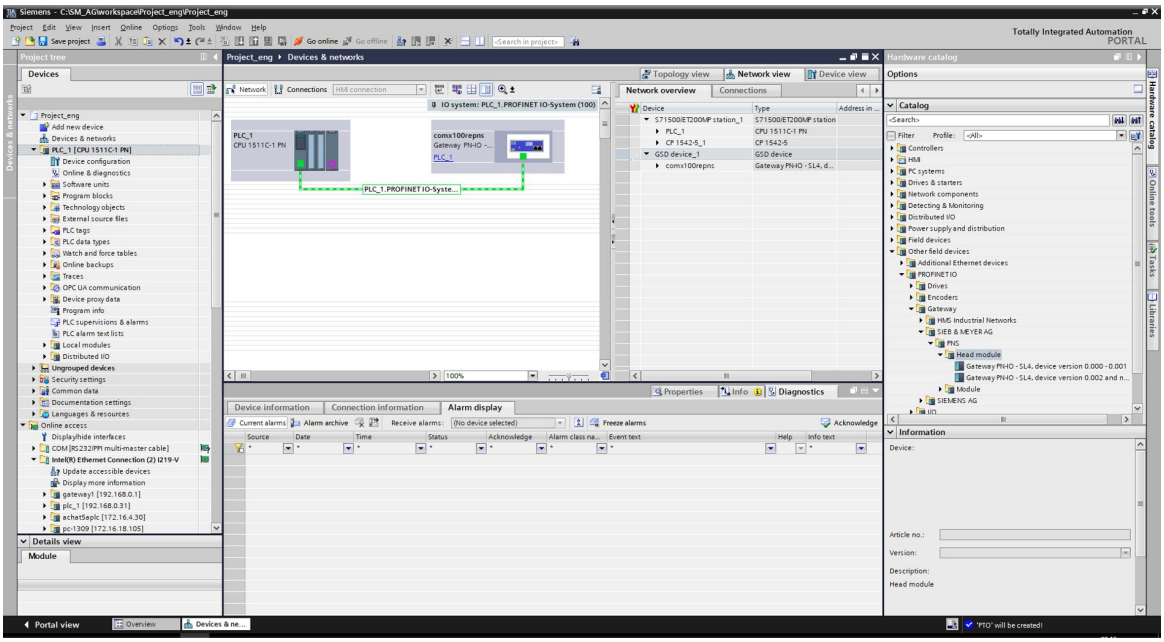
The IO device "device name" could not be added to the project because the article number of the online device could not be identified.

Since the gateway is not a product by Siemens and therefore has no Siemens article number, TIA Portal cannot identify and add the device. This convenience function is only implemented for Siemens devices. Other device must be integrated manually via the hardware catalog.

1. Switch to the network view.
2. Select the suitable gateway in the hardware catalog: Other field devices → PROFINET IO → Gateway → SIEB & MEYER AG → PNS → Gateway PNS-SL4, device version 0.002 and higher.
3. Select the device and drag it into the network window.

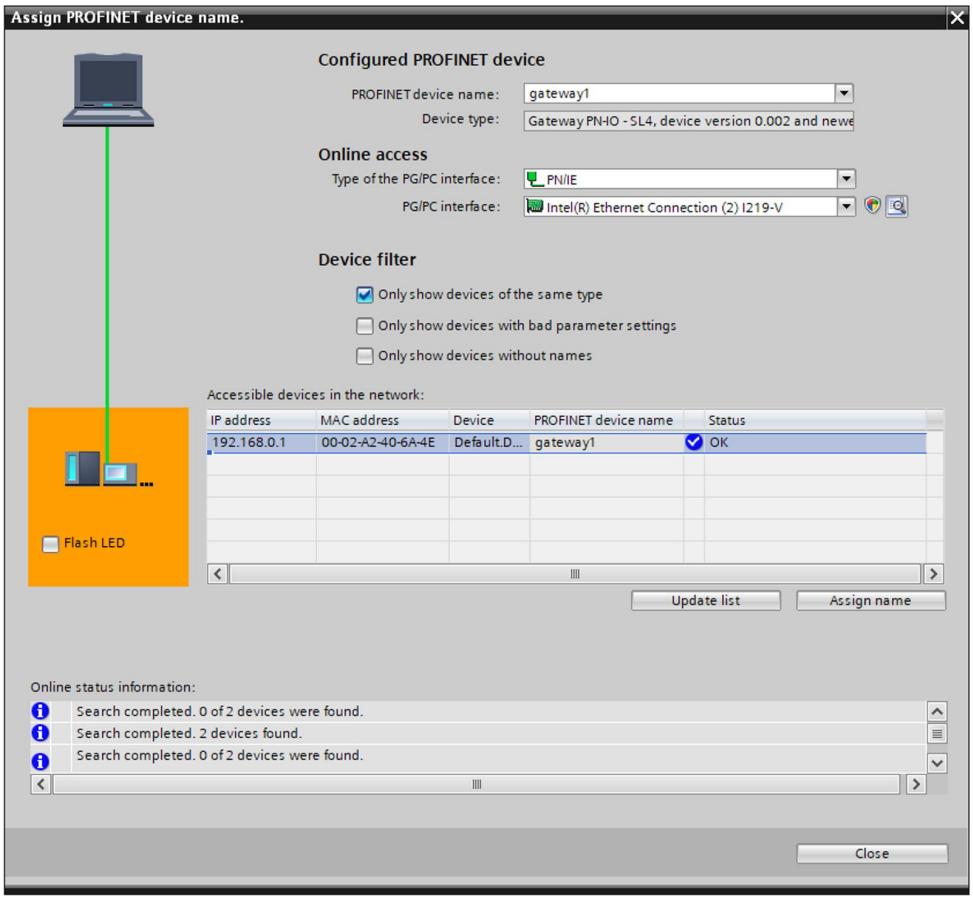
4. Right-click on the text “Not assigned” and select the entry “Assign to new IO controller” from the context menu. The following window appears:

5. Select the PROFINET interface in the window and click “OK”. The gateway is connected with the IO controller, now.



6. Switch to the device view.
7. To rename the gateway click on the module name “comx100repns” and enter the desired name.

8. Then, assign this device name to the gateway via the icon “Name”. The window “Assign PROFINET device name” appears.



9. Select the servo drives in the hardware catalog: Module → Input/output modules → Servo drive 16 byte input/output. Repeat this step for each servo drive. With a double servo drive you must select two drives.
10. Click “Update list”. All accessible devices are displayed.
11. Select the gateway and click “Assign name”. Close the window afterwards.

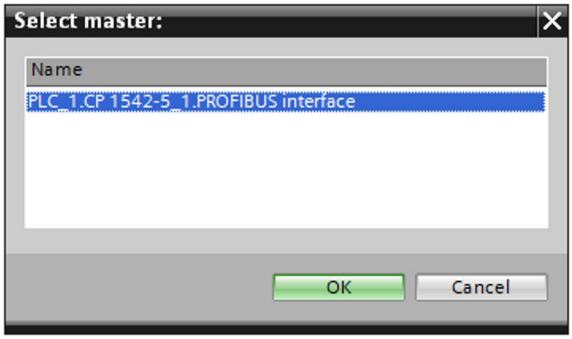
✓ Now, the gateway is integrated in the project including all its servo drives. For additional gateways, you must repeat this procedure for each gateway.

5.4 Integrate a PROFIBUS DP Gateway 0362151(A)

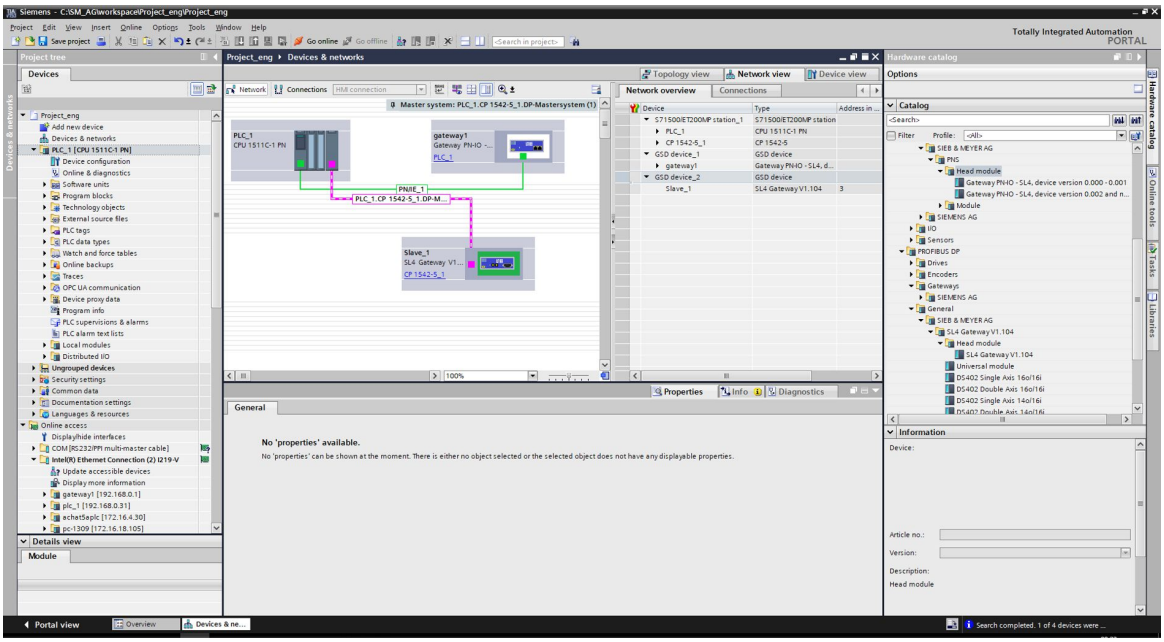
To integrate a PROFIBUS DP gateway 0362151(A) proceed as follows:

1. Switch to the network view.
2. Select the suitable gateway in the hardware catalog: Other field devices → PROFIBUS DP → General → SIEB & MEYER AG → SL4 gateway V1.104 → SL4 gateway V1.104.
3. Select the device and drag it into the network window.

4. Right-click on the text “Not assigned” and select the entry “Assign to new master” from the context menu. The following window appears:



5. Select the PROFIBUS interface in the window and click “OK”. The gateway is connected with the master, now.



6. Switch to the device view.
7. To rename the gateway click on the module name “Slave_1” and enter the desired name.
8. Select the servo drives in the hardware catalog: DS402 Single Axis 16o/16i or DS402 Double Axis 16o/16i. Repeat this step for each servo drive.

✓ Now, the gateway is integrated in the project including all its servo drives. For additional gateways, you must repeat this procedure for each gateway.

6 Loading the Program Example

6.1 Compile project

Before you can load the program example, it must be compiled at first:

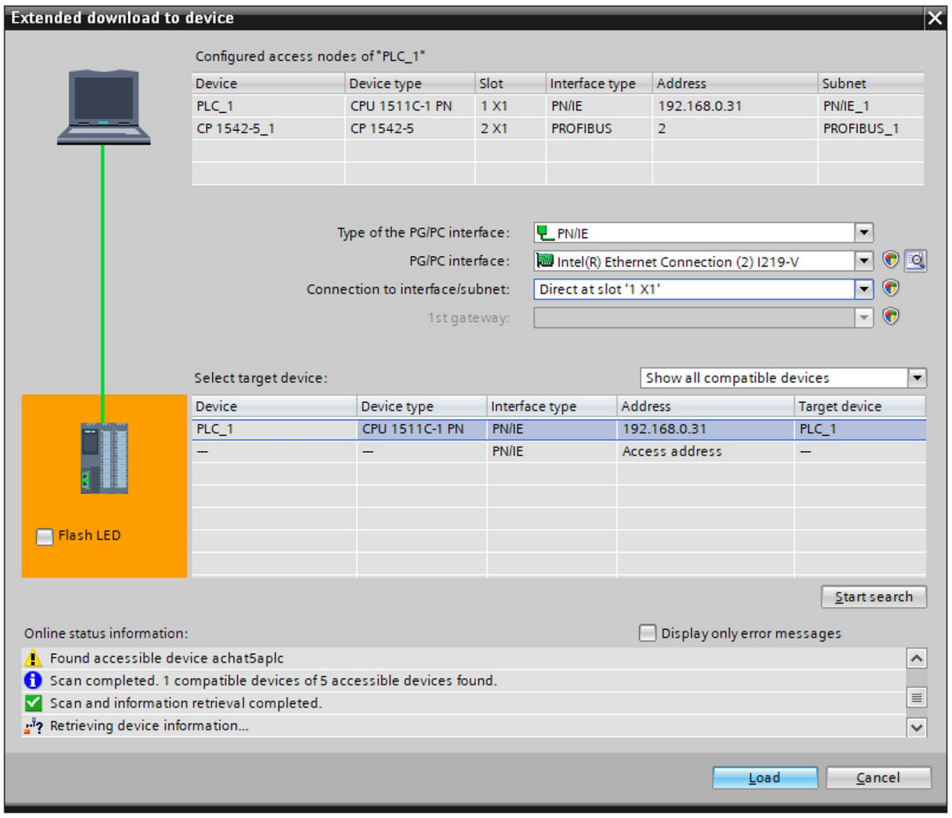
1. Select the CPU “PLC_1 [CPU 1511C-1 PN]” in the project tree.
2. Click on the icon for “Compile” in the tool bar.

✓ Now, the project is compiled. The results are displayed in the message window.

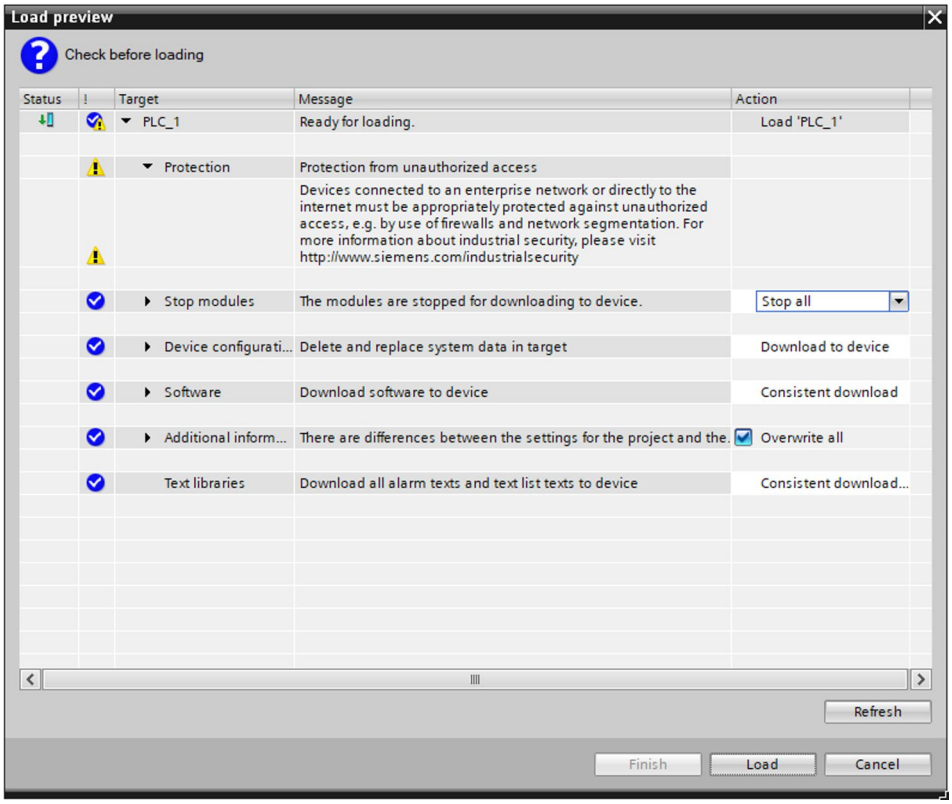
6.2 Load program example

Now, you can load the program example into the device as described in the following.

1. Click on the icon for “Download to device”. The window “Extended download to device” appears.
2. Select your PG interface and click “Start search”. After the search run the accessible target devices are displayed.



3. Select the desired PLC and click “Load”. The window “Load preview” appears.



4. Activate the listed actions as required (depending on the status of the PLC). Then, click “Load”. The loading progress is displayed.
5. After loading, click “Finish”.

✓ Now, the program example is loaded. The entire connection chain must have the status “OK”.

7 Error Detection

In the following several errors regarding the conversion from fieldbus to .

7.1 Error in S7 PLC

PLC failure

The gateway detects a bus error and sets the output data to SERVOLINK 4 to zero. The drive SD2x switches off because the control word has the value zero.

The gateway signals the error E16.

PLC stops

The S7 PLC sets the output data to zero. The gateway forwards this data to SERVOLINK 4. The drive SD2x switches off because the control word has the value zero.

The gateway signals the error E10.

Exchange of fieldbus data

During each program run the process data must be exchanged with the fieldbus data. If there is no data exchange, the former data pattern is kept and sent again and again. The gateway forwards these data via SERVOLINK 4. The drive is not able to distinguish whether the same data should be sent repeatedly or the data exchange is not called.

7.2 Error in Drive Control SD2x

In the event of a failure of the drive SD2x, the SERVOLINK 4 telegram is answered faulty or not at all.

Thus the gateway detects an error. The device distinguishes between a general error in SERVOLINK 4 and a slot error of an individual participant. If a general error is detected, the output data to the field bus are set to zero. If a slot error is detected, the output data are set to 0FFh. The status in the PLC program can be set correspondingly and the PLC program can be set to switch to an error routine.

8 SERVOLINK 4 gateway Test Assembly

The test assembly is done as shown in the hardware description. When the system is connected and ready for operation you can start checking.

8.1 S7 PLC Check

CPU

The front panel provides the following LEDs at the top, which indicate the states of the CPU and the PROFINET IO controllers:

- RUN/STOP LED (green/yellow)
- ERROR LED (red)
- MAINT LED (yellow)

During normal operation, only the RUN/STOP LED lights green. In case of a bus error, the ERROR LED additionally flashes red. For more information on the LED signals refer to the PLC manual.

In order to fix errors, check that the gateway is connected correctly. Check also whether the right number of drives is connected.

PROFIBUS DP master

This communication module comes with own LEDs that indicate the current status:

- RUN/STOP LED (green/yellow)
- ERROR LED (red)
- MAINT LED (yellow)

During normal operation, only the RUN/STOP LED lights green. In case of a bus error, the ERROR LED additionally flashes red. For more information on the LED signals refer to the PLC manual.

In order to fix errors, check that the gateway is connected correctly. Check also whether the right number of drives is connected.

8.2 Drive Check

You can check the received SERVOLINK 4 data using the bus monitor in the software *drivemaster2*.

8.2.1 Motor Control

When a motor is connected to the drive, it can be controlled via the inputs of the PLC.

- Via the input I_SwOn_0 the output stage is switched on. The drive switches to the state “switched on”.
- Via the input I_EnOp_0 the operation is enabled and the motor can rotate. The drive switches to the state “operation enabled”.
- Via the input I_QStop_0 the quick stop function is triggered. If the input is not active, the drive switches to the state “quick stop”.
- Via the input I_ResFault_0 a drive error is reset in the state “fault”. Then the drive switches to the state “ready for switch-on”.

- The input I_ResFault_SC resets an error in the safety function SFM or SLOF.

The inputs of I_Speed_0 allow setting the speed setpoints via the values 0 to 10. This corresponds to 0 to 100 % of the maximum motor speed. The demo program multiplies the set value with 1638 and applies the result as speed setpoint. The current setpoint has the fix value 100 % in the demo program.

The maximum values (= 100 %) for speed and current are 16380_{10} or $3FFC_{16}$.

Inputs	Percentage value [%]	Ref. value	
		Decimal value	Hexadecimal value
0	0	0	0x0000
1	10	1638	0x0666
2	20	3276	0x0CCC
3	30	4914	0x1332
4	40	6552	0x1998
5	50	8190	0x1FFE
6	60	9828	0x2664
7	70	11466	0x2CCA
8	80	13104	0x3330
9	90	14742	0x3996
10	100	16380	0x3FFC

Note

For additional drives you must connect the additional inputs accordingly.

9 Switching the Fieldbus System

If an existing project is to be switched to another fieldbus system, the software part can remain unchanged. Only the hardware configuration and the links to the variables must be adapted.

In the according gateway manual you find instructions on the connection of the gateway configuration to the PLC software exemplary for the TwinCAT 3 software. For other PLC system you can use this description as general basis.

10 Related Documents

The following documents provide more information on this topic:

Supplier	Document name
SIEB & MEYER AG	Gateway 0362151 – PROFIBUS to SERVOLINK 4 Connection
	Gateway 0362151A – PROFIBUS to SERVOLINK 4 and Safety
	Gateway 0362157 – PROFINET IO to SERVOLINK 4 Connection
	Gateway 0362157A – PROFINET IO to SERVOLINK 4 and Safety
	Drive System SD2 – Safety Functions SFM / SLOF
	Drive System SD2 – SERVOLINK 4 Bus System Description
	Drive System SD2 - Device Control
Siemens AG	profinet_step7_v15_function_manual_en-US_en-US.pdf
	81318674_Programming_guideline_DOKU_v13_en.pdf
	SCE_EN_012-100 Unspecific Hardware Configuration S7-1500_R1703.pdf
	Summary_SCE_Training_Curriculum_S7-1500_EN.pdf