SIEMENS

Drive ES SIMATIC V5.4
Function Block Library DRVDPS7

Function Description **Edition 04/06**

Safety Instructions

This Manual contains information which you should carefully observe to ensure your own personal safety and the prevention of damage to the system. This information is highlighted by a warning triangle and presented in one of the following ways depending on the degree of risk involved:

DANGER

indicates that death, severe personal injury or substantial property damage **will** result if proper precautions are not taken.

WARNING

indicates that death, severe personal injury or substantial property damage **can** result if proper precautions are not taken.

CAUTION

indicates that minor personal injury or property damage can result if proper precautions are not taken.

NOTE

contains important information about the product, its operation or a part of the document to which special attention is drawn.

Qualified person

The unit may only be started up and operated by a **qualified person or persons**. Qualified persons as referred to in the safety guidelines in this manual are those who are authorized to start up, earth and label units, systems and circuits in accordance with relevant safety standards.

Rules for proper use

Please note the following:

WARNING

The unit may be used only for the applications described in the catalog or the technical description, and only in combination with the equipment, components and devices of other manufacturers as far as this is recommended or permitted by Siemens.

The successful and safe operation of this equipment is dependent on proper transportation, storage, erection and installation and on careful operation and maintenance.

Trademarks

SIMATIC®, SIMATIC HMI®, SIMATIC NET®, SIROTEC®, SINUMERIK® and USS® are registered trademarks of Siemens AG. Other names in this publication might be trademarks whose use by a third party for his own purposes may violate the rights of the registered holder.

Siemens AG
Bereich Automatisierungs- und Antriebstechnik (A&D)
Geschäftsgebiet Motion Control Systems (MC)
Postfach 3180, D-91050 Erlangen

Table of Contents

1 Introduction

1.1 General

"Drive ES SIMATIC" supports the connection of variable-speed drive systems SIMOREG, SIMOVERT, SIMODRIVE and SINAMICS to a higher-level SIMATIC S7 control system. This link is made via the standardized communications system PROFIBUS DP in accordance with the "PROFIBUS Profile Drive Technology" or, alternatively for SIMOREG and SIMOVERT, via the Universal Serial Interface (USS®) Protocol.

This Manual describes the required STEP 7 user program for profile-compliant user data exchange between a "master" SIMATIC S7-300 or S7-400 PLC and "slave" drives on the **PROFIBUS DP** bus system.

The associated software is an integral component of the "Drive ES SIMATIC" product and stored in the STEP 7 library "DRVDPS7". Examples of configuration are given in the STEP 7 project example, "ZXy51_03_DriveES_SAMP".

1.2 Preconditions

Hardware

- PC/PG with Pentium 4, 600 MHz processor and 256 MB main memory (512 MB are recommended).
- S7-CPU 3xx/4xx with at least 48 KB user memory and integrated PROFIBUS DP interface or S7-CPU 4xx with at least 48 KB user memory and CP443-5

Software

- Operating system MS Windows 2000 Professional + SP3 / SP4 or MS Windows XP Professional + SP1 / SP1a /SP2
- STEP 7, version 5.3 or later
- Library Drive ES SIMATIC-DRVDPS7
- Tool "Drive ES - Generate DRIVDBx"

1.3 Installation

To start the installation, please insert the Drive ES SIMATIC-CD into the CD-ROM drive of your PG/PC and start the "install.bat" routine. All further instructions will be displayed during the installation process. Please also follow the instructions in the readme file.

1.4 Block functionality

The blocks stored in library DRVDPS7 offer the following functionality:

- Send control words and setpoints of selectable length (1 to 16 words) to the drive
- Receive status words and actual values with selectable length (1 to 16 words) from the drive
- Parameterize a drive in cyclic data exchange (one double word per job max.)
- Parameterize a drive in acyclic data exchange (up to 117 words per job)
- Read out a fault buffer of a SIMOVERT MASTERDRIVES, SIMOREG DC-MASTER, MICROMASTER 4xx or SINAMICS unit
- Up-/Download drive parameters which are stored in a job data block.
- Signal a communications error (slave failed or deactivated).

The blocks require a data block containing the configuration data of the drive slaves. This DB is easy to create with the tool "Drive ES - Generate DRIVDBx" (see Subsection 3.4.2).

2 Configuring the communication link

Before starting up the PROFIBUS communication link, you must configure the following:

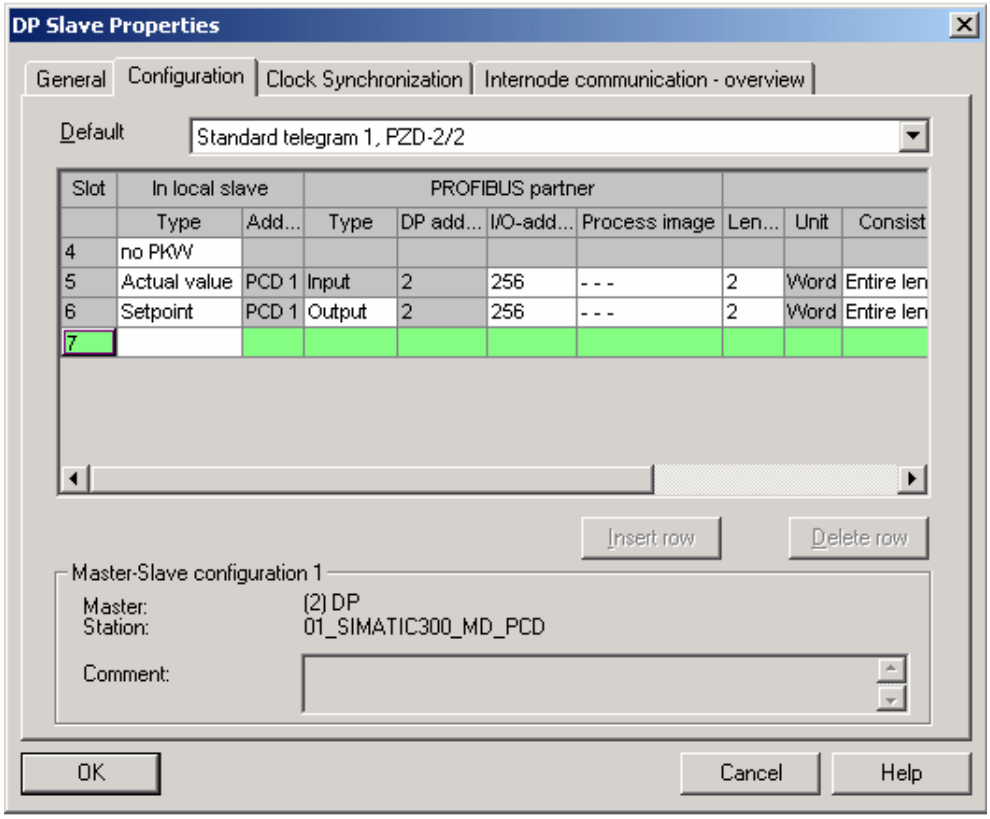
- Parameterize the DP interface in the SIMATIC
- Create a communications program
- Parameterize the drives.

2.1 Parameterizing the DP interface in the SIMATIC

The DP interface is parameterized as part of the hardware configuring process in STEP 7.

When you select a CPU with integrated DP interface or a DP communications processor from the hardware catalog of STEP 7, a PROFIBUS DP system is made available in the hardware configuration. Once you have assigned the parameters for the master (e.g. baud rate), you must select the slaves from the hardware catalog and position them in the PROFIBUS chain.

Installing Drive ES SIMATIC sets up a "Drive ES" profile in the hardware catalog. A station number and useful data structure (telegram types or slots with settings for type, length, consistency, I/O address (basic I/O address of slot)), including the partial process image assignment if appropriate, must be parameterized for the slaves which are available in this profile:



As an alternative, you can configure the drives as a standard slave with standard functionality. In this case, however, not all the functions listed in Section 1.4 will be available.

2.2 Creating a communications program

You can create a communications program using the STEP 7 programming tool (LAD, FPD, STL, SCL, CFC) by calling and parameterizing individual functions (function blocks) or, more practically, by using a block envelope (call block for individual functions per drive).

In either case, you must create data block DRIVDBx first and preset it to the configuration data of all drives from the STEP 7 hardware configuration (see Section 3.4 "Data blocks").

2.2.1 Sequence of steps when calling and parameterizing individual blocks

- Copy all blocks from library DRVDPS7 into the current project (including UDTs!).
- Create data block DRIVDBx with the configuration data of the drive slaves according to the HW configuration using tool "Drive ES - Generate DRIVDBx" (see Subsection 3.4.2).
- Call and parameterize standard blocks FB31 to FB42 and FC60 in the user program depending on the functionality you require (e. g. OB1) (parameter CFG_DATA: Reference to the corresponding SLOT_UDT in DRIVDBx).

NOTES:

You will require the following FB calls to transfer the data configured in HW Config of STEP 7:

FB31 per setpoint slot with own instance

FB32 per actual value slot with own instance

FB31 and FB32 1x each for a combined setpoint/actual value slot with own instance

FB33 for a PIV slot with own instance.

- Interconnect send and receive mailboxes in the instance data blocks with the control program.
- Load program to CPU.

2.2.2 Sequence of steps when using a block envelope

Advantages of using a block envelope:

- ❑ All necessary block calls for a drive are united in one block => improved program structure
- ❑ The multi-instances called in an FB (e.g. FB31–FB40) do not need their own instance DBs => Number of DBs reduced

- ➢ Copy all blocks from library DRVDPS7 into the current project (including UDTs!).
- ➢ Create data block DRIVDBx with the configuration data of the drive slaves according to the HW configuration using tool "Drive ES - Generate DRIVDBx" (see Subsection 3.4.2).
- ➢ Call and parameterize standard blocks FB31 to FB42 required for each drive as a multi-instance and FC60 in a block envelope (see also online help for "Multi-instance" in the SIMATIC Manager).
- ➢ If you need to use a standard block frequently (because, for example, several actual value slots are configured), then it must be integrated the corresponding number of times into the variable declaration.

NOTES:

You will require the following FB calls to transfer the data configured in HW Config of STEP 7:

FB31 per setpoint slot with own instance

FB32 per actual value slot with own instance

FB31 and FB32 1x each for a combined setpoint/actual value slot with own instance

FB33 for a PIV slot with own instance.

- ➢ Interconnect send and receive mailboxes in the multi-instance DB with the control program.
- ➢ Load program to CPU.

2.3 Parameterizing the drives

The drives are linked to the PROFIBUS DP via supplementary boards, interface modules or integrated PROFIBUS interfaces in the drives. These are

- the CB1, CBP or CBP2 communications board for MASTERDRIVES FC, VC with CU2 and SC,
- the CBP or CBP2 communications board for MASTERDRIVES Motion Control, Vector Control with CUVC and SIMOREG DC-MASTER,
- the CB24 communications board for SIMOREG K 6RA24,
- the OPMP interface module for MICRO-/MIDIMASTER 6SE31,
- the CB15 interface module for MICRO-/MIDIMASTER 6SE32,
- the CB155 interface module for COMBIMASTER,
- the interface module for MICROMASTER 4xx,
- the PROFIBUS options module for SIMODRIVE 611U,
- the integrated PROFIBUS interface for POSMO S/C and
- the integrated PROFIBUS interface for SINAMICS.

For information about parameterizing these interfaces in the devices, please refer to the operating instructions for the relevant board or device.

3 Description of blocks

3.1 Areas of application

The following blocks can be used both for PROFIBUS DP and for PROFINET IO.

3.1.1 Which block for which device

Usable blocks / Types of device	FB31 PCD_SEND	FB32 PCD_RECV	FB33 PDAT_CY	FB34 PDAT_AC	FB35 DEV_FLT1	FB36 PDAT_AC2	FB37 DEV_FLT2	FB38 DEV_FLT3	FB39 DEV_FLT4	FB40 PDAT_DL	FB41 PDAT_UD	FB42 PDAT_UD2	FC60 COM_STAT
MASTERDRIVES VC	Yes	Yes	Yes	Yes	Yes	Yes[1*)]	No	No	No	Yes	Yes	Yes[2*)]	Yes
MASTERDRIVES MC	Yes	Yes	Yes	Yes	Yes	Yes[1*)]	No	No	No	Yes	Yes	Yes[2*)]	Yes
MICROMASTER 3. Gen.	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	No	Yes
MICROMASTER 4. Gen.	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	Yes	Yes
SIMODRIVE 611U	Yes	Yes	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes
POSMO CA/CD/SI	Yes	Yes	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes
POSMO A	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	Yes
SIMOREG DC Master	Yes	Yes	Yes	Yes	No	Yes[1*)]	Yes	No	No	Yes	Yes	Yes[2*)]	Yes
SINAMICS G/S	Yes	Yes	No	No	No	Yes	No	No	Yes	No	No	Yes	Yes

1*) only with a CBP2 with FW version >= V2.20 and later

2*) only with a CBP2 with FW version >= V2.20 and later, and only for DBs created manually, i.e. not generated by Drive ES

Recommended combination

3.1.2 Which block for which application

Block	Typical application
FB31 PCD_SEND	Sending process data (control words and setpoints) to the drive
FB32 PCD_RECV	Receiving process data (status words and actual values) from the drive
FB33 PDAT_CY	Reading or writing parameters via the four-word wide PIV channel in the cyclical telegram. This function should especially be used when few parameters (< 5) are to be exchanged with as many drives as possible at the same time. This method of transferring parameters is only possible if PPO types have been selected as the telegram and it is no longer available with new drives such as SINAMICS.
FB34 PDAT_AC	Reading or writing parameters by means of 244-byte wide acyclical telegrams, using the Siemens proprietary data record DS100. This function should especially be used when ➢ an individual, indexed parameter with many indices is to be replaced ➢ administration for automatic processing of several parameters is not needed.
FB36 PDAT_AC2	Reading or writing parameters by means of 244-byte wide acyclical telegrams, using the DS47 data record standardized in PROFIdrive profile V3. This function should especially be used when ➢ a single, indexed parameter with many indices is to be replaced ➢ administration for automatic processing of several parameters is not needed.

Block	**Typical application**
FB35 DEV_FLT1	This block has been specially programmed for reading out the complete diagnostic buffer of a MASTERDRIVES drive. This block reads out the fault number, the fault value, the fault text and the time of the fault from the drive.
FB37 DEV_FLT2	This block has been specially programmed for reading out the complete diagnostic buffer of a SIMOREG DC Master drive. This block reads out the fault number, the fault value, the fault text and the time of the fault from the drive.
FB38 DEV_FLT3	This block has been specially programmed for reading out the complete diagnostic buffer of a MICROMASTER 4xx drive. This block reads out the fault number and the fault value from the drive.
FB39 DEV_FLT4	This block has been specially programmed for reading out the complete diagnostic buffer of a SINAMICS drive. This block reads out the fault number and the fault value from the drive.
FB40 PDAT_DL	This block processes automatic DOWNLOAD of one or even several data blocks from the CPU to the drive using the Siemens proprietary data record DS100. This block should be used when many parameters, up to a whole parameter set, are to be transferred automatically to the drive. ➢ e.g. for reparameterizing a drive after replacement of equipment ➢ e.g. for changing the parameters of a drive due a product, recipe or batch change.
FB41 PDAT_UD	This block processes automatic DOWNLOAD or UPLOAD (update) of one or even several data blocks using the Siemens proprietary data record DS100. This block should be used when ➢ many parameters, up to a complete parameter set, are to be transferred to the drive automatically - for reparameterizing a drive after replacement of equipment - for changing the parameters of a drive due to a product, recipe or batch change ➢ the parameter values that are stored in a DB are to be updated with the latest values from the drive - after parameter changes locally at the drive - to provide additional display values on HMI systems, apart from the available PCD data
FB42 PDAT_UD2	This block processes automatic DOWNLOAD or UPLOAD (update) of one or even several data blocks using the DS47 data record standardized in PROFIdrive profile V3. This block should be used when ➢ many parameters, up to a complete parameter set, are to be transferred to the drive automatically - for reparameterizing a drive after replacement of equipment - for changing the parameters of a drive due to a product, recipe or batch change ➢ the parameter values that are stored in a DB are to be updated with the latest values from the drive - after parameter changes locally at the drive - to provide additional display values on HMI systems, apart from the available PCD data
FC60 COM_STAT	

3.2 Function blocks

3.2.1 FB PCD_SEND: Write process data

FB 31

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task:	OB1
Cyclic interrupt OB:	e.g. OB32

Description of function

Adhering to the relevant consistency conditions, the block transfers the process data (control words, setpoints) cyclically from the SIMATIC to the drive.

If several setpoint slots are configured for the data exchange between the SIMATIC and drive, the FB must be called once per slot with its own instance. In this case, however, it must be ensured that the process data of the relevant slot are correctly assigned to the block inputs (see example in Section 7.1).

Exactly one call must be made for a combined setpoint/actual value slot.

The FB must also be called once per axis and setpoint slot with its own instance for multi-axis drives.

PZD setpoint interface

You can freely specify the length of the PZD interface up to a total length of 16 words. The interface is parameterized in HW Config. For this purpose, it is possible to directly select the standard telegrams according to the PROFIdrive Profile Drive Technology, or PPO types 1 to 5.

The first word in the setpoint range (PCD_1) must always be occupied by the control word.

Communications interface

Parameter	Declaration	Data type	Memory area	Description
CFG_DATA	INPUT	SLOT_UDT (application-specific)	D, L	Slot-specific configuration data (format, see Section 3.4)
CFG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Slave or slot not configured or incorrectly configured
PCD_1	INPUT	WORD	I, Q, M, D, L, Const.	Control word
PCD_2	INPUT	WORD	I, Q, M, D, L, Const.	Main setpoint
PCD_3	INPUT	WORD	I, Q, M, D, L, Const.	Setpoint/additional control word
...				
PCD_16	INPUT	WORD	I, Q, M, D, L, Const.	Setpoint/additional control word
SFC_ERR	OUTPUT	BOOL	I, Q, M, D, L	SFC 15 DPWR_DAT signals error

Data area

Parameter	Declaration	Data type	Description
SFC_RET_VAL	STAT	INT	Return value of SFC15 DPWR_DAT

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Slot type not identical to setpoint or combined setpoint/actual value (i.e. parameter SLOT_ID <> 1 or 3) ➢ Logical basic address = 0 ➢ Data length = 0 ➢ Process data address = 0 or > (16 + 1 - data length)
SFC_ERR	SFC error with data transfer using system function SFC15 DPWR_DAT (return value < 0) The return value is stored in instance DB (SFC_RET_VAL).

NOTE:

The block signals CFG_ERR if PCD_SEND for a setpoint slot is called exclusively with internode communication data.

Block call (STL source code)

```
CALL       PCD_SEND, DB_PCD_SEND(
           CFG_DATA     := DRIVDB1.SLAVE_1.SLOT_6,
           PCD_1        := MW0,
           PCD_2        := MW2,
           PCD_3        := MW4,
           PCD_4        := MW6,
           PCD_5        := MW8,
           PCD_6        := MW10,
           PCD_7        := MW12,
           PCD_8        := MW14,
           PCD_9        := MW16,
           PCD_10       := MW18,
           PCD_12       := MW20,
           PCD_13       := MW22,
           PCD_14       := MW24,
           PCD_15       := MW26,
           PCD_16       := MW28,
           SFC_ERR      := M30.0,
           CFG_ERR      := M30.1);
```

3.2.2 FB PCD_RECV: Read process data

FB 32

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task: OB1

Cyclic interrupt OB: e.g. OB32

Description of function

Adhering to the relevant consistency conditions, the block transfers the process data (status words, actual values) cyclically from the drive to the SIMATIC.

If several actual value slots are configured for the data exchange between the drive and SIMATIC, the FB must be called once per slot with its own instance. In this case, it must be ensured that the process data of the relevant slot are correctly assigned to the block outputs (according to Section 7.1).

Exactly one call must be made for a combined setpoint/actual value slot.

The FB must also be called once per axis and actual value slot with its own instance for multi-axis drives.

PZD actual value interface

You can freely specify the length of the PZD interface up to a total length of 16 words. The interface is parameterized in HW Config. For this purpose, it is possible to directly select the standard telegrams according to the PROFIdrive Profile Drive Technology, or PPO types 1 to 5. To ensure correct operation of the block, it is absolute essential to allocate status word 1 of the drive (e.g. MASTERDRIVES P734.1 = 32) to the first word in the actual value area (PCD_1).

Communications interface

Parameter	Declaration	Data type	Memory area	Description
CFG_DATA	INPUT	SLOT_UDT (application-specific)	D, L	Slot-specific configuration data (format, see Section 3.4)
CFG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Slave or slot not configured or incorrectly configured
DEV_FLT	OUTPUT	BOOL	I, Q, M, D, L	Drive signals device fault
DEV_WAR	OUTPUT	BOOL	I, Q, M, D, L	Drive signals device alarm
PCD_1	OUTPUT	WORD	I, Q, M, D, L	Status word
PCD_2	OUTPUT	WORD	I, Q, M, D, L	Main actual value
PCD_3	OUTPUT	WORD	I, Q, M, D, L	Actual value/additional status word
...				
PCD_16	OUTPUT	WORD	I, Q, M, D, L	Actual value/additional status word
PLC_CTRL	OUTPUT	BOOL	I, Q, M, D, L	PLC control requested by slave
SFC_ERR	OUTPUT	BOOL	I, Q, M, D, L	SFC 14 DPRD_DAT signals error

Data area

Parameter	Declaration	Data type	Description
SFC_RET_VAL	STAT	INT	Return value of SFC14 DPRD_DAT

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Slot type does not match actual value or combined setpoint/actual value (i.e. parameter SLOT_ID <> 2 or 3) ➢ Logical basic address = 0 ➢ Data length = 0 ➢ Process data address = 0 or > (16 + 1 - data length)
DEV_FLT	DEV_FLT is set as a function of the drive fault bit in the status word. This bit can be used to activate function block DEV_FLT for the purpose of reading out the fault buffer.
DEV_WAR	DEV_WAR is set as a function of the drive alarm bit in the status word.
SFC_ERR	SFC error with data transfer using system function SFC DPRD_DAT (return value < 0) The return value is stored in instance DB (SFC_RET_VAL). The receive mailbox is deleted.

Block call (STL source code)

```
CALL       PCD_RECV, DB_PCD_RECV(
           CFG_DATA     := DRIVDB1.SLAVE_1.SLOT_5,
           PCD_1        := MW0,
           PCD_2        := MW2,
           PCD_3        := MW4,
           PCD_4        := MW6,
           PCD_5        := MW8,
           PCD_6        := MW10,
           PCD_7        := MW12,
           PCD_8        := MW14,
           PCD_9        := MW16,
           PCD_10       := MW18,
           PCD_12       := MW20,
           PCD_13       := MW22,
           PCD_14       := MW24,
           PCD_15       := MW26,
           PCD_16       := MW28,
           PLC_CTRL     := M30.0,
           DEV_FLT      := M30.1,
           DEV_WAR      := M30.2,
           SFC_ERR      := M30.3,
           CFG_ERR      := M30.4);
```

3.2.3 FB PDAT_CY: Process parameters cyclically

FB 33

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task: OB1

Cyclic interrupt OB: e.g. OB32

Description of function

The block processes the optional, four-word wide PIV interface in the cyclical telegram. A PIV job entered in the interface is processed completely. A new PIV job is transferred when a positive edge appears at the START input of the block. The START bit should be reset by the BUSY signal. The next job can then be written to the transmit interface, but must not be started until checkback signal READY or xyz_ERR arrives.

The interface is parameterized in HW Config.

The FB must be called once per axis and PIV slot with its own instance for multi-axis drives.

Communications interface

Parameter	Declaration	Data type	Memory area	Description	
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Job in progress	
CFG_DATA	INPUT	SLOT_UDT (application-specific)	D, L	Slot-specific configuration data (format, see Section 3.4)	
CFG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Slave or slot not configured or incorrectly configured	
R_ID_IN	INPUT	BYTE	I, Q, M, D, L, Const.	Job identifier	Send mailbox
P_NO_IN[1)]	INPUT	WORD	I, Q, M, D, L, Const.	Parameter number	
P_IND_IN	INPUT	BYTE	I, Q, M, D, L, Const.	Parameter index	
P_VAL_IN	INPUT	DWORD	I, Q, M, D, L, Const.	Parameter value	
R_ID	OUTPUT	BYTE	I, Q, M, D, L	Response identifier	Receive mailbox
P_NO	OUTPUT	WORD	I, Q, M, D, L	Parameter number	
P_IND	OUTPUT	BYTE	I, Q, M, D, L	Parameter index	
P_VAL	OUTPUT	DWORD	I, Q, M, D, L	Parameter value	
READY	OUTPUT	BOOL	I, Q, M, D, L	Job finished without error	
REQ_ERR	OUTPUT	BOOL	I, Q, M, D, L	Response contains error identifier	
SFC_ERR	OUTPUT	BOOL	I, Q, M, D, L	SFC 14 DPRD_DAT/SFC 15 DPWR_DAT signals error	
START	INPUT	BOOL	I, Q, M, D, L	Accept start pulse for job	
WDOG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Watchdog error, no plausible response data available	

1) "Parameter page selection" (setting of Para_Page_Select - bits in index word) can be ignored when the parameter number (P_NO_IN) is specified. Parameter numbers from 0h to 7CFFh (0 to 31999) can be entered directly in P_NO_IN.

If a job is processed correctly, the signaling bits and data at the block outputs remain valid until the next job is initiated.

The xyz_ERR outputs are updated cyclically. If an SFC_ERR occurs, the READY output is set. When SFC_ERR has been eliminated, you must trigger the job again to obtain new, up-to-date status information.

Data area

Parameter	Declaration	Data type	Description
SFC14_RET_VAL	STAT	INT	Return value of SFC14 DPRD_DAT
SFC15_RET_VAL	STAT	INT	Return value of SFC15 DPWR_DAT
WDOG_CYCLE_NO	STAT	WORD	Number of cycles in wait period for arrival of plausible response data before a watchdog error is signaled (default = 100)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Slot type does not match PIV (i.e. parameter SLOT_ID <> 4) ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier = 8)
REQ_ERR	Job contains error identifier (response identifier = 7) P_VAL also contains a more detailed description of the error (see Section 7.2 "Formulating parameter jobs (data record 100)"
SFC_ERR	SFC error with data transfer using system functions SFC DPWR_DAT/DPRD_DAT (return value < 0) The return value is stored in instance DB (SFC14_RET_VAL or SFC15_RET_VAL). The meaning of the return value can be found in the online help for the SFC. The receive mailbox is cleared if an error in reading the data occurs.
WDOG_ERR	No plausible response data within monitoring period

NOTE:

Bit 11 in parameter 879 must be set when the 611U is parameterized if SIMODRIVE 611U parameters are to be processed via the cyclic PROFIBUS interface by means of PDAT_CY. An error in interpreting the subindex in the PIV task will otherwise occur.

Block call (STL source code)

```
NETWORK
TITLE =Check starting conditions of PDAT_CY

      U    "DI_PDAT_CY".SFC_ERR;   // Retrigger of START in case of SFC error
      UN   "DI_PDAT_CY".START;
      =    "DI_PDAT_CY".START;
      U    "DI_PDAT_CY".SFC_ERR;
      SPB  PDAT;

      UN   "DI_PDAT_CY".BUSY;      // Starting condition after complete
      UN   "DI_PDAT_CY".READY;     // download of the program
      UN   "DI_PDAT_CY".SFC_ERR;
      UN   "DI_PDAT_CY".WDOG_ERR;
      UN   "DI_PDAT_CY".CFG_ERR;
      UN   "DI_PDAT_CY".REQ_ERR;
      S    "DI_PDAT_CY".START;

      U    "DI_PDAT_CY".BUSY;      // Reset of START
      R    "DI_PDAT_CY".START;

      U    "DI_PDAT_CY".READY;     // Start new request
      S    "DI_PDAT_CY".START;

      U    "DI_PDAT_CY".REQ_ERR;   // Start new request or error routine
      S    "DI_PDAT_CY".START;

      U    "DI_PDAT_CY".WDOG_ERR;  // Start new request or error routine
      S    "DI_PDAT_CY".START;

      U    "DI_PDAT_CY".CFG_ERR;   // Start new request or error routine
      S    "DI_PDAT_CY".START;

NETWORK
TITLE =assembling cyclic requests

      UN   "DI_PDAT_CY".START;
      SPB  PDAT;

// assemble cyclic requests

NETWORK
TITLE =  Call for PDAT_CY

PDAT: CALL  PDAT_CY, DI_PDAT_CY(
            CFG_DATA   := DRIVDB1.SLAVE_1.SLOT_4,
            START      := ,
            R_ID_IN    := MB0,
            P_NO_IN    := MW2,
            P_IND_IN   := MB4,
            P_VAL_IN   := MD6,
            R_ID       := MB10,
            P_NO       := MW12,
            P_IND      := MB14,
            P_VAL      := MD16,
            BUSY       := ,
            READY      := ,
            REQ_ERR    := ,
            WDOG_ERR   := ,
            SFC_ERR    := ,
            CFG_ERR    := );

NETWORK
TITLE = interpret answer according to the containing data
```

3.2.4 FB PDAT_AC: Process parameters acyclically (DS100)

FB 34

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task: OB1

Cyclic interrupt OB: e.g. OB32

Description of function

The block coordinates acyclic communications between the drive and S7-CPU for the transmission of parameter data in accordance with the PIV mechanism specified in the USS protocol specification. Data exchange takes place via the S7 communication services "Read/write data record" (data record number100).

The PDAT_AC processes one **individual** parameter job per call. The PIV job is directly written to block inputs and the PIV response is available at block outputs. The values to be written must be stored in a data structure (PVAL_UDT). After job processing has been completed, the received values are available in such an UDT as well.

A new job is transferred when a positive edge appears at the START input of the block. The job is then executed exactly once. The START bit must be reset by the BUSY signal. The next job can then be written to the transmit interface, but must not be started until checkback signal READY or xyz_ERR arrives.

There is no coordination of jobs requested by different applications. This must be implemented within the applications if necessary. See Subsection 3.2.13 "Locking the blocks with acyclical communication"..

The block must be called once per axis with its own instance on multi-axis drives.

The block supports the following tasks:

- Read/write parameter value, simple (word/double word)
- Read/write parameter value, one or more array elements (117 max.)
- Read parameter description, completely or a single descriptive element
- Read parameter texts, individually or several text elements.

Communications interface

Parameter	Declaration	Data type	Memory area	Description	
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Job in progress	
CFG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Slave not configured or incorrectly configured	
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Diagnostic address of slave[1)]	
R_ID_IN	INPUT	BYTE	I, Q, M, D, L, Const.	Job identifier	Send mailbox
P_NO_IN[2)]	INPUT	WORD	I, Q, M, D, L, Const.	Parameter number	
P_IND_IN	INPUT	BYTE	I, Q, M, D, L, Const.	Parameter index	
P_VAL_IN	INPUT	PVAL_UDT	D, L	Parameter value	
R_ID	OUTPUT	BYTE	I, Q, M, D, L	Response identifier	Receive mailbox
P_NO	OUTPUT	WORD	I, Q, M, D, L	Parameter number	
P_IND	OUTPUT	BYTE	I, Q, M, D, L	Parameter index	
P_VAL	OUTPUT	PVAL_UDT	D, L	Parameter value	
READY	OUTPUT	BOOL	I, Q, M, D, L	Job finished without error	
REQ_ERR	OUTPUT	BOOL	I, Q, M, D, L	Response contains error identifier	
SFB_ERR	OUTPUT	BOOL	I, Q, M, D, L	SFB 53 WRREC / SFB 52 RDREC signals error	
START	INPUT	BOOL	I, Q, M, D, L	Accept start pulse for job	
WDOG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Watchdog error, no plausible response data available	

1) The I/O basic address of any slot (exception: PIV or empty slot) from the area of the axis to be addressed must be specified here for multi-axis drives according to DS 100. In the case of mixed configurations of single and multi-axis drives, it is advisable to use the I/O basic address for both single-axis and multi-axis drives according to DS 100.

2) "Parameter page selection" (setting of Para_Page_Select - bits in index word) can be ignored when the parameter number (P_NO_IN) is specified. Parameter numbers from 0h to 7CFFh (0 to 31999) can be entered directly in P_NO_IN.

If a job is processed correctly, the signaling bits and data at the block outputs remain valid until the next job is initiated.

Data type PVAL_UDT

The parameter values from both the send mailbox and the receive mailbox are of the user-defined data type PVAL_UDT.

PVAL_UDT: UDT 32 (data type for memory area "parameter values for acyclic communication")
The number of the UDT must **not** be changed.

Parameter	Data type	Description
DWORD_1	DWORD	1st double word of parameter values
DWORD_2	DWORD	2nd double word of parameter values
...		
DWORD_59	DWORD	59th double word of parameter values

Data area

Parameter	Declaration	Data type	Description
NODATA_CYCLE_NO	STAT	INT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
NODATA_RETRY_NO	STAT	BYTE	Number of job repetitions if no response data have been received (default = 5)
SFB52_RET_VAL	STAT	INT	Return value of SFB 52 RDREC
SFB53_RET_VAL	STAT	INT	Return value of SFB53 WRREC
WDOG_RETRY_NO	STAT	BYTE	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier = 8)
REQ_ERR	Job contains error identifier (response identifier = 7) P_VAL.DWORD_2 also contains a more detailed description of the error (see Section: 7.2 Formulating parameter jobs (data record 100)
SFB_ERR	SFB error with data transfer using system functions SFB RDREC/WRREC (return value < 0) The return value is stored in instance DB (SFB53_RET_VAL or SFB52_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14. The receive mailbox is cleared if an error in reading the data occurs.
WDOG_ERR	No plausible response data within monitoring period (default: After five job repetitions)

NOTE:

SFB_ERR is not signaled immediately on some CPUs of the S7-300 (old versions!) when the communication link is faulty and a job has been initiated. Instead, the block continues to signal BUSY until either the error is eliminated so that the job can be completed, or the cycle monitoring function (default: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been cancelled, you must restart your job.

Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.

Block call (STL source code)

```
NETWORK
TITLE =Check starting conditions of PDAT_AC

      U    "DI_PDAT_AC".SFB_ERR;   // Retrigger of START in case of SFB error
      UN   "DI_PDAT_AC".START;
      =    "DI_PDAT_AC".START;
      U    "DI_PDAT_AC".SFB_ERR;
      SPB  PDAT;

      UN   "DI_PDAT_AC".BUSY;      // Starting condition after complete
      UN   "DI_PDAT_AC".READY;     // download of the program
      UN   "DI_PDAT_AC".SFB_ERR;
      UN   "DI_PDAT_AC".WDOG_ERR;
      UN   "DI_PDAT_AC".CFG_ERR;
      UN   "DI_PDAT_AC".REQ_ERR;
      S    "DI_PDAT_AC".START;

      U    "DI_PDAT_AC".BUSY;      // Reset of START
      R    "DI_PDAT_AC".START;

      U    "DI_PDAT_AC".READY;     // Start new request
      S    "DI_PDAT_AC".START;

      U    "DI_PDAT_AC".REQ_ERR;   // Start new request or error routine
      S    "DI_PDAT_AC".START;

      U    "DI_PDAT_AC".WDOG_ERR;  // Start new request or error routine
      S    "DI_PDAT_AC".START;

      U    "DI_PDAT_AC".CFG _ERR;  // Start new request or error routine
      S    "DI_PDAT_AC".START;

NETWORK
TITLE =assembling acyclic requests

      UN   "DI_PDAT_AC".START;
      SPB  PDAT;                   // assemble acyclic requests
```

```
NETWORK
TITLE =  Call for PDAT_AC

PDAT: CALL  "PDAT_AC" , "DI_PDAT_AC" (
            LADDR     :=   "DRIVDB1".SLAVE_5.DADDR,
            START     :=   ,
            R_ID_IN   :=   MB0,
            P_NO_IN   :=   MW2,
            P_IND_IN  :=   MB4,
            P_VAL_IN  :=   MOTOR.SEND_BUFFER,  //Refers to data type PVAL_UDT
            R_ID      :=   MB5,
            P_NO      :=   MW6,
            P_IND     :=   MB8,
            P_VAL     :=   MOTOR.SEND_BUFFER,  //Refers to data type PVAL_UDT
            BUSY      :=   ,
            READY     :=   ,
            REQ_ERR   :=   ,
            WDOG_ERR  :=   ,
            SFB_ERR   :=   ,
            CFG_ERR   :=   );

NETWORK
TITLE = interpret answer according to the containing data
```

Multi-axis drive according to DS 100:

```
CALL        PDAT_AC, DI_PDAT_AC(
            LADDR          := DRIVDB1.SLAVE_1.SLOT_x.LADDR,
                                  // x=any I/O basic address
                                  // of a slot from the area
                                  // of the axis to be addressed
                                  // ≠PIV or empty slot
            START          := ,
            R_ID_IN        := MB0,
            P_NO_IN        := MW2,
            P_IND_IN       := MB4,
            P_VAL_IN       := MOTOR.SEND_BUFFER, //Refers to data type PVAL_UDT
            R_ID           := MB5,
            P_NO           := MW6
            P_IND          := MB8
            P_VAL          := MOTOR.SEND_BUFFER, //Refers to data type PVAL_UDT
            BUSY           := ,
            READY          := ,
            REQ_ERR        := ,
            WDOG_ERR       := ,
            SFB_ERR        := ,
            CFG_ERR        := );
```

3.2.5 FB PDAT_AC2: Process parameters acyclically (DS 47)

FB 36

The block number can be altered.

This block cannot be used on some series 300 CPUs on which the SFC24 (TEST_DB) is not installed.

Calling OBs

The block can be incorporated into one of the following OBs:

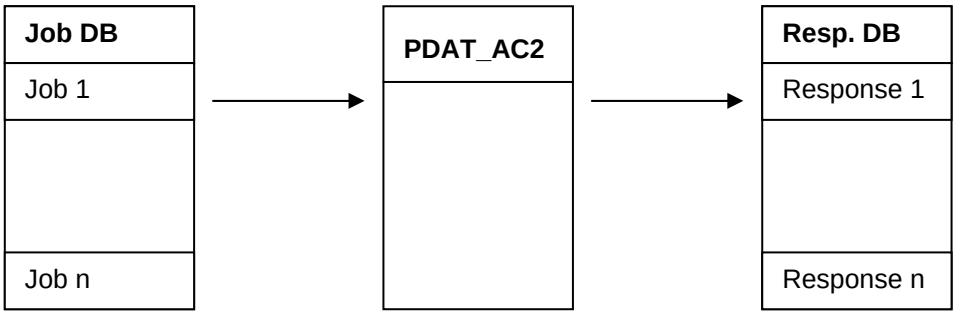
Cyclical task: OB1

Cyclic interrupt OB: e.g. OB32

Description of function

The block coordinates acyclic communications between the driver and the S7-CPU for the transmission of parameter data in accordance with the extended PROFIdrive Profile Drive Technology. Data exchange takes place via the S7 communications services "Read/write data record" (data record number 47).

The number of the data record to be used can be entered via input DS_NO. If the value 0 is assigned to the input, data record 47 is used. In the case of values <> 0, the value at the input is interpreted as the number of the data record to be used. However, the telegram must still be structured in accordance with PROFIdrive Profile Drive Technology, Version 3 as the block will otherwise report errors, i.e. any data record supported by the drive can be used provided the contents of the telegram conform to PROFIdrive Profile Drive Technology, Version 3. This function is relevant mainly for drives with a PROFInet interface.



The parameter jobs are stored in a job data block. Each job comprises at least one "Request header" and one "Parameter address". A write job also contains format information, the number of values to be written and a value field. A job always has a fixed length of 240 bytes. When the program is started, block PDAT_AC2 transfers a **single** job to the drive or, in the case of a multi-axis drive, to one of its axes. The block then receives the response data from the drive, or the drive axis, and stores it in a response data block. Each response comprises the "response header" and a value field. The response data are always 240 bytes long as well.

There is no coordination mechanism for the jobs received from different applications. Jobs must be coordinated within the applications themselves. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

Example for the structure of the job DB

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	REQUEST_1	STRUCT		Change parameter value, single
+0.0	REQUEST_HEADER	"HEADER"		
+4.0	PARAMETER_ADDRESS	"PARAMETERADDR"		
+10.0	FORMAT	BYTE	B#16#42	Format: WORD
+11.0	NO_VALUES	BYTE	B#16#1	
+12.0	VALUES	ARRAY[0..113]		
*2.0		WORD		
=240.0		END_STRUCT		
+240.0	REQUEST_2	STRUCT		Request parameter value, single
+0.0	REQUEST_HEADER	"HEADER"		
+4.0	PARAMETER_ADDRESS	"PARAMETERADDR"		
+10.0	VALUES	ARRAY[0..114]		
*2.0		WORD		
=240.0		END_STRUCT		
+480.0	REQUEST_3	STRUCT		Change parameter value, multi-parameter
+0.0	REQUEST_HEADER	"HEADER"		
+4.0	PARAMETER_ADDRESS_1	"PARAMETERADDR"		
+10.0	PARAMETER_ADDRESS_2	"PARAMETERADDR"		
+16.0	FORMAT1	BYTE	B#16#42	Format: WORD
+17.0	NO_VALUES1	BYTE	B#16#1	single (array) element
+18.0	VALUES1	ARRAY[0..0]		
*2.0		WORD		
+20.0	FORMAT2	BYTE	B#16#42	Format: WORD
+21.0	NO_VALUES2	BYTE	B#16#5	several array elements
+22.0	VALUES2	ARRAY[0..108]		
*2.0		WORD		
=240.0		END_STRUCT		
+720.0	REQUEST_4	STRUCT		Request parameter value, multi-parameter
+0.0	REQUEST_HEADER	"HEADER"		
+4.0	PARAMETER_ADDRESS_1	"PARAMETERADDR"		
+10.0	PARAMETER_ADDRESS_2	"PARAMETERADDR"		
+16.0	VALUES	ARRAY[0..111]		
*2.0		WORD		
=240.0		END_STRUCT		
=960.0		END_STRUCT		

The header and parameter address are to be created as a UDT data type. The numbers of the UDT are freely selectable.

Structure of the header UDT (REQUEST_1)

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	REQ_REF	BYTE	B#16#1	Request reference
+1.0	REQ_ID	BYTE	B#16#2	Request ID
+2.0	AXIS	BYTE	B#16#0	Axis addressing
+3.0	NO_PARAM	BYTE	B#16#1	Number of parameters
=4.0		END_STRUCT		

Structure of the parameter address UDT (REQUEST_1)

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	ATTRIBUTE	BYTE	B#16#10	Attribute
+1.0	NO_ELEM	BYTE	B#16#0	Number of elements
+2.0	PARA_NO	WORD	W#16#3	Parameter number
+4.0	SUBINDEX	WORD	W#16#0	Subindex
=6.0		END_STRUCT		

Example for the structure of the response DB

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	RESPONSE_1	STRUCT		
+0.0	RESPONSE_HEADER	"HEADER"		
+4.0	VALUE	ARRAY[0..117]		
*2.0		WORD		
=240.0		END_STRUCT		
+240.0	RESPONSE_2	STRUCT		
+0.0	RESPONSE_HEADER	"HEADER"		
+4.0	VALUE	ARRAY[0..117]		
*2.0		WORD		
=240.0		END_STRUCT		
+480.0	RESPONSE_3	STRUCT		
+0.0	RESPONSE_HEADER	"HEADER"		
+4.0	VALUE	ARRAY[0..117]		
*2.0		WORD		
=240.0		END_STRUCT		
+720.0	RESPONSE_4	STRUCT		
+0.0	RESPONSE_HEADER	"HEADER"		
+4.0	VALUE	ARRAY[0..117]		
*2.0		WORD		
=240.0		END_STRUCT		
=960.0		END_STRUCT		

At the input DB_NO_OR of the PDAT_AC2, the number of the job DB is indicated. The start address of the job to be transmitted is indicated at the input OFFSET_OR. The number of the response DB is indicated at the input DB_NO_AN and, at the input OFFSET_AN, the address is given from where the response of the drive is to be stored. A new job is transferred when a positive edge appears at the START input of the block. The job is then executed exactly once. The START bit must be reset with the BUSY signal. The next job can then be written to the transmit interface but must not be started until checkback signal DONE or xyz_ERR is started.

There is no coordination of jobs requested by different applications. This must be implemented within the applications if necessary. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

On multi-axis drives, the axis is addressed via the "Axis" byte in the parameter job header.

The block supports the following tasks:

- Read/write parameter value, simple (word/double word)
- Read/write parameter value, one or more array elements (234 max.)
- Read/write parameter value, multi-parameter
- Read parameter description, completely or a single descriptive element
- Read parameter texts, individually or several text elements
- Read parameters, multi-parameters with different attributes (value, description, text).

NOTE

A detailed description of the structure of parameter jobs and the associated responses can be found in Section 7.3 "Formulating parameter jobs (data record 47)".

Communications interface

Parameter	Declaration	Data type	Memory area	Description
BUSY	OUTPUT	BOOL	E, A, M, D, L	Job in progress
CFG_ERR	OUTPUT	BOOL	E, A, M, D, L	Slave not configured or incorrectly configured
DB_ACT	OUTPUT	INT	E, A, M, D, L	Job DB currently being processed
DB_ERR	OUTPUT	BOOL	E, A, M, D, L	Error in job DB or response DB
DB_ERRNO	OUTPUT	INT	E, A, M, D, L	Error code for DB_ERR
DB_NO_AN	INPUT	INT	E, A, M, D, L, Const.	DB No. of response DB
DB_NO_OR	INPUT	INT	E, A, M, D, L, Const.	DB No. of job DB
DONE	OUTPUT	BOOL	E, A, M, D, L	Job finished without error
DS_NO	INPUT	WORD	E, A, M, D, L, Const.	Number of data record to be read out
LADDR	INPUT	WORD	E, A, M, D, L, Const.	Diagnostic address of slave
OFFSET_ACT	OUTPUT	INT	E, A, M, D, L	Offset of job in the job DB
OFFSET_AN	INPUT	INT	E, A, M, D, L, Const.	Start address of area after which the response data can be stored
OFFSET_OR	INPUT	INT	E, A, M, D, L, Const.	Start address of job in the job DB
REQ_ERR	OUTPUT	BOOL	E, A, M, D, L	Job finished with error (response contains error identifier)
SFB_ERR	OUTPUT	BOOL	E, A, M, D, L	SFB 53 WRREC / SFB 52 RDREC signals error
START	INPUT	BOOL	E, A, M, D, L	Accept start pulse for job
WDOG_ERR	OUTPUT	BOOL	E, A, M, D, L	Watchdog error, no plausible response data available

The signaling bits are valid until the next job is received. The outputs are deleted from the drive when the new data are received. The new data are then sent to the outputs.

Data area

Parameter	Declaration	Data type	Description
si_NODATA_CYCLE_NO	STAT	INT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
si_NODATA_RETRY_NO	STAT	INT	Number of job repetitions if no response data have been received (default = 5)
si_SFB52_RET_VAL	STAT	INT	Return value of SFB 52 RDREC
si_SFB53_RET_VAL	STAT	INT	Return value of SFB53 WRREC
si_WDOG_RETRY_NO	STAT	INT	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	The configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: Logical basic address = 0 No parameter setting authorization on drive
DB_ERR	Error in job DB or response DB
DB_ERRNO	Error code for DB_ERR 0 = No error occurred 1 = No job DB or incorrect length of DB 2 = No response DB or incorrect length of DB 3 = Number of parameters < 1 or > 39 4 = Number of elements < 0 or > 234
REQ_ERR	Response contains error identifier The response also contains a more detailed description of the error (see Section 7.2 Formulating parameter jobs (data record 100)
SFB_ERR	SFB error during data transmission using system functions SFB RDREC / SFB WRREC (return value < 0) The return value is stored in instance DB (SFB53_RET_VAL or SFB52_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14. The receive mailbox is cleared if an error in reading the data occurs.
WDOG_ERR	No plausible response data within monitoring period

NOTE:

SFB_ERR is not signaled immediately on some CPUs of the S7-300 (older versions) if the communication link is faulty and a job has been initiated. Instead, the block continues to signal BUSY until either the error is eliminated so that the job can be completed or the cycle monitoring function (default setting: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been cancelled, you must restart your job.

Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.

Block call (STL source code)

```
NETWORK
TITLE =Check starting conditions of PDAT_AC2

      U   "DI_PDAT_AC2".SFB_ERR;  // Retrigger of START in case of SFB error
      UN  "DI_PDAT_AC2".START;
      =   "DI_PDAT_AC2".START;
      U   "DI_PDAT_AC2".SFB_ERR;
      SPB  PDAT;

      UN  "DI_PDAT_AC2".BUSY;     // Starting condition after complete
      UN  "DI_PDAT_AC2".READY;    // download of the program
      UN  "DI_PDAT_AC2".SFB_ERR;
      UN  "DI_PDAT_AC2".WDOG_ERR;
      UN  "DI_PDAT_AC2".CFG_ERR;
      UN  "DI_PDAT_AC2".REQ_ERR;
      S   "DI_PDAT_AC2".START;

      U   "DI_PDAT_AC2".BUSY;     // Reset of START
      R   "DI_PDAT_AC2".START;

      U   "DI_PDAT_AC2".READY;    // Start new request
      S   "DI_PDAT_AC2".START;

      U   "DI_PDAT_AC2".REQ_ERR;  // Start new request or error routine
      S   "DI_PDAT_AC2".START;

      U   "DI_PDAT_AC2".WDOG_ERR; // Start new request or error routine
      S   "DI_PDAT_AC2".START;

      U   "DI_PDAT_AC2".CFG _ERR; // Start new request or error routine
      S   "DI_PDAT_AC2".START;

NETWORK
TITLE =assembling acyclic requests

      UN  "DI_PDAT_AC2".START;
      SPB  PDAT;

// assemble acyclic requests

NETWORK
TITLE =  Call for PDAT_AC2

PDAT: CALL  PDAT_AC2, DI_PDAT_AC2(
            LADDR     :=  "DRIVDB1".SLAVE_5.DADDR,
            START     :=  ,
            DS_NO     :=  MW16,
            DB_NO_OR  :=  MW0,
            OFFSET_OR :=  MW2,
            DB_NO_AN  :=  MW4,
            OFFSET_AN :=  MW6,
            BUSY      :=  ,
            READY     :=  ,
            REQ_ERR   :=  ,
            WDOG_ERR  :=  ,
            SFB_ERR   :=  ,
            CFG_ERR   :=  ,
            DB_ERR    :=  M8.0,
            DB_ERRNO  :=  MW10,
            DB_ACT    :=  MW12,
            OFFSET_AC :=  MW14);

NETWORK
TITLE = interpret answer according to the containing data
```

3.2.6 FB DEV_FLT1: Read out fault buffer of a MASTERDRIVES

FB 35

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task:	OB1
Cyclic interrupt OB:	e.g. OB32

Description of function

The block reads out the last fault stored in the fault memory of the drive. A read operation is initiated by a positive edge at the START input of the block.

The START bit must be reset by the BUSY signal. A new block call may not be made until checkback signal READY or xyz_ERR arrives. The FB uses the acyclic communication mechanism for the data transfer. There is no coordination of the block calls which use the acyclic communication channel for the same slave. This must be implemented within the applications if necessary. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

Communications interface

Parameter	Declaration	Data type	Memory area	Description	
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Job in progress	
CFG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Slave not configured or incorrectly configured	
ERR_NO1	OUTPUT	WORD	I, Q, M, D, L	Number of fault 1 of fault event	
ERR_TXT1	OUTPUT	STRING[16]	D, L	Fault text for fault number of fault 1	
ERR_VAL1	OUTPUT	WORD	I, Q, M, D, L	Fault value for fault number of fault 1	
...					
ERR_NO8	OUTPUT	WORD	I, Q, M, D, L	Number of fault 8 of fault event	
ERR_TXT8	OUTPUT	STRING[16]	D, L	Fault text for fault number of fault 8	
ERR_VAL8	OUTPUT	WORD	I, Q, M, D, L	Fault value for fault number of fault 8	
ETC_DAY	OUTPUT	WORD	I, Q, M, D, L	Date of fault event	Hours run counter
ETC_HOUR	OUTPUT	BYTE	I, Q, M, D, L	Hour of fault event	
ETC_MIN	OUTPUT	BYTE	I, Q, M, D, L	Minute of fault event	
ETC_SEC	OUTPUT	BYTE	I, Q, M, D, L	Second of fault event	
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Diagnostic address of slave	
READY	OUTPUT	BOOL	I, Q, M, D, L	Job finished without error	
REQ_ERR	OUTPUT	BOOL	I, Q, M, D, L	Response contains error identifier	
SFB_ERR	OUTPUT	BOOL	I, Q, M, D, L	SFB 53 WRREC / SFB 52 RDREC signals error	
START	INPUT	BOOL	I, Q, M, D, L	Accept start pulse for job	
WDOG_ERR	OUTPUT	BOOL	I, Q, M, D, L	Watchdog error, no plausible response data available	

If a job is processed correctly, the signaling bits and data at the block outputs remain valid until the next job is initiated.

Data area

Parameter	Declaration	Data type	Description
NODATA_CYCLE_NO	STAT	INT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
NODATA_RETRY_NO	STAT	BYTE	Number of job repetitions if no response data have been received (default = 5)
SFB52_RET_VAL	STAT	INT	Return value of SFB 52 RDREC
SFB53_RET_VAL	STAT	INT	Return value of SFB53 WRREC
WDOG_RETRY_NO	STAT	BYTE	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier = 8)
REQ_ERR	Job contains error identifier (response identifier = 7)
SFB_ERR	SFB error with data transfer using system functions SFB RDREC/WRREC (return value < 0) The return value is stored in instance DB (SFB53_RET_VAL or SFB52_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14. No fault information is output if errors occur in writing or reading the data.
WDOG_ERR	No plausible response data within monitoring period (default: After five job repetitions)

NOTES:

- If a xyz_ERR occurs, no information about the fault is displayed at the block outputs.
- SFB_ERR is not signaled immediately on some CPUs of the S7-300 (older versions!) when the communication link is faulty and DEV_FLT1 is active. Instead, the block continues to signal BUSY until either the error is eliminated so that the fault memory readout job can be completed, or until the cycle monitoring function (default: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been cancelled, you must restart the DEV_FLT1. Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.
- When the fault memory of MASTERDRIVES with a CU1 or CU2 board is read out, it is possible that the fault texts will be incorrectly displayed.
- If an attempt is made to use DEV_FLT1 to read out the fault memory of a type of drive other than MASTERDRIVES, an incorrect result can occur.

Block call (STL source code)

```
U     Read out device fault
S     "DB_DEV_FLT1".START

CALL  DEV_FLT1, DB_DEV_FLT1(
          LADDR        := DRIVDB1.SLAVE_1.DADDR,
          START        := ,
          BUSY         := ,
          READY        := M30.2,
          REQ_ERR      := M30.3,
          WDOG_ERR     := M30.4,
          SFB_ERR      := M30.5,
          CFG_ERR      := M30.6,
          ETC_DAY      := MOTOR.DAY,
          ETC_HOUR     := MOTOR.HOUR,
          ETC_MIN      := MOTOR.MIN,
          ETC_SEC      := MOTOR.SEC,
          ERR_NO1      := MOTOR.ERR_NO1
          ERR_TXT1     := MOTOR.ERR_TXT1,
          ERR_VAL1     := MOTOR.ERR_VAL1,
          ERR_NO2      := MOTOR.ERR_NO2
          ERR_TXT2     := MOTOR.ERR_TXT2,
          ERR_VAL2     := MOTOR.ERR_VAL2,
          ERR_NO3      := MOTOR.ERR_NO3
          ERR_TXT3     := MOTOR.ERR_TXT3,
          ERR_VAL3     := MOTOR.ERR_VAL3,
          ERR_NO4      := MOTOR.ERR_NO4
          ERR_TXT4     := MOTOR.ERR_TXT4,
          ERR_VAL4     := MOTOR.ERR_VAL4,
          ERR_NO5      := MOTOR.ERR_NO5,
          ERR_TXT5     := MOTOR.ERR_TXT5,
          ERR_VAL5     := MOTOR.ERR_VAL5,
          ERR_NO6      := MOTOR.ERR_NO6,
          ERR_TXT6     := MOTOR.ERR_TXT6,
          ERR_VAL6     := MOTOR.ERR_VAL6,
          ERR_NO7      := MOTOR.ERR_NO7,
          ERR_TXT7     := MOTOR.ERR_TXT7,
          ERR_VAL7     := MOTOR.ERR_VAL7,
          ERR_NO8      := MOTOR.ERR_NO8,
          ERR_TXT8     := MOTOR.ERR_TXT8,
          ERR_VAL8     := MOTOR.ERR_VAL8);

U     "DB_DEV_FLT1".BUSY;
R     "DB_DEV_FLT1".START;
```

3.2.7 FB DEV_FLT2: Read out fault buffer of a DC-MASTER

FB 37

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task:	OB1
Cyclic interrupt OB:	e.g. OB32

Description of function

The block reads out the last fault stored in the fault memory of the drive. A read operation is initiated by a positive edge at the START input of the block.

The START bit must be reset by the BUSY signal. A new block call may not be made until checkback signal READY or xyz_ERR arrives. The FB uses the acyclic communication mechanism for the data transfer. There is no coordination of the block calls which use the acyclic communication channel for the same slave. This must be implemented within the applications if necessary. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

Communications interface

Parameter	Declaration	Data type	Memory are	Description
BUSY	OUTPUT	BOOL	E, A, M, D, L	Job in progress
CFG_ERR	OUTPUT	BOOL	E, A, M, D, L	Slave not configured or incorrectly configured
ERR_NO1	OUTPUT	WORD	E, A, M, D, L	Number of fault 1 of fault event
ERR_TXT1	OUTPUT	STRING[16]	D, L	Fault text for fault number of fault 1
ERR_VAL1	OUTPUT	WORD	E, A, M, D, L	Fault value for fault number of fault 1
...				
ERR_NO8	OUTPUT	WORD	E, A, M, D, L	Number of fault 8 of fault event
ERR_TXT8	OUTPUT	STRING[16]	D, L	Fault text for fault number of fault 8
ERR_VAL8	OUTPUT	WORD	E, A, M, D, L	Fault value for fault number of fault 8
ETC_HOUR	OUTPUT	WORD	E, A, M, D, L	Hour of fault event (hours run counter)
LADDR	INPUT	WORD	E, A, M, D, L, Const.	Diagnostic address of slave
READY	OUTPUT	BOOL	E, A, M, D, L	Job finished without error
REQ_ERR	OUTPUT	BOOL	E, A, M, D, L	Response contains error identifier
SFB_ERR	OUTPUT	BOOL	E, A, M, D, L	SFB 53 WRREC /SFB 52 RDREC signals error
START	INPUT	BOOL	E, A, M, D, L	Accept start pulse for job
WDOG_ERR	OUTPUT	BOOL	E, A, M, D, L	Watchdog error, no plausible response data available

If a job is processed correctly, the signaling bits and data at the block outputs remain valid until the next job is initiated.

Data area

Parameter	Declaration	Data type	Description
NODATA_CYCLE_NO	STAT	INT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
NODATA_RETRY_NO	STAT	BYTE	Number of job repetitions if no response data have been received (default = 5)
SFB52_RET_VAL	STAT	INT	Return value of SFB 52 RDREC
SFB53_RET_VAL	STAT	INT	Return value of SFB 53 WRREC
WDOG_RETRY_NO	STAT	BYTE	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier = 8)
REQ_ERR	Job contains error identifier (response identifier = 7)
SFB_ERR	SFB error during data transfer using system functions SFB RDREC/WRREC (return value < 0) The return value is stored in instance DB (SFB53_RET_VAL or SFB52_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14.
WDOG_ERR	No plausible response data within monitoring period (default: After five job repetitions)

NOTE:

- If a xyz_ERR occurs, no information about the fault is displayed at the block outputs.
- SFB_ERR is not signaled immediately on some CPUs of the S7-300 (older versions!) when the communication link is faulty and DEV_FLT2 is active. Instead, the block continues to signal BUSY until either the error is eliminated so that the fault memory readout job can be completed, or until the cycle monitoring function (default: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been cancelled, you must restart the DEV_FLT2. Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.
- If an attempt is made to use DEV_FLT2 to read out the fault memory of a type of drive other than a DC-MASTER, an incorrect result can occur.

Block call (STL source code)

```
U     Read out device fault
S     "DB_DEV_FLT2".START

CALL  DEV_FLT2, DB_DEV_FLT2(
          LADDR        := DRIVDB1.SLAVE_1.DADDR,
          START        := ,
          BUSY         := ,
          READY        := M30.2,
          REQ_ERR      := M30.3,
          WDOG_ERR     := M30.4,
          SFB_ERR      := M30.5,
          CFG_ERR      := M30.6,
          ETC_HOUR     := MOTOR.HOUR,
          ERR_NO1      := MOTOR.ERR_NO1
          ERR_TXT1     := MOTOR.ERR_TXT1,
          ERR_VAL1     := MOTOR.ERR_VAL1,
          ERR_NO2      := MOTOR.ERR_NO2
          ERR_TXT2     := MOTOR.ERR_TXT2,
          ERR_VAL2     := MOTOR.ERR_VAL2,
          ERR_NO3      := MOTOR.ERR_NO3
          ERR_TXT3     := MOTOR.ERR_TXT3,
          ERR_VAL3     := MOTOR.ERR_VAL3,
          ERR_NO4      := MOTOR.ERR_NO4
          ERR_TXT4     := MOTOR.ERR_TXT4,
          ERR_VAL4     := MOTOR.ERR_VAL4,
          ERR_NO5      := MOTOR.ERR_NO5,
          ERR_TXT5     := MOTOR.ERR_TXT5,
          ERR_VAL5     := MOTOR.ERR_VAL5,
          ERR_NO6      := MOTOR.ERR_NO6,
          ERR_TXT6     := MOTOR.ERR_TXT6,
          ERR_VAL6     := MOTOR.ERR_VAL6,
          ERR_NO7      := MOTOR.ERR_NO7,
          ERR_TXT7     := MOTOR.ERR_TXT7,
          ERR_VAL7     := MOTOR.ERR_VAL7,
          ERR_NO8      := MOTOR.ERR_NO8,
          ERR_TXT8     := MOTOR.ERR_TXT8,
          ERR_VAL8     := MOTOR.ERR_VAL8);

U     "DB_DEV_FLT2".BUSY;
R     "DB_DEV_FLT2".START;
```

3.2.8 FB DEV_FLT3: Read out fault buffer of a MICROMASTER 4xx

FB 38

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task: OB1

Cyclic interrupt OB: e.g. OB32

Description of function

The block reads out the last fault stored in the fault memory of the drive. A read operation is initiated by a positive edge at the START input of the block.

The START bit must be reset by the BUSY signal. A new block call may not be made until checkback signal DONE or xyz_ERR arrives. The FB uses the acyclic communication mechanism for the data transfer. There is no coordination of the block calls which use the acyclic communication channel for the same slave. This must be implemented within the applications if necessary. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

Communications interface

Parameter	Declaration	Data type	Memory are	Description
BUSY	OUTPUT	BOOL	E, A, M, D, L	Job in progress
CFG_ERR	OUTPUT	BOOL	E, A, M, D, L	Slave not configured or incorrectly configured
DONE	OUTPUT	BOOL	E, A, M, D, L	Job finished without error
ERR_NO1	OUTPUT	WORD	E, A, M, D, L	Number of fault 1 of fault event
ERR_VAL1	OUTPUT	WORD	E, A, M, D, L	Fault value for fault number of fault 1
ERR_NO2	OUTPUT	WORD	E, A, M, D, L	Number of fault 2 of fault event
ERR_VAL2	OUTPUT	WORD	E, A, M, D, L	Fault value for fault number of fault 2
LADDR	INPUT	WORD	E, A, M, D, L, Const.	Diagnostic address of slave
REQ_ERR	OUTPUT	BOOL	E, A, M, D, L	Response contains error identifier
SFB_ERR	OUTPUT	BOOL	E, A, M, D, L	SFB 53 WRREC / SFB 52 RDREC signals error
START	INPUT	BOOL	E, A, M, D, L	Accept start pulse for job
WDOG_ERR	OUTPUT	BOOL	E, A, M, D, L	Watchdog error, no plausible response data available

If a job is processed correctly, the signaling bits and data at the block outputs remain valid until the next job is initiated.

Data area

Parameter	Declaration	Data type	Description
NODATA_CYCLE_NO	STAT	INT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
NODATA_RETRY_NO	STAT	BYTE	Number of job repetitions if no response data have been received (default = 5)
SFB52_RET_VAL	STAT	INT	Return value of SFB 52 RDREC
SFB53_RET_VAL	STAT	INT	Return value of SFB 53 WRREC
WDOG_RETRY_NO	STAT	BYTE	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier = 8)
REQ_ERR	Job contains error identifier (response identifier = 7)
SFB_ERR	SFB error during data transfer using system functions SFB RDREC/WRREC (return value < 0) The return value is stored in instance DB (SFB53_RET_VAL or SFB52_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14.
WDOG_ERR	No plausible response data within monitoring period (default: After five job repetitions)

NOTE:

- If a xyz_ERR occurs, no information about the fault is displayed at the block outputs.
- SFB_ERR is not signaled immediately on some CPUs of the S7-300 (older versions!) when the communication link is faulty and DEV_FLT3 is active. Instead, the block continues to signal BUSY until either the error is eliminated so that the fault memory readout job can be completed, or until the cycle monitoring function (default: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been cancelled, you must restart the DEV_FLT3. Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.
- If an attempt is made to use DEV_FLT3 to read out the fault memory of a type of drive other than a MICROMASTER 4xx, an incorrect result can occur.

Block call (STL source code)

```
U      Read out device fault
S      "DB_DEV_FLT3".START

CALL   DEV_FLT3, DB_DEV_FLT3(
           LADDR        := DRIVDB1.SLAVE_1.DADDR,
           START        := ,
           BUSY         := ,
           DONE         := M30.2,
           REQ_ERR      := M30.3,
           WDOG_ERR     := M30.4,
           SFB_ERR      := M30.5,
           CFG_ERR      := M30.6,
           ERR_NO1      := MOTOR.ERR_NO1
           ERR_VAL1     := MOTOR.ERR_VAL1,
           ERR_NO2      := MOTOR.ERR_NO2
           ERR_VAL2     := MOTOR.ERR_VAL2,

U      "DB_DEV_FLT3".BUSY;
R      "DB_DEV_FLT3".START;
```

3.2.9 FB DEV_FLT4: Read out fault buffer of a SINAMICS G/S

FB 39

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task:	OB1
Cyclic interrupt OB:	e.g. OB32

Description of function

The block reads out the last fault stored in the fault memory of the drive. A read operation is initiated by a positive edge at the START input of the block. The telegram for reading out the fault memory of the drive is stored in the block. Its structure conforms to PROFIdrive Profile Drive Technology, Version 3.

The number of the data record to be used can be entered via input DS_NO. If the value 0 is assigned to the input, data record 47 is used. In the case of values <> 0, the value at the input is interpreted as the number of the data record to be used. However, the telegram must still be structured in accordance with PROFIdrive Profile Drive Technology, Version 3 as the block will otherwise report errors, i.e. any data record supported by the drive can be used provided the contents of the telegram conform to PROFIdrive Profile Drive Technology, Version 3. This function is relevant mainly for drives with a PROFInet interface.

The START bit must be reset by the BUSY signal. A new block call may not be made until checkback signal DONE or xyz_ERR arrives. The FB uses the acyclic communication mechanism for the data transfer. There is no coordination of the block calls which use the acyclic communication channel for the same slave. This must be implemented within the applications if necessary. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

Communications interface

Parameter	Declaration	Data type	Memory are	Description
AXIS[1]	INPUT	BYTE	E, A, M, D, L, Const.	Drive ID on multi-axis drives
BUSY	OUTPUT	BOOL	E, A, M, D, L	Job in progress
CFG_ERR	OUTPUT	BOOL	E, A, M, D, L	Slave not configured or incorrectly configured
DONE	OUTPUT	BOOL	E, A, M, D, L	Job finished without error
DS_NO	INPUT	WORD	E, A, M, D, L, Const.	Number of data record to be read out
ERR_NO1	OUTPUT	WORD	E, A, M, D, L	Number of fault 1 of fault event
ERR_VAL1	OUTPUT	WORD	E, A, M, D, L	Fault value for fault number of fault 1
...				
ERR_NO8	OUTPUT	WORD	E, A, M, D, L	Number of fault 8 of fault event
ERR_VAL8	OUTPUT	WORD	E, A, M, D, L	Fault value for fault number of fault 8
LADDR	INPUT	WORD		Diagnostic address of slave
REQ_ERR	OUTPUT	BOOL	E, A, M, D, L	Response contains error identifier
SFB_ERR	OUTPUT	BOOL	E, A, M, D, L	SFB 53 WRREC /SFB 52 RDREC signals error
START	INPUT	BOOL	E, A, M, D, L	Accept start pulse for job
WDOG_ERR	OUTPUT	BOOL	E, A, M, D, L	Watchdog error, no plausible response data available

If a job is processed correctly, the signaling bits and data at the block outputs remain valid until the next job is initiated.

[1] See Subsection 7.6.2 (SINAMICS G) or Subsection 7.6.3 (SINAMICS S)

Data area

Parameter	Declaration	Data type	Description
NODATA_CYCLE_NO	STAT	INT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
NODATA_RETRY_NO	STAT	BYTE	Number of job repetitions if no response data have been received (default = 5)
SFB52_RET_VAL	STAT	INT	Return value of SFB 52 RDREC
SFB53_RET_VAL	STAT	INT	Return value of SFB 53 WRREC
WDOG_RETRY_NO	STAT	BYTE	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	A configuration error is displayed if no configuration data or faulty configuration data are entered. The following data are checked: ➢ Logical basic address = 0
REQ_ERR	Job contains error identifier (response identifier = 7)
SFB_ERR	SFB error during data transfer using system functions SFB RDREC/WRREC (return value < 0) The return value is stored in instance DB (SFB53_RET_VAL or SFB52_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14.
WDOG_ERR	No plausible response data within monitoring period (default: After five job repetitions)

NOTE:

- If an xyz_ERR occurs, no information about the fault is displayed at the block outputs.
- SFB_ERR is not signaled immediately on some CPUs of the S7-300 (older versions!) when the communication link is faulty and DEV_FLT4 is active. Instead, the block continues to signal BUSY until either the error is eliminated so that the fault memory readout job can be completed, or until the cycle monitoring function (default: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been cancelled, you must restart the DEV_FLT4. Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.
- If an attempt is made to use DEV_FLT4 to read out the fault memory of a type of drive other than a SINAMICS G/S, an incorrect result can occur.
- Entry of the fault information in the SINAMICS fault buffer is delayed in relation to the message via bit 3 of status word 1. This must be taken into account when the block call is programmed. See e.g. station 08_SIMATIC300_S120_ALL_s in sample project Zxy51_03_DriveES_SAMP

Block call (STL source code)

```
U      Read out device fault
S      "DB_DEV_FLT4".START

CALL   DEV_FLT4, DB_DEV_FLT4(
           LADDR        := DRIVDB1.SLAVE_1.DADDR,
           DS_NO        := MW32,
           START        := ,
           BUSY         := ,
           DONE         := M30.2,
           REQ_ERR      := M30.3,
           WDOG_ERR     := M30.4,
           SFB_ERR      := M30.5,
           CFG_ERR      := M30.6,
           ERR_NO1      := MOTOR.ERR_NO1
           ERR_VAL1     := MOTOR.ERR_VAL1,
           ERR_NO2      := MOTOR.ERR_NO2
           ERR_VAL2     := MOTOR.ERR_VAL2,
           ERR_NO3      := MOTOR.ERR_NO3
           ERR_VAL3     := MOTOR.ERR_VAL3,
           ERR_NO4      := MOTOR.ERR_NO4
           ERR_VAL4     := MOTOR.ERR_VAL4,
           ERR_NO5      := MOTOR.ERR_NO5,
           ERR_VAL5     := MOTOR.ERR_VAL5,
           ERR_NO6      := MOTOR.ERR_NO6,
           ERR_VAL6     := MOTOR.ERR_VAL6,
           ERR_NO7      := MOTOR.ERR_NO7,
           ERR_VAL7     := MOTOR.ERR_VAL7,
           ERR_NO8      := MOTOR.ERR_NO8,
           ERR_VAL8     := MOTOR.ERR_VAL8);

U      "DB_DEV_FLT34".BUSY;
R      "DB_DEV_FLT34".START;
```

3.2.10 FB PDAT_DL: Download drive parameters (DS 100)

FB 40

The block number can be altered.

This block cannot be used on some series 300 CPUs on which the SFC24 (TEST_DB) is not installed.

In the case of CPUs with MMC memory cards, the section "Data Management in the Loading Memory" must be complied with (FB50 "SHELL_MMC_PDAT_DL").

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task:	OB1
Cyclic interrupt OB:	e.g. OB32

Features

- Can be used for MASTERDRIVES and DC-MASTER
- Download / partial download functionality
- The parameter DB structure must comply with the "Structure of parameter job DB" definition ..." (see Subsection 7.4.1)
- If the parameter DB is generated by the system ("Convert parameter set to DB" function in the SIMATIC Manager"), the Drive ES Drive Monitor must also be installed. This function can be used only for MASTERDRIVES and DC Master and generates DS 100 jobs.

Description of function

The block transfers the parameterizing data of a drive from a data area of the CPU to the drive. The parameterizing data can be distributed among several data blocks. The data blocks themselves are either in the working memory or in the loading memory of the CPU. Only write jobs are supported. The corresponding identifiers are checked by the block. A job with a non-permitted job identifier is ignored and not signaled as an error by the block. The jobs can vary in length. The response data are not permanently stored. The block is also capable of transferring individual sections of the DB/DBs. This requires a modification to the DB structure (see Subsection 7.4.1.2).

The FB uses the acyclic communication mechanism for the data transfer. The jobs are formulated according to the PIV mechanism in line with the USS protocol specification (data record 100).

The jobs which are requested by different applications are not coordinated. This must be implemented within the applications. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

In the case of multi-axis drives, "download" must be carried out for each axis.

Each individual axis is addressed via the logical basic address (I/O basic address) of any slot (exception: PIV slot or empty slot) from the area of the axis to which data is to be sent:

The block supports the following tasks:

- Write parameter value, simple (word/double word): PIV job identifiers 2, 3, 13, 14
- Write parameter value, array (word/double word/several array elements): PIV job identifiers 7, 8, 11, 12

NOTE

If a task with the index 255 is used, the required parameter values must be entered in the task for all indices. Otherwise 0 will be written to the indices for which no parameter value is entered.

Data retention in load memory

Data blocks which are programmed in a source file as part of an STL program can be characterized as "not relevant to the process (key word UNLINKED). This means that, during loading into the CPU, these data blocks are only stored in the load memory. Their content is then copied into the working memory. This function is integrated in block PDAT_DL.

Because data are kept in the load memory, space in the working memory can be saved. The extendable load memory acts as an intermediate memory (e.g. for parameter DB: only the parameter DB which is to be processed is loaded into the working memory).

If the data block is produced with the parameter "UNLINKED", the input "DB_UNLINKED" must be set to "TRUE". At input "DB_NO", the number of the DB in the working memory into which the data is copied is indicated. The number of the first parameter DB in the load memory is at input DB_NO_LM.

If the input "DB_UNLINKED" has been set to "FALSE", the number of the first parameter DB in the working memory must be indicated at the input "DB_NO". The input "DB_NO_LM" is then irrelevant.

The copy DB in the working memory and the data blocks in the load memory must be 8192 bytes in size (see Section 7.7 Tip for example of how to generate a corresponding DB). If the data are only kept in the working memory, the data blocks can be even larger or smaller than 8192 bytes, depending on the CPU.

If several parameter data blocks are copied into the load memory or working memory, the next data block is indicated at the end of each of these data blocks. If no further data block follows, "Number of the next data block" = 0 is to be entered in the last data word.

CPU with MMC memory card:

If a CPU with an MMC memory card is used, the shell block FB50 (SHELL_MMC_PDAT_DL) must be used. FB50 has the same I/O strip as the FB40 and calls it internally. As a result, the call of FB40 can easily be replaced by the call of FB50. FB50 has no KNOW HOW protection. The number of FB40 can therefore be changed at any time and only the call in FB50 has to be adapted.

NOTE:

If you have used the function "Convert parameter set to DB" (for MASTERDRIVES and DC Master only) to create several DBs, and if some or all DB numbers are outside the permissible DB numerical range for the target CPU, you will need to adjust the DB numbers to the CPU-specific range afterwards. In this case, you must also alter the number of the following DB in each DB!

Convert parameter set to DB: available in the SIMATIC-Manager > menu options: *Edit* > *Convert Parameter Set to DB* after Drive ES Basic has been installed

Error logging

With the input "LOG_FCT", the reaction of block PDAT_DL to an error during download (REQ_ERR) is parameterized:

- If the status of the input is "FALSE", download in stopped when the first error occurs. The number of the parameter DB currently being processed is indicated at the output "DB_NO_ACT". The number of the parameter is indicated at the output "PA_NO" and the error number is indicated at output "ERR_NO" (NOT_TERMINATED = TRUE).
- If the status of the input is "TRUE", download is not stopped when a "REQ_ERR" occurs. The error is logged (log file is in the instance of the DB under sx_LOG...) and downloading is continued. The DB number, the parameter number, the index and the error number are stored in the log. Downloading is only stopped after 20 logged errors (NOT_TERMINATED = TRUE). At the outputs "DB_NO_ACT", "PA_NO" and "ERR_NO", the data of the last error are indicated (number of the parameter DB, parameter number and error number).

Downloading can be stopped at the end of each parameter job with the input "CANCEL" (NOT_TERMINATED = TRUE).

Ignoring SFB errors

In the instance data of the block (sx_PARA_NO. sw_PARA[0] ... sx_PARA_NO. sw_PARA[4]), it is possible to enter whether the SFB error is to be displayed for specific parameters. According to the default setting, these are parameters 970, 971 and 972. They can be altered by the user by means of "Modify variable". It is necessary to enter -1 in the storage cells that are not being used. It only makes sense to use this function if the drive breaks off communication for a short time when these parameters are being written (copying of RAM2ROM, PowerOnReset). In addition, these parameters should be at the end of the jobs to be processed because no check is made to see when the drive is ready for communication again and therefore subsequent jobs report an SFB error again, causing download to be stopped. This also means that each download is only permitted to contain one job from the above list. If several parameters from the above list have to be transferred, it is necessary to resort to the 'partial download' function.

Processing orders

1. Enter the number of the first parameter DB into the corresponding interface:
   - If the parameter DB/DBs Is/are in the **working memory**, the number of the first parameter DB is indicated at the input DB_NO (Parameter DB_UNLINKED = 0).
   - If the parameter DB/DBs is/are in the **load memory**, the number of the DB into which the data from the load memory are to be copied is indicated at the input DB_NO. The number of the first parameter DM in the load memory is indicated at input DB_NO_LM (Parameter DB_UNLINKED = 1).
2. Enter the start address (corresponds to the address of the field in which the version identifier is stored) of the first job or partial job at input START_ADDR.
3. Initiate download with the start bit (START).
4. Block checks whether the DB or DBs indicated exist(s) in the CPU (if not: DB_ERR = 1).
5. The block takes the first job from the parameter DB, enters it into the transmit buffer and transmits it to the drive (BUSY = TRUE). In addition, the number of the current parameter to be transmitted is shown in parameter PA_NO.

6. The block checks the data received from the drive to see whether:
   - the response identifier matches the job identifier
   - the received parameter number matches the one sent
   - the received parameter index matches the one sent
7. Response in the receive mail box is positive

   If the parameter CANCEL = FALSE, the next job from the parameter DB is taken and transmitted. This is repeated until all jobs have been sent to the drive and processed (DONE = TRUE, BUSY = FALSE).

   In order to stop transmission, the "CANCEL" parameter must be set to "TRUE". If this is the case, transmission is stopped when the parameter being transmitted ends. (NOT_TERMINATED = TRUE). In addition, the current DB number at the output DB_NO_ACT and the number of the last parameter processed are indicated at the output PA_NO.
8. Response in the receive mail box is negative (response identifier = 7dec)

   Job finished with error (REQ_ERR). The error number is located in the parameter value of the response. This error number is indicated in the output parameter ERR_NO. If the logging function is active, the defective partial job is logged.
9. No **plausible** response to the job which was sent:
   - Job is transmitted to the drive again and the received data are checked.
   - The stipulated number of job repetitions has been carried out without a plausible response: watchdog error
10. There are **no** response data after a stipulated number of cycles (SFB signals error 80C0):
    - Job is transmitted to the drive again
    - The stipulated number of job repetitions has been carried out and there are still not response data: SFB error
11. Group error is signaled at the block output ERROR when one of the following errors occurs: REQ_ERR, WDOG_ERR, SFB_ERR, CFG_ERR, or DB_ERR.
12. When the first job is finished, the next job is executed. This process continues until the block encounters the next end identifier (16#EEEE EEEE). It then checks the next word to ascertain whether its value is < > 0 (next DB). If it is, the jobs are executed in this DB, otherwise the download/partial download is ended.

Communications interface

Parameter	Data type	Kind	Description
BUSY	BOOL	OUT	=1: Download in progress (parameters are being transmitted)
CANCEL	BOOL	IN	Stop download (download is stopped after completion of the current parameter job)
CFG_ERR	BOOL	OUT	Slave not configured or configured incorrectly
DB_ERR	BOOL	OUT	Error in parameter DB (for explanation, see DB_ERRNO)
DB_ERRNO	INT	OUT	Error code for DB_ERR 0 = No error occurred 1 = DB indicated does not exist in working memory 2 = DB indicated does not exist in load memory 3 = DB in working memory < > 8192 bytes 4 = Job > 240 bytes 5 = Address of the data to be copied is outside the data DB 7 = Wrong identifier for job start or end identifier 8 = Start address + minimum job length > DB size 9 = Wrong start address (start address + 2 <> data record identifier) 10 = Start address not at word limit or job length an uneven number
DB_NO	INT	IN	DB_UNLINKED = 1: Number of the DB in the working memory into which the data from the load memory are copied. DB_UNLINKED = 0: Number of the first parameter DB in the working memory
DB_NO_ACT	INT	OUT	Number of the current job DB being processed
DB_NO_LM	INT	IN	Number of the first parameter DB in the load memory
DB_UNLINKED	BOOL	IN	=1:Parameters DB(s) is (are) in the load memory
DONE	BOOL	OUT	=1:Download finished without errors (all parameter jobs have been transmitted without error)
ERR_NO	INT	OUT	Drive reports back error (from "PWE1" in the case of DS100, from "Error value" in the case of DS47)
ERROR	BOOL	OUT	Group error
LADDR	WORD	IN	Diagnostics address of the slave [1)]
LOG_FCT	BOOL	IN	=1:Errors during download are logged (20 errors max.)
NOT_TERMINATED	BOOL	OUT	Download stopped by user ➢ Stopped after first REQ_ERR (LOG_FCT = FALSE) ➢ Stopped after 20ieth REQ_ERR (LOG_FCT = TRUE)
PA_NO	INT	OUT	Number of the last edited parameter
REQ_ERR	BOOL	OUT	Response contains error identifier ("7" in the case of DS 100, "82h" in the case of DS 47)
SFB_ERR	BOOL	OUT	SFB 53 WRREC / SFB 52 RDREC signals error
START	BOOL	IN	Start download
START_ADDR	INT	IN	Absolute address as of which the data to be transferred are stored in the DB (corresponds to the address of the field in which the version identifier is stored) (see Subsection 7.4.1 Structure of the parameter job DB for FB PDAT_DL / PDAT_UD)
WDOG_ERR	BOOL	OUT	Watchdog error

1) The I/O basic address of any slot (exception: PIV or empty slot) from the area of the axis to be addressed must be specified here for multi-axis drives according to DS 100.

The signaling bits and the data are valid until initiation of the next download. They are deleted when PDAT_DL is started again.

Data area

Parameter	Data type	Kind	Comments
RD	ARRAY [1..240] of BYTE	STAT	Receive mail box for acyclical communication
SD	ARRAY [1..240] of BYTE	STAT	Send mail box for acyclical communication
si_NODATA_CYCLE_NO	INT	STAT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
si_SFB52_RET_VAL	INT	STAT	Return value of SFB 52 RDREC
si_SFB53_RET_VAL	INT	STAT	Return value of SFB 53 WRREC
sx_LOG	ARRAY [0..19] of STRUCT	STAT	Logged data of the parameter jobs containing errors
sx_LOG.si_DB_NO	INT	STAT	DB number of the parameter with the error
sx_LOG.si_INDEX	INT	STAT	Index of the parameter with the error
sx_LOG.si_PARA_NO	INT	STAT	Number of the parameter with the error
sx_LOG.sw_ERR_NO	WORD	STAT	Error code of the parameter with the error
sx_PARA_NO. sw_PARA[0]	WORD	STAT	Parameter numbers for which the SFB error is not evaluated (default: P970, 971, 972). The user can enter additional or different parameter numbers.
sx_PARA_NO. sw_PARA[1]	WORD	STAT	
sx_PARA_NO. sw_PARA[2]	WORD	STAT	
sx_PARA_NO. sw_PARA[3]	WORD	STAT	
sx_PARA_NO. sw_PARA[4]	WORD	STAT	
sy_NODATA_RETRY_NO	BYTE	STAT	Number of job repetitions if no response data have been received (default = 5)
sy_WDOG_RETRY_NO	BYTE	STAT	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	If no configuration data or incorrect configuration data have been entered, a configuration error is signaled. The following data are checked: ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier "8" in the case of DS100)
DB_ERR	Error in parameter DB Parameter-DB (for explanation, see DB_ERRNO)
DB_ERRNO	Error code for DB_ERR 0 = No error occurred 1 = DB indicated does not exist in working memory 2 = DB indicated does not exist in load memory 3 = DBs in the working memory < > 8192 bytes 4 = Job > 240 bytes 5 = Address of the data to be copied is outside the data DB 7 = Wrong identifier for job start or end identifier 8 = Start address + minimum job length > DB size 9 = Wrong start address (start address + 2 <> data record identifier) 10 = Start address not at word limit or job length an uneven number
ERR_NO	Drive reports back error from "PWE1" in the case of DS100, for description of error see Section 7.2 "Formulating parameter jobs (data record 100)"
ERROR	Group error
NOT_TERMINATED	Download stopped by user ➢ Stopped after first REQ_ERR (LOG_FCT = FALSE) ➢ Stopped after 20ieth REQ_ERR (LOG_FCT = TRUE)
REQ_ERR	Response contains error identifier ("7" in the case of DS100)
SFB_ERR	SFB error during data transmission using system functions SFB RDREC / WRREC (SFB return value < 0) ➢ The return value is stored in the instance DB (SFB53_RET_VAL or SFB52_RET_VAL). ➢ The receive mail box is deleted if an error occurs when the data are being read. No response data available (SFB return value = 80C0) ➢ The error is reported after the number of job repetitions with the respective number of waiting cycles. ➢ The return value is stored in the instance DB (SFB53_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14..
WDOG_ERR	No plausible response data within the monitoring period (job data of the job and the response do not match)

NOTE:

If the communications link is defective and a job has been initiated, SFB_ERR is not immediately signaled on some CPUs of the S7-300 (older versions!). Instead, the block continues to signal BUSY until either the error is eliminated so that the job is completed or until the cycle monitoring function (default setting: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been rectified, you must restart the job.

Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.

Block call (STL source code)

```
U      Start download
S      " DB_PDAT_DL".START

CALL   PDAT_DL , DB_PDAT_DL(
           LADDR             := DRIVDB1.SLAVE_1.DADDR,
           START_ADDR        := MW10,
           START             := ,
           CANCEL            := M12.1,
           DB_NO             := MW 14,
           DB_NO_LM          := MW 16,
           DB_UNLINKED       := M  12.2,
           LOG_FCT           := M  12.3,
           BUSY              := ,
           DONE              := M  12.5,
           ERROR             := M  13.4,
           WDOG_ERR          := M  12.7,
           SFB_ERR           := M  13.0,
           CFG_ERR           := M  13.1,
           DB_ERR            := M  13.2,
           DB_ERRNO          := MW 20,
           REQ_ERR           := M  12.6,
           ERR_NO            := MW 18,
           NOT_TERMINATED    := M  13.3
           DB_NO_ACT         := MW 22,
           PA_NO             := MW 24)

U      "DB_PDAT_DL".BUSY
R      "DB_PDAT_DL".START
```

Multi-axis drive according to DS 100:

```
U      Start download;
S      " DB_PDAT_DL".START;

CALL       PDAT_DL, DB_PDAT_DL(
           LADDR              :=DRIVDB1.SLAVE_1.SLOT_x.LADDR,
                              // x=any I/O base address of a slot
                              // from the area of the
                              // axis to be addressed
                              // • PIV or empty slot
           START_ADDR         := MW10,
           START              := ,
           CANCEL             := M  12.1,
           DB_NO              := MW 14,
           DB_NO_LM           := MW 16,
           DB_UNLINKED        := M  12.2,
           LOG_FCT            := M  12.3,
           BUSY               := ,
           DONE               := M  12.5,
           ERROR              := M  13.4,
           WDOG_ERR           := M  12.7,
           SFB_ERR            := M  13.0,
           CFG_ERR            := M  13.1,
           DB_ERR             := M  13.2,
           DB_ERRNO           := MW 20,
           REQ_ERR            := M  12.6,
           ERR_NO             := MW 18,
           NOT_TERMINATED     := M  13.3
           DB_NO_ACT          := MW 22,
           PA_NO              := MW 24)

U      "DB_PDAT_DL".BUSY;
R      "DB_PDAT_DL".START;
```

Structure of the parameter job DB (example of complete DB transfer)

See Subsection 7.4.1.1

Structure of the parameter job (example of partial DB transfer)

See Subsection 7.4.1.2

3.2.11 FB PDAT_UD: Up- / Download drive parameters (DS100)

FB 41

The block number can be altered.

This block cannot be used on some series 300 CPUs on which the SFC24 (TEST_DB) is not installed.

In the case of CPUs with MMC memory cards, the section "Data Management in the Loading Memory" must be complied with (FB51 "SHELL_MMC_PDAT_DL").

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task:	OB1
Cyclic interrupt OB:	e.g. OB32

Features

- Can be used for MASTERDRIVES and DC-MASTER
- Download/partial download functionality
- Upload/partial upload functionality
- The parameter DB structure must comply with the "Structure of parameter job DB" definition ..." (see Subsection 7.4.1)
- If the parameter DB is generated by the system ("Convert parameter set to DB" function in the SIMATIC Manager"), the Drive ES Drive Monitor must also be installed. This function can be used only for MASTERDRIVES and DC Master and generates DS100 jobs.

Description of function

The block transfers the parameterizing data of a drive from a data area of the CPU to the drive or reads them back to the CPU. The data can be distributed among several data blocks. The data blocks themselves are either in the working memory or in the loading memory of the CPU. Only write jobs are supported if the download function is activated (READ_EN = 0). The corresponding identifiers are checked by the block. A job with a non-permitted job identifier is ignored and not signaled as an error by the block. The jobs can vary in length. The response data are not stored.

If the upload function is activated (READ_EN = 1), the jobs contained in the parameter DBs are converted to read jobs, thereby allowing previously written parameters to be read back to the CPU (synchronization of DB data with modified drive parameters).

The parameter DB can also contain read jobs which are executed at the same time when the DB is uploaded. In this context, it must be ensured that there is enough space available for the read jobs in the parameter DBs so that the data read back can be stored in the DB. These data cannot be reloaded to the drive as the block does not support read-to-write job conversion.

Please note that parameter values can be read back only if all parameter DBs are stored in the main memory.

The block is also capable of writing or reading back individual sections of the DB(s). This requires a modification to the DB structure (see Subsection 7.4.1.2).

The FB uses the acyclic communication mechanism for the data transfer. The jobs are formulated according to the PIV mechanism in line with the USS protocol specification (data record 100).

The jobs which are requested by different applications are not coordinated. This must be implemented within the applications. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

In the case of multi-axis drives, "download" must be carried out for each axis.

Each individual axis is addressed via the logical basic address (I/O basic address) of any slot (exception: PIV slot or empty slot) from the area of the axis to which data is to be sent:

The block supports the following tasks:

- Request parameter value, simple (word/double word): PIV job identifier 1
- Write parameter value, simple (word/double word): PIV job identifiers 2, 3, 13, 14
- Request parameter value, array (word/double word/several array elements): PIV job identifier 6
- Write parameter value, array (word/double word/several array elements): PIV job identifiers 7, 8, 11, 12

NOTE

If a task with the index 255 is used, the required parameter values must be entered in the task for all indices. Otherwise 0 will be written to the indices for which no parameter value is entered.

Data retention in load memory

Data blocks which are programmed in a source file as part of an STL program can be characterized as "not relevant to the process (key word UNLINKED). This means that, during loading into the CPU, these data blocks are only stored in the load memory. Their content is then copied into the working memory. This function is integrated in block PDAT_UD.

Because data are kept in the load memory, space in the working memory can be saved. The extendable load memory acts as an intermediate memory (e.g. for parameter DB: only the parameter DB which is to be processed is loaded into the working memory).

If the data block is produced with the parameter "UNLINKED", the input "DB_UNLINKED" must be set to "TRUE". At input "DB_NO", the number of the DB in the working memory into which the data is copied is indicated. The number of the first parameter DB in the load memory is at input DB_NO_LM. In this case, however, it is not possible to read the data back from the converter as they are not written back to the load memory again.

If the input "DB_UNLINKED" has been set to "FALSE", the number of the first parameter DB in the working memory must be indicated at the input "DB_NO". The input "DB_NO_LM" is then irrelevant.

The copy DB in the working memory and the data blocks in the load memory must be 8192 bytes in size (see Section 7.7 for example of how to generate a corresponding DB). If the data are only kept in the working memory, the data blocks can be even larger or smaller than 8192 bytes, depending on the CPU.

If several parameter data blocks are copied into the load memory or working memory, the next data block is indicated at the end of each of these data blocks. If no further data block follows, "Number of the next data block" = 0 is to be entered in the last data word.

CPU with MMC memory card:

If a CPU with an MMC memory card is used, the shell block FB51 (SHELL_MMC_PDAT_DL) must be used. FB51 has the same I/O strip as the FB41 and calls it internally. As a result, the call of FB41 can easily be replaced by the call of FB51. FB51 has no KNOW HOW protection. The number of FB41 can therefore be changed at any time and only the call in FB51 has to be adapted.

NOTE:

If you have used the function "Convert parameter set to DB" (for MASTERDRIVES and DC Master only) to create several DBs, and if some or all DB numbers are outside the permissible DB numerical range for the target CPU, you will need to adjust the DB numbers to the CPU-specific range afterwards. In this case, you must also alter the number of the following DB in each DB!

Convert parameter set to DB: available in the SIMATIC Manager > menu options: *Edit > Convert Parameter Set to DB* after Drive ES Basic has been installed

Error logging

With the input "LOG_FCT", the reaction of block PDAT_UD to an error during download (REQ_ERR) is parameterized:

- If the status of the input is "FALSE", download in stopped when the first error occurs. The number of the parameter DB currently being processed is indicated at the output "DB_NO_ACT". The number of the parameter is indicated at the output "PA_NO" and the error number is indicated at output "ERR_NO". In the case of multi-parameter jobs, only the first faulty part of the job is indicated (NOT_TERMINATED = TRUE).
- If the status of the input is "TRUE", download is not stopped when a "REQ_ERR" occurs. The error is logged (log file is in the instance of the DB under sx_LOG...) and downloading is continued. The DB number, the parameter number, the index and the error number are stored in the log. Downloading is only stopped after 20 logged errors (NOT_TERMINATED = TRUE). At the outputs "DB_NO_ACT", "PA_NO" and "ERR_NO", the data of the last error are indicated (number of the parameter DB, parameter number and error number).

Downloading can be stopped at the end of each parameter job with the input "CANCEL" (NOT_TERMINATED = TRUE).

Ignoring SFB errors

In the instance data of the block (sx_PARA_NO. sw_PARA[0] ... sx_PARA_NO. sw_PARA[4]), it is possible to enter whether the SFB error is to be displayed for specific parameters. According to the default setting, these are parameters 970, 971 and 972. They can be altered by the user by means of "Modify variable". It is necessary to enter -1 in the storage cells that are not being used. It only makes sense to use this function if the drive breaks off communication for a short time when these parameters are being written (copying of RAM2ROM, PowerOnReset). In addition, these parameters should be at the end of the jobs to be processed because no check is made to see when the drive is ready for communication again and therefore subsequent jobs report an SFB error again, causing download to be stopped. This also means that each download is only permitted to contain one job from the above list. If several parameters from the above list have to be transferred, it is necessary to resort to the 'partial download' function.

Processing jobs (Download)

1. Enter the number of the first parameter DB into the corresponding interface:
   - If the parameter DB/DBs Is/are in the **working memory**, the number of the first parameter DB is indicated at the input DB_NO (Parameter DB_UNLINKED = 0).
   - If the parameter DB/DBs is/are in the **load memory**, the number of the DB into which the data from the load memory are to be copied is indicated at the input DB_NO. The number of the first parameter DM in the load memory is indicated at input DB_NO_LM (Parameter DB_UNLINKED = 1).
2. Enter the start address (corresponds to the address of the field in which the version identifier is stored) of the first job or partial job at input START_ADDR.
3. Parameterize the relevant drive product range at input DRIVE.
4. Initiate download with the start bit (START).
5. Block checks whether the DB or DBs indicated exist(s) in the CPU (if not: DB_ERR = 1).
6. The block takes the first job from the parameter DB, enters it into the transmit buffer and transmits it to the drive (BUSY = TRUE).
7. The block checks the data received from the drive to see whether:
   - the response identifier matches the job identifier
   - the received parameter number matches the one sent
   - the received parameter index matches the one sent
8. Response in the receive mail box is positive

   If the parameter CANCEL = FALSE, the next job from the parameter DB is taken and transmitted. This is repeated until all jobs have been sent to the drive and processed (DONE = TRUE, BUSY = FALSE).

   In order to stop transmission, the "CANCEL" parameter must be set to "TRUE". If this is the case, transmission is stopped when the parameter being transmitted ends. (NOT_TERMINATED = TRUE). In addition, the current DB number at the output DB_NO_ACT and the number of the last parameter processed are indicated at the output PA_NO.
9. Response in the receive mail box is negative (response identifier = 7dec)

   Job finished with error (REQ_ERR). The error number is located in the parameter value of the response. This error number is indicated in the output parameter ERR_NO.
10. No **plausible** response to the job which was sent:
    - Job is transmitted to the drive again and the received data are checked.
    - The stipulated number of job repetitions has been carried out without a plausible response: watchdog error
11. There are **no** response data after a stipulated number of cycles (SFB signals error 80C0):
    - Job is transmitted to the drive again
    - The stipulated number of job repetitions has been carried out and there are still not response data: SFB error
12. Group error is signaled at the block output ERROR when one of the following errors occurs: REQ_ERR, WDOG_ERR, SFB_ERR, CFG_ERR, or DB_ERR.
13. When the first job is finished, the next job is executed. This process continues until the block encounters the next end identifier (16#EEEE EEEE). It then checks the next word to ascertain whether its value is < > 0 (next DB). If it is, the jobs are executed in this DB, otherwise the download/partial download is ended.

Processing jobs (Upload)

1. Enter the number of the first parameter DB into the corresponding interface:
   - If the parameter DB/DBs is/are in the **working memory**, the number of the first parameter DB is indicated at the input DB_NO (Parameter DB_UNLINKED = 0).
   - If the parameter DB/DBs is/are in the **load memory**, the data cannot be uploaded (DB_ERR = TRUE; DB_ERRNO = 14)
2. Enter the start address (version identifier) of the first job or partial job at input START_ADDR.
3. Set input READ_EN = TRUE and parameterize the relevant drive product range at input DRIVE
4. Initiate upload with the start bit (START).
5. Block checks whether the DB or DBs indicated exist(s) in the CPU (if not: DB_ERR = 1).
6. The block takes the first job from the parameter DB and enters it into the transmit buffer. It then converts it to a read job, or accepts the preformulated read job and transmits it to the drive (BUSY = TRUE). If a write job is converted to a read job, specific parameters are not read back depending on the drive type (see table below).
7. The block checks the data received from the drive to see whether:
   - the response identifier matches the job identifier
   - the received parameter number matches the one sent
   - the received parameter index matches the one sent
8. Response in the receive mail box is positive

   If the parameter CANCEL = FALSE, the next job from the parameter DB is taken and transmitted. This is repeated until all jobs have been sent to the drive and processed (DONE = TRUE, BUSY = FALSE).

   In order to stop transmission, the "CANCEL" parameter must be set to "TRUE". If this is the case, transmission is stopped when the parameter being transmitted ends. (NOT_TERMINATED = TRUE). In addition, the current DB number at the output DB_NO_ACT and the number of the last parameter processed are indicated at the output PA_NO.
9. Response in the receive mail box is negative (response identifier = 7dec)

   Job finished with error (REQ_ERR). The error number is located in the parameter value of the response. This error number is indicated in the output parameter ERR_NO.
10. No **plausible** response to the job which was sent:
    - Job is transmitted to the drive again and the received data are checked.
    - The stipulated number of job repetitions has been carried out without a plausible response: Watchdog error
11. There are **no** response data after a stipulated number of cycles (SFB signals error 80C0):
    - Job is transmitted to the drive again
    - The stipulated number of job repetitions has been carried out and there are still no response data: SFB error
12. Group error is signaled at the block output ERROR when one of the following errors occurs REQ_ERR, WDOG_ERR, SFB_ERR, CFG_ERR or DB_ERR.

13. When the first job is finished, the next job is executed. This process continues until the block encounters the next end identifier (16#EEEE EEEE). It then checks the next word to ascertain whether its value is < > 0 (next DB). If it is, the jobs are executed in this DB, otherwise the download/partial download is ended.

Parameters which are ignored during uploads as a function of drive type

The following table lists the parameters for which write jobs are not converted to read jobs if READ_EN = TRUE.

Drive range	Param. setting at input DRIVE	Parameter number
DC Master	2	51, 927
MASTERDRIVES	1	53, 60, 927

Communications interface

Parameter	Data type	Kind	Description
BUSY	BOOL	OUT	=1: Download in progress (parameters are being transmitted)
CANCEL	BOOL	IN	Stop download (download is stopped after completion of the current parameter job)
CFG_ERR	BOOL	OUT	Slave not configured or configured incorrectly
DB_ERR	BOOL	OUT	Error in parameter DB (for explanation, see DB_ERRNO)
DB_ERRNO	INT	OUT	Error code for DB_ERR 0 = No error occurred 1 = DB indicated does not exist in working memory 2 = DB indicated does not exist in load memory 3 = DB in working memory < > 8192 bytes 4 = Job > 240 bytes 5 = Address of the data to be copied is outside the data DB 7 = Wrong identifier for job start or end identifier 8 = Start address + minimum job length > DB size 9 = Wrong start address (start address + 2 <> data record identifier) 10 = Start address not at word limit or job length an uneven number 11 = Read job without four bytes free for data 13 = Unknown drive selected 14 = Parameters cannot be read back because DBs stored in load memory
DB_NO	INT	IN	DB_UNLINKED = 1: Number of the DB in the working memory into which the data from the load memory are copied. DB_UNLINKED = 0: Number of the first parameter DB in the working memory
DB_NO_ACT	INT	OUT	Number of the current job DB being processed
DB_NO_LM	INT	IN	Number of the first parameter DB in the load memory
DB_UNLINKED	BOOL	IN	=1: Parameters DB(s) is (are) in the load memory
DONE	BOOL	OUT	=1: Download finished without errors (all parameter jobs have been transmitted without error)
DRIVE	INT	IN	=0 all parameters are readable =1 MASTERDRIVES =2 DC-Master

Parameter	Data type	Kind	Description
ERR_NO	INT	OUT	Drive reports back error (from "PWE1" in the case of DS100, from "Error value" in the case of DS47)
ERROR	BOOL	OUT	Group error
LADDR	WORD	IN	Diagnostics address of the slave [1)]
LOG_FCT	BOOL	IN	=1: Errors during download are logged (max. of 20 errors)
NOT_TERMINATED	BOOL	OUT	Download stopped by user ➢ Stopped after first REQ_ERR (LOG_FCT = FALSE) ➢ Stopped after 20ieth REQ_ERR (LOG_FCT = TRUE)
PA_NO	INT	OUT	Number of the last edited parameter
READ_EN	BOOL	IN	=1 Read parameter; =0 write parameter
REQ_ERR	BOOL	OUT	Response contains error identifier ("7" in the case of DS100, "82h" in the case of DS47)
SFB_ERR	BOOL	OUT	SFB 53 WRREC / SFB 52 RDREC signals error
START	BOOL	IN	Start download
START_ADDR	INT	IN	Absolute address as of which the data to be transferred are stored in the DB (corresponds to the address of the field in which the version identifier is stored) (refer to Subsection 7.4.1 Structure of the parameter job DB for FB PDAT_DL / PDAT_UD)
WDOG_ERR	BOOL	OUT	Watchdog error

1) The I/O basic address of any slot (exception: PIV or empty slot) from the area of the axis to be addressed must be specified here for multi-axis drives according to DS 100.

The signaling bits and the data are valid until initiation of the next download. They are deleted when PDAT_DL is started again.

Data area

Parameter	Data type	Kind	Comments
RD	ARRAY [1..240] of BYTE	STAT	Receive mail box for acyclical communication
SD	ARRAY [1..240] of BYTE	STAT	Send mail box for acyclical communication
si_NODATA_CYCLE_NO	INT	STAT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
si_SFB52_RET_VAL	INT	STAT	Return value of SFB 52 RDREC
si_SFB53_RET_VAL	INT	STAT	Return value of SFB 53 WRREC
sx_LOG	ARRAY [0..19] of STRUCT	STAT	Logged data of the parameter jobs containing errors
sx_LOG.si_DB_NO	INT	STAT	DB number of the parameter with the error
sx_LOG.si_INDEX	INT	STAT	Index of the parameter with the error
sx_LOG.si_PARA_NO	INT	STAT	Number of the parameter with the error
sx_LOG.sw_ERR_NO	WORD	STAT	Error code of the parameter with the error
sx_PARA_NO. sw_PARA[0]	WORD	STAT	Parameter numbers for which the SFB error is not evaluated (default: P970, 971, 972). The user can enter additional or different parameter numbers.
sx_PARA_NO. sw_PARA[1]	WORD	STAT	
sx_PARA_NO. sw_PARA[2]	WORD	STAT	
sx_PARA_NO. sw_PARA[3]	WORD	STAT	
sx_PARA_NO. sw_PARA[4]	WORD	STAT	
sy_NODATA_RETRY_NO	BYTE	STAT	Number of job repetitions if no response data have been received (default = 5)
sy_WDOG_RETRY_NO	BYTE	STAT	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	If no configuration data or incorrect configuration data have been entered, a configuration error is signaled. The following data are checked: ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier "8" in the case of DS100)
DB_ERR	Error in parameter DB (for explanation, see DB_ERRNO)
DB_ERRNO	Error code for DB_ERR 0 = No error occurred 1 = DB indicated does not exist in working memory 2 = DB indicated does not exist in load memory 3 = DBs in the working memory < > 8192 bytes 4 = Job > 240 bytes 5 = Address of the data to be copied is outside the data DB 7 = Wrong identifier for job start or end identifier 8 = Start address + minimum job length > DB size 9 = Wrong start address (start address + 2 <> data record identifier) 10 = Start address not at word limit or job length an uneven number 11 = Read job without six bytes free for data 12 = Read job without eight bytes free for data (response identifier 2 or 5) 13 = Unknown drive selected 14 = Parameters cannot be read back because DBs stored in load memory
ERR_NO	Drive reports back error from "PWE1" in the case of DS100 Section 7.2, for description of error see Section "Formulating parameter jobs (data record 100)"
ERROR	Group error
NOT_TERMINATED	Download stopped by user ➢ Stopped after first REQ_ERR (LOG_FCT = FALSE) ➢ Stopped after 20ieth REQ_ERR (LOG_FCT = TRUE)
REQ_ERR	Response container error identifier ("7" in the case of DS100)
SFB_ERR	SFB error during data transmission using system functions SFB RDREC / WRREC (SFB return value < 0) ➢ The return value is stored in the instance DB (SFB53_RET_VAL or SFB52_RET_VAL). ➢ The receive mail box is deleted if an error occurs when the data are being read. No response data available (SFB return value = 80C0) ➢ The error is reported after the number of job repetitions with the respective number of waiting cycles. ➢ The return value is stored in the instance DB (SFB53_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14..
WDOG_ERR	No plausible response data within the monitoring period (job data of the job and the response do not match)

NOTE:

If the communications link is defective and a job has been initiated, SFB_ERR is not immediately signaled on some CPUs of the S7-300 (older versions!). Instead, the block continues to signal BUSY until either the error is eliminated so that the job is completed or until the cycle monitoring function (default setting: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been rectified, you must restart the job.

Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.

Block call (STL source code)

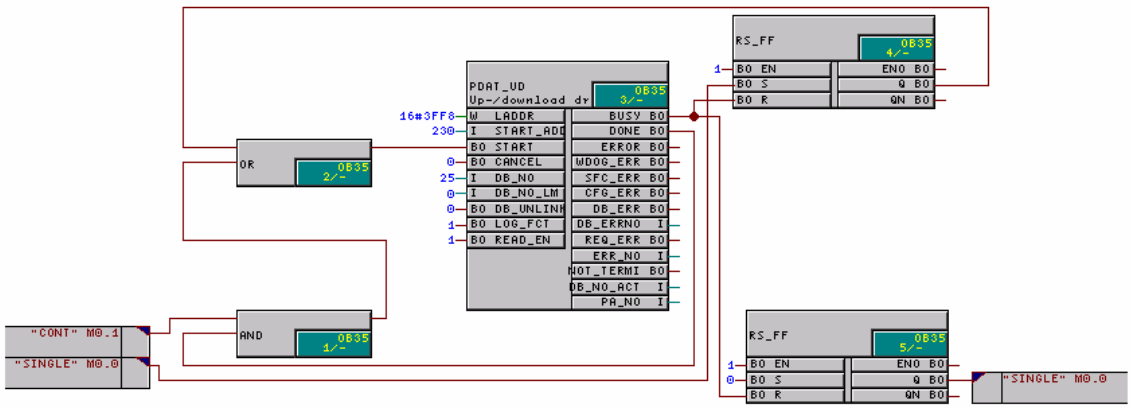
```
U  "START_UPLOAD_DOWNLOAD;
S  "DB_PDAT_UD".START;

CALL "PDAT_UD" , "DB_PDAT_UD" (
      LADDR            := "DRIVDB1".SLAVE_3.DADDR,
      START_ADDR       := MW    20,
      START            := ,
      CANCEL           := M     22.1,
      DB_NO            := MW    24,
      DB_NO_LM         := MW    26,
      DB_UNLINKED      := M     22.2,
      LOG_FCT          := M     22.3,
      READ_EN          := M     22.4,
      DRIVE            := MW    28,
      BUSY             := ,
      DONE             := M     22.6,
      ERROR            := M     22.7,
      WDOG_ERR         := M     23.0,
      SFB_ERR          := M     23.1,
      CFG_ERR          := M     23.2,
      DB_ERR           := M     23.3,
      DB_ERRNO         := MW    30,
      REQ_ERR          := M     23.4,
      ERR_NO           := MW    32,
      NOT_TERMINATED   := M     23.5,
      DB_NO_ACT        := MW    34,
      PA_NO            := MW    36);

U  "DB_PDAT_UD".BUSY;
R  "DB_PDAT_UD".START;
```

Multi-axis drive according to DS 100:

```
U   "START_UPLOAD_DOWNLOAD;
S   "DB_PDAT_UD".START;

CALL "PDAT_UD" , "DB_PDAT_UD" (
      LADDR                    := DRIVDB1.SLAVE_3.SLOT_x.LADDR,
                               // x=any I/O basic address
                               // of a slot from the area of
                               // the axis to be addressed
                               // • PIV- or empty slot,
      START_ADDR       := MW    20,
      START            := ,
      CANCEL           := M     22.1,
      DB_NO            := MW    24,
      DB_NO_LM         := MW    26,
      DB_UNLINKED      := M     22.2,
      LOG_FCT          := M     22.3,
      READ_EN          := M     22.4,
      DRIVE            := MW    28,
      BUSY             := ,
      DONE             := M     22.6,
      ERROR            := M     22.7,
      WDOG_ERR         := M     23.0,
      SFB_ERR          := M     23.1,
      CFG_ERR          := M     23.2,
      DB_ERR           := M     23.3,
      DB_ERRNO         := MW    30,
      REQ_ERR          := M     23.4,
      ERR_NO           := MW    32,
      NOT_TERMINATED   := M     23.5,
      DB_NO_ACT        := MW    34,
      PA_NO            := MW    36);

U   "DB_PDAT_UD".BUSY;
R   "DB_PDAT_UD".START;
```

Read parameters cyclically back from drive

The upload function of the block can be programmed (see logic below) to read parameters cyclically.



Structure of the parameter job (example of complete DB transfer)

See Subsection 7.4.1.1

Structure of the parameter job (example of partial DB transfer)

See Subsection 7.4.1.2

3.2.12 FB PDAT_UD2: Up- / Download drive parameters (DS47)

FB 42

The block number can be altered.

This block cannot be used on some series 300 CPUs on which the SFC24 (TEST_DB) is not installed.

In the case of CPUs with MMC memory cards, the section "Data Management in the Loading Memory" must be complied with (FB52 "SHELL_MMC_PDAT_DL").

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task:	OB1
Cyclic interrupt OB:	e.g. OB32

Features

- Can be used for all drives which support "Parameter access with DPV1" (DS47) in accordance with "PROFIBUS Profile Drive Technology, V3.1, November 2002" (see Section 7.3 “Formulating parameter jobs (data record 47))
- Download/partial download functionality
- Upload/partial upload functionality
- The parameter DB structure must comply with the "Structure of parameter job DB" definition ..." (see Subsection 7.4.1.3).
- The parameter DB should be created by the system via the "Convert parameter set to DB" function in Starter Drive ES. This function can be used for MICROMASTER 4xx and SINAMICS only and generates DS47 jobs.

Description of function

The block transfers the parameterizing data of a drive from a data area of the CPU to the drive or reads them back to the CPU. The data can be distributed among several data blocks. The data blocks themselves are either in the working memory or in the loading memory of the CPU. If the download function is activated (READ_EN = 0), only write jobs and specially identified read jobs (job IDs 2 and 6) are supported). The corresponding job identifiers / job IDs are checked by the block. A job with a non-permitted job identifier / job ID is ignored and not signaled as an error by the block. The jobs can vary in length. The response data are not stored.

If the upload function is activated (READ_EN = 1), the jobs contained in the parameter DBs are converted to read jobs (except for jobs with ID 1 - 6), thereby allowing previously written parameters to be read back to the CPU (synchronization of DB data with modified drive parameters).

The parameter DB can also contain read jobs which are executed at the same time when the DB is uploaded. In this context, it must be ensured that there is enough space available for the read jobs in the parameter DBs so that the data read back can be stored in the DB. These data cannot be reloaded to the drive as the block does not support read-to-write job conversion.

Please note that parameter values can be read back only if all parameter DBs are stored in the main memory.

The block is also capable of writing or reading back individual sections of the DB(s). This requires a modification to the DB structure (see Subsection 7.4.2.2).

The FB uses the acyclic communication mechanism for the data transfer. The jobs are formulated according to the PROFIdrive profile, Version 3 (data record 47).

The number of the data record to be used can be entered via input DS_NO. If the value 0 is assigned to the input, data record 47 is used. In the case of values <> 0, the value at the input is interpreted as the number of the data record to be used. However, the telegram must still be structured in accordance with PROFIdrive Profile Drive Technology, Version 3 as the block will otherwise report errors, i.e. any data record supported by the drive can be used provided the contents of the telegram conform to PROFIdrive Profile Drive Technology, Version 3.

The jobs which are requested by different applications are not coordinated. This must be implemented within the applications. See Subsection 3.2.13 "Locking the blocks with acyclical communication".

On multi-axis drives, the axis is addressed via the "Axis" byte in the parameter job header.

DS47 (PROFIdrive profile)

- Request parameter value, simple (word/double word) job identifier 1
- Write parameter value, simple (word/double word) job identifier 2
- Request parameter value, several array elements (234 max.) job identifier 1
- Write parameter value, several array elements (234 max.) job identifier 2
- Request parameter value, multi-parameter job identifier 1
- Write parameter value, job identifier 2

Data retention in load memory

Data blocks which are programmed in a source file as part of an STL program can be characterized as "not relevant to the process (key word UNLINKED). This means that, during loading into the CPU, these data blocks are only stored in the load memory. Their content is then copied into the working memory. This function is integrated in block PDAT_UD2.

Because data are kept in the load memory, space in the working memory can be saved. The extendable load memory acts as an intermediate memory (e.g. for parameter DB: only the parameter DB which is to be processed is loaded into the working memory).

If the data block is produced with the parameter "UNLINKED", the input "DB_UNLINKED" must be set to "TRUE". At input "DB_NO", the number of the DB in the working memory into which the data is copied is indicated. The number of the first parameter DB in the load memory is at input DB_NO_LM. In this case, however, it is not possible to read the data back from the converter as they are not written back to the load memory again.

If the input "DB_UNLINKED" has been set to "FALSE", the number of the first parameter DB in the working memory must be indicated at the input "DB_NO". The input "DB_NO_LM" is then irrelevant.

The copy DB in the working memory and the data blocks in the load memory must be 8192 bytes in size (see Section 7.7 Tip for example of how to generate a corresponding DB). If the data are only kept in the working memory, the data blocks can be even larger or smaller than 8192 bytes, depending on the CPU.

If several parameter data blocks are copied into the load memory or working memory, the next data block is indicated at the end of each of these data blocks. If no further data block follows, "Number of the next data block" = 0 is to be entered in the last data word.

CPU with MMC memory card:

If a CPU with an MMC memory card is used, the shell block FB52 (SHELL_MMC_PDAT_DL) must be used. FB52 has the same I/O strip as the FB42 and calls it internally. As a result, the call of FB42 can easily be replaced by the call of FB52. FB52 has no KNOW HOW protection. The number of FB42 can therefore be changed at any time and only the call in FB52 has to be adapted.

NOTE:

If the function "Convert parameter set to DB" (for MICROMASTER 4xx and SINAMICS only) has been used to set up several DBs, and if some or all DB numbers are outside the DB number range permitted for the target CPU, you must subsequently adjust the DB numbers to the range specifically permitted for the CPU. In this case, you must also alter the number of the following DB in each DB!

Convert parameter set to DB: available after installation of Drive ES basic STARTER in the SIMATIC Manager > open "STARTER" with double click on drive > create expert list > click button "Convert parameter set to DB"

Error logging

With the input "LOG_FCT", the reaction of block PDAT_UD2 to an error during download (REQ_ERR) is parameterized:

- If the status of the input is "FALSE", download in stopped when the first error occurs. The number of the parameter DB currently being processed is indicated at the output "DB_NO_ACT". The number of the parameter is indicated at the output "PA_NO" and the error number is indicated at output "ERR_NO". In the case of multi-parameter jobs, only the first faulty part of the job is indicated (NOT_TERMINATED = TRUE).
- If the status of the input is "TRUE", download is not stopped when a "REQ_ERR" occurs. The error is logged (log file is in the instance of the DB under sx_LOG...) and downloading is continued. The DB number, the parameter number, the index and the error number are stored in the log. Downloading is only stopped after 20 logged errors (NOT_TERMINATED = TRUE). At the outputs "DB_NO_ACT", "PA_NO" and "ERR_NO", the data of the last error are indicated (number of the parameter DB, parameter number and error number). In the case of multi-parameter jobs, each faulty part of the job is entered in the download log.

Downloading can be stopped at the end of each parameter job with the input "CANCEL" (NOT_TERMINATED = TRUE).

Ignoring SFB errors

In the instance data of the block (sx_PARA_NO. sw_PARA[0] ... sx_PARA_NO. sw_PARA[4]), it is possible to enter whether the SFB error is to be displayed for specific parameters. According to the default setting, these are parameters 970, 971 and 972. They can be altered by the user by means of "Modify variable". It is necessary to enter -1 in the storage cells that are not being used. It only makes sense to use this function if the drive breaks off communication for a short time when these parameters are being written (copying of RAM2ROM, PowerOnReset). In addition, the write jobs for these parameters should be at the end of the jobs to be processed because no check is made to see when the drive is ready for communication again and therefore subsequent jobs report an SFB error again, causing download to be stopped. This also means that each download is only permitted to contain one job from the above list. If several parameters from the above list have to be transferred, it is necessary to resort to the 'partial download' function. In the case of the job "Store parameters in the EEPROM", which is generated by the tool "Convert parameter set into DB", however, the subsequent read job as to whether storage was successful is also executed and time-monitored.

Processing orders (Download)

1. Enter the number of the first parameter DB into the corresponding interface:
   - If the parameter DB/DBs Is/are in the **working memory**, the number of the first parameter DB is indicated at the input DB_NO (Parameter DB_UNLINKED = 0).
   - If the parameter DB/DBs is/are in the **load memory**, the number of the DB into which the data from the load memory are to be copied is indicated at the input DB_NO. The number of the first parameter DM in the load memory is indicated at input DB_NO_LM (Parameter DB_UNLINKED = 1).
2. Enter the start address (corresponds to the address of the field in which the version identifier is stored) of the first job or partial job at input START_ADDR.
3. Parameterize the relevant drive product range at input DRIVE.
4. Initiate download with the start bit (START).
5. Block checks whether the DB or DBs indicated exist(s) in the CPU (if not: DB_ERR = 1).
6. The block takes the first job from the parameter DB. It reads the job ID, enters it in the transmit buffer and transfers it to the drive according to the drive ID (BUSY = TRUE).
7. The block checks the data received from the drive:
   - Response reference (reflected job reference)
   - Special jobs have been executed without error and received data match the comparison values
8. Response in the receive mail box is positive

   If the parameter CANCEL = FALSE, the next job from the parameter DB is taken and transmitted. This is repeated until all jobs have been sent to the drive and processed (DONE = TRUE, BUSY = FALSE).

   In order to stop transmission, the "CANCEL" parameter must be set to "TRUE". If this is the case, transmission is stopped when the parameter being transmitted ends. (NOT_TERMINATED = TRUE). In addition, the current DB number at the output DB_NO_ACT and the number of the last parameter processed are indicated at the output PA_NO.
9. Response in the receive mail box is negative (response identifier = 82hex)

   Job finished with error (REQ_ERR). The error number is located in the parameter value of the response or several error numbers are located in the partial jobs (DS47, multi-parameter). This error number is indicated in the output parameter ERR_NO. In the case of a multi-parameter job (DS47), the error number of the first part of the job containing the error is indicated here. If the logging function is active, all parts of the job which have errors are logged.
10. No **plausible** response to the job which was sent:
    - Job is transmitted to the drive again and the received data are checked.
    - The stipulated number of job repetitions has been carried out without a plausible response: watchdog error

11. There are **no** response data after a stipulated number of cycles (SFB signals error 80C0):
    - Job is transmitted to the drive again
    - The stipulated number of job repetitions has been carried out and there are still not response data: SFB error
12. Group error is signaled at the block output ERROR when one of the following errors occurs: REQ_ERR, WDOG_ERR, SFB_ERR, CFG_ERR, or DB_ERR.
13. When the first job is finished, the next job is executed. This process continues until the block encounters the next end identifier (16#EEEE EEEE). It then checks the next word to ascertain whether its value is < > 0 (next DB). If it is, the jobs are executed in this DB, otherwise the download/partial download is ended.

Processing jobs (Upload)

1. Enter the number of the first parameter DB into the corresponding interface:
    - If the parameter DB/DBs Is/are in the **working memory**, the number of the first parameter DB is indicated at the input DB_NO (Parameter DB_UNLINKED = 0).
    - If the parameter DB/DBs is/are in the **load memory**, the data cannot be uploaded (DB_ERR = TRUE; DB_ERRNO = 14)
2. Enter the start address (version identifier) of the first job or partial job at input START_ADDR.
3. Set input READ_EN = TRUE and parameterize the relevant drive range at input DRIVE
4. Initiate download with the start bit (START).
5. Block checks whether the DB or DBs indicated exist(s) in the CPU (if not: DB_ERR = 1).
6. The block takes the first job from the parameter DB, enters it into the transmit buffer. It then converts it to a read job, or accepts the preformulated read job and transmits it to the drive (BUSY = TRUE). Write jobs with special identifier are not converted to a read job (job ID 1–6) and are thus skipped without error message.
7. The block checks the data received from the drive:
    - Response reference (reflected job reference)
    - Special jobs have been executed without error and received data match the comparison values
8. Response in the receive mail box is positive

   If the parameter CANCEL = FALSE, the next job from the parameter DB is taken and transmitted. This is repeated until all jobs have been sent to the drive and processed (DONE = TRUE, BUSY = FALSE).

   In order to stop transmission, the "CANCEL" parameter must be set to "TRUE". If this is the case, transmission is stopped when the parameter being transmitted ends. (NOT_TERMINATED = TRUE). In addition, the current DB number at the output DB_NO_ACT and the number of the last parameter processed are indicated at the output PA_NO.

9. Response in the receive mail box is negative (response identifier = 82hex)

   Job finished with error (REQ_ERR). The error number is located in the parameter value of the response or several error numbers are located in the partial jobs (DS47, multi-parameter). This error number is indicated in the output parameter ERR_NO. In the case of a multi-parameter job (DS47), the error number of the first part of the job containing the error is indicated here. If the logging function is active, all parts of the job which have errors are logged.

10. No **plausible** response to the job which was sent:
    - Job is transmitted to the drive again and the received data are checked
    - The stipulated number of job repetitions has been carried out without a plausible response: Watchdog error

11. There are **no** response data after a stipulated number of cycles (SFB signals error 80C0):
    - Job is transmitted to the drive again
    - The stipulated number of job repetitions has been carried out and there are still no response data: SFB error

12. Group error is signaled at the block output ERROR when one of the following errors occurs REQ_ERR, WDOG_ERR, SFB_ERR, CFG_ERR or DB_ERR.

13. When the first job is finished, the next job is executed. This process continues until the block encounters the next end identifier (16#EEEE EEEE). It then checks the next word to ascertain whether its value is < > 0 (next DB). If it is, the jobs are executed in this DB, otherwise the download/partial download is ended.

Communications interface

Parameter	Data type	Kind	Description
BUSY	BOOL	OUT	=1: Download in progress (parameters are being transmitted)
CANCEL	BOOL	IN	Stop download (download is stopped after completion of the current parameter job)
CFG_ERR	BOOL	OUT	Slave not configured or configured incorrectly
DB_ERR	BOOL	OUT	Error in parameter DB (for explanation, see DB_ERRNO)
DB_ERRNO	INT	OUT	Error code for DB_ERR 0 = No error occurred 1 = DB indicated does not exist in working memory 2 = DB indicated does not exist in load memory 3 = DB in working memory < > 8192 bytes 4 = Job > 240 bytes 5 = Number of parameters < 1 or > 39 (DS47) 6 = Number of elements < 0 or > 234 (DS47) 7 = Wrong identifier for job start or end identifier 8 = Start address + minimum job length > DB size 9 = Wrong start address (start address + 2 <> data record identifier) 10 = Start address not at word limit or job length an uneven number 11 = Read job without four bytes free for data 12 = Address of the data to be copied is outside the data DB 14 = Parameters cannot be read back because DBs stored in load memory 15 = Number of parameters/elements with units system > 2 or < 1 16 = Incorrect units system 17 = Parameter not saved to EEPROM 18 = Write job (save parameter(s) to EEPROM) not followed by read job 19 = Download operation is not terminated after transfer of parameters to EEPROM 20 = Download not started with first DL-DB
DB_NO	INT	IN	DB_UNLINKED = 1: Number of the DB in the working memory into which the data from the load memory are copied. DB_UNLINKED = 0: Number of the **first** parameter DB in the working memory
DB_NO_ACT	INT	OUT	Number of the current job DB being processed
DB_NO_LM	INT	IN	Number of the **first** parameter DB in the load memory
DB_UNLINKED	BOOL	IN	=1: Parameters DB(s) is (are) in the load memory
DONE	BOOL	OUT	=1: Download finished **without** errors (all parameter jobs have been transmitted without error)
DS_NO	WORD	IN	Number of data record to be read out
ERR_NO	INT	OUT	Drive reports back error
ERROR	BOOL	OUT	Group error
LADDR	WORD	IN	Diagnostics address of the slave
LOG_FCT	BOOL	IN	=1: Errors during download are logged (20 errors max.)
NOT_TERMINATED	BOOL	OUT	Download stopped by user ➢ Stopped after first REQ_ERR (LOG_FCT = FALSE) ➢ Stopped after 20ieth REQ_ERR (LOG_FCT = TRUE)

Parameter	Data type	Kind	Description
PA_NO	INT	OUT	Number of the last edited parameter
READ_EN	BOOL	IN	=1 Read parameter; =0 write parameter
REQ_ERR	BOOL	OUT	Response contains error identifier"82h"
SFB_ERR	BOOL	OUT	SFB 53 WRREC / SFB 52 RDREC signals error
START	BOOL	IN	Start download
START_ADDR	INT	IN	Absolute address as of which the data to be transferred are stored in the DB (corresponds to the address of the field in which the version identifier is stored) (see Subsection: 7.4.1.3)
WDOG_ERR	BOOL	OUT	Watchdog error

The signaling bits and the data are valid until initiation of the next download. They are deleted when PDAT_DL is started again.

Data area

Parameter	Data type	Kind	Comments
RD	ARRAY [1..240] of BYTE	STAT	Receive mail box for acyclical communication
SD	ARRAY [1..240] of BYTE	STAT	Send mail box for acyclical communication
si_NODATA_CYCLE_NO	INT	STAT	Number of cycles in wait period for arrival of response data before the job is repeated (default = 2500)
si_SFB52_RET_VAL	INT	STAT	Return value of SFB 52 RDREC
si_SFB53_RET_VAL	INT	STAT	Return value of SFB 53 WRREC
sx_LOG	ARRAY [0..19] of STRUCT	STAT	Logged data of the parameter jobs containing errors
sx_LOG.si_DB_NO	INT	STAT	DB number of the parameter with the error
sx_LOG.si_INDEX	INT	STAT	Index of the parameter with the error
sx_LOG.si_PARA_NO	INT	STAT	Number of the parameter with the error
sx_LOG.sw_ERR_NO	WORD	STAT	Error code of the parameter with the error
sx_PARA_NO. sw_PARA[0]	WORD	STAT	Parameter numbers for which the SFB error is not evaluated (default: P970, 971, 972). The user can enter additional or different parameter numbers.
sx_PARA_NO. sw_PARA[1]	WORD	STAT	
sx_PARA_NO. sw_PARA[2]	WORD	STAT	
sx_PARA_NO. sw_PARA[3]	WORD	STAT	
sx_PARA_NO. sw_PARA[4]	WORD	STAT	
sy_NODATA_RETRY_NO	BYTE	STAT	Number of job repetitions if no response data have been received (default = 5)
sy_WDOG_RETRY_NO	BYTE	STAT	Number of job repetitions if no plausible response data have been received (default = 5)

Error reactions

The block contains the following error displays:

Output	Description of error
CFG_ERR	If no configuration data or incorrect configuration data have been entered, a configuration error is signaled. The following data are checked: ➢ Logical basic address = 0 ➢ No parameter setting authorization on drive (response identifier "0Bh")
DB_ERR	Error in parameter DB Parameter-DB (for explanation, see DB_ERRNO)
DB_ERRNO	Error code for DB_ERR 0 = No error occurred 1 = DB indicated does not exist in working memory 2 = DB indicated does not exist in load memory 3 = DBs in the working memory < > 8192 bytes 4 = Job > 240 bytes 5 = Number of parameters < 1 or > 39 (DS47) 6 = Number of elements < 0 or > 234 (DS47) 7 = Wrong identifier for job start or end identifier 8 = Start address + minimum job length > DB size 9 = Wrong start address (start address + 2 <> data record identifier) 10 = Start address not at word limit or job length an uneven number 11 = Read job without four bytes free for data 12 = Address of the data to be copied is outside the data DB 14 = Parameters cannot be read back because DBs stored in load memory 15 = No. of parameters/elements with units system > 2 or < 1 16 = Incorrect units system 17 = Parameter not saved to EEPROM 18 = Write job (save parameter(s) to EEPROM) not followed by read job 19 = Download operation is not terminated after transfer of parameters to EEPROM 20 = Download not started with first DL-DB
ERR_NO	➢ Drive reports back error. For description of error see Section 7.3 "Formulating parameter jobs (data record 47)"
ERROR	Group error
NOT_TERMINATED	Download stopped by user ➢ Stopped after first REQ_ERR (LOG_FCT = FALSE) ➢ Stopped after 20ieth REQ_ERR (LOG_FCT = TRUE)
REQ_ERR	Response container error identifier "82h"
SFB_ERR	SFB error during data transmission using system functions SFB RDREC / WRREC (SFB return value < 0) ➢ The return value is stored in the instance DB (SFB53_RET_VAL or SFB52_RET_VAL). ➢ The receive mail box is deleted if an error occurs when the data are being read. No response data available (SFB return value = 80C0) ➢ The error is reported after the number of job repetitions with the respective number of waiting cycles. ➢ The return value is stored in the instance DB (SFB53_RET_VAL). The meaning of the return value can be found in the online help for the SFB or Section 7.3 "Formulating parameter jobs (data record 47)", Table 14.
WDOG_ERR	No plausible response data within the monitoring period (job data of the job and the response do not match)

NOTE:

If the communications link is defective and a job has been initiated, SFB_ERR is not immediately signaled on some CPUs of the S7-300 (older versions!). Instead, the block continues to signal BUSY until either the error is eliminated so that the job is completed or until the cycle monitoring function (default setting: NODATA_CYCLE_NO x NODATA_RETRY_NO = 12500 cycles!) responds and signals SFB_ERR. Once SFB_ERR has been rectified, you must restart the job.

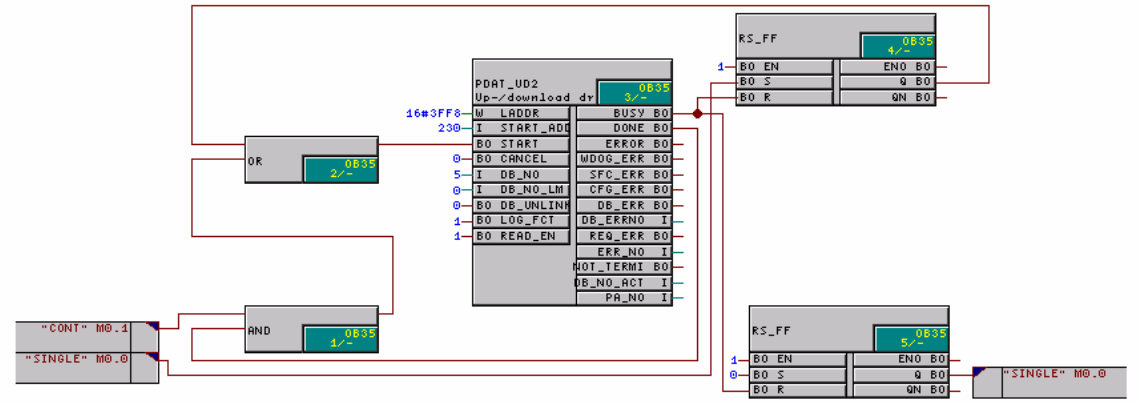
Whatever the circumstances, however, FC COM_STAT (FC60) always displays a communication breakdown immediately.

Block call (STL source code)

```
U  "START_UPLOAD_DOWNLOAD;
S  "DB_PDAT_UD2".START;

CALL "PDAT_UD2" , "DB_PDAT_UD2" (
      LADDR            := "DRIVDB1".SLAVE_3.DADDR,
      DS_NO            .= MW    38,
      START_ADDR       := MW    20,
      START            := ,
      CANCEL           := M     22.1,
      DB_NO            := MW    24,
      DB_NO_LM         := MW    26,
      DB_UNLINKED      := M     22.2,
      LOG_FCT          := M     22.3,
      READ_EN          := M     22.4,
      BUSY             := ,
      DONE             := M     22.6,
      ERROR            := M     22.7,
      WDOG_ERR         := M     23.0,
      SFB_ERR          := M     23.1,
      CFG_ERR          := M     23.2,
      DB_ERR           := M     23.3,
      DB_ERRNO         := MW    30,
      REQ_ERR          := M     23.4,
      ERR_NO           := MW    32,
      NOT_TERMINATED   := M     23.5,
      DB_NO_ACT        := MW    34,
      PA_NO            := MW    36);

U  "DB_PDAT_UD2".BUSY;
R  "DB_PDAT_UD2".START;
```

Read parameters cyclically back from drive

The upload function of the block can be programmed (see logic below) to read parameters cyclically.



Structure of the parameter job (example of complete DB transfer)

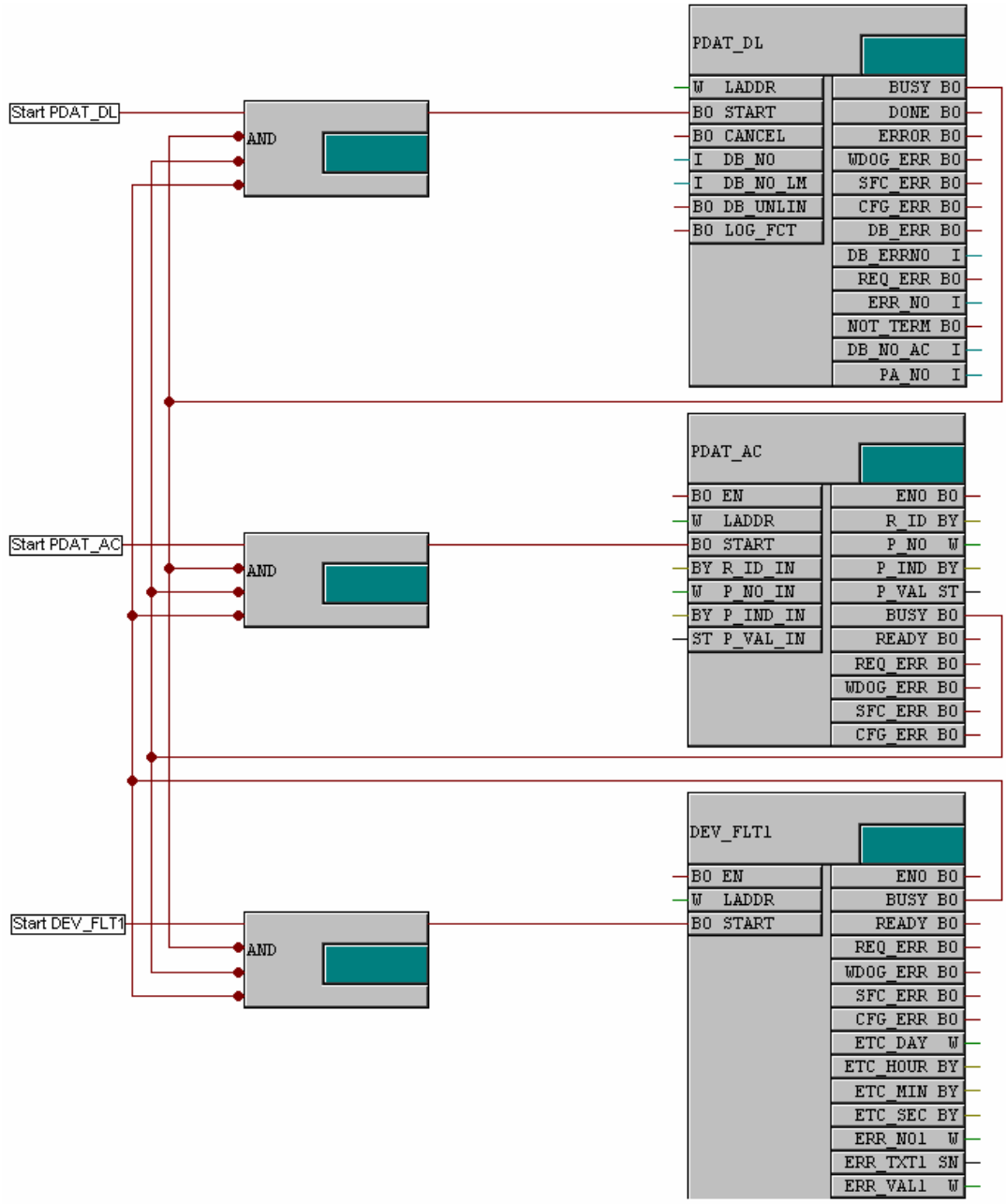
See Subsection 7.4.2.1

Structure of the parameter job (example of partial DB transfer)

See Subsection 7.4.2.2

3.2.13 Locking the blocks with acyclical communications

Because the acyclical communication link between the S7-CPU and the drive can be used by only one application at a time, several applications with acyclical communications—as shown in the following example—must be "locked" against each other.



NOTE

The start bit must be reset as soon as the BUSY output of the activated block is "TRUE".

Otherwise, the corresponding block will always be initiated again (BUSY has the value "FALSE" when a job has been completed).

3.3 Functions

3.3.1 FC COM_STAT: Signal communications fault

FC 60

The block number can be altered.

Calling OBs

The block can be incorporated into one of the following OBs:

Cyclical task: OB1

Cyclic interrupt OB: e.g. OB32

Description of function

Supported by system function SFC 51 RDSYSST, the block evaluates the system status list of the CPU and signals whether the slave to be processed is faulty or deactivated. It must be called once for each slave to be activated.

Bit "COM_FLT" is set to signal that the slave can no longer be addressed by the master.

Slaves can be activated or deactivated by SFC12 while the plant is in operation. When a slave is deactivated, no error messages are generated, no LEDs light up and telegram repetitions avoided. The "deactivated" status is read by the block and signaled in bit "DEACTIV".

Communications interface

Parameter	Declaration	Data type	Memory area	Description
COM_FLT	OUTPUT	BOOL	I, Q, M, D, L	DP slave failed
DEACTIV	OUTPUT	BOOL	I, Q, M, D, L	DP slave deactivated
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Diagnostic address of slave
SFC_ERR	OUTPUT	BOOL	I, Q, M, D, L	SFC 51 RDSSYST signals error
SFC_FLT	OUTPUT	INT	I, Q, M, D, L	Return value of SFC 51 RDSSYST

Block call (STL source code)

```
CALL       COM_STAT(
           LADDR         := DRIVDB1.SLAVE_1.DADDR,
           COM_FLT       := M30.0,
           DEACTIV       := M30.1,
           SFC_ERR       := M30.2,
           SFC_FLT       := MW32);
```

3.4 Data blocks

3.4.1 DB DRIVDBx: Configuration data of drives

You must provide a data block DB DRIVDBx (x = serial number of configuration DBs) to be able to transfer configuration data to communication blocks from Section 3.1. You must enter all values in hexadecimal format.

Parameter	Declaration	Data type	Description
SLAVE_n	STAT	STRUCT	Slave-specific configuration data Slave n
DPADDR	STAT	WORD	PROFIBUS address/device number of slave (from HW configuration, not evaluated by blocks)
DADDR	STAT	WORD	Diagnostic address of slave (from HW configuration)
SLOT_m	STAT	SLOT_UDT	Slot-specific configuration data Slot m (UDT 31, see below)
...			
SLOT_m+y	STAT	SLOT_UDT	Slot-specific configuration data Slot m + y (UDT 31, see below)
...			
SLAVE_n+1	STAT	STRUCT	Slave-specific configuration data Slave n + 1
DPADDR	STAT	WORD	PROFIBUS address/device number of slave (from HW configuration, not evaluated by blocks)
DADDR	STAT	WORD	Diagnostic address of slave (from HW configuration)
SLOT_m	STAT	SLOT_UDT	Slot-specific configuration data Slot m (UDT 31, see below)
...			
SLOT_m+y	STAT	SLOT_UDT	Slot-specific configuration data Slot m + y (UDT 31, see below)
...			
SLAVE_n+x	STAT	STRUCT	Slave-specific configuration data Slave n + x
DPADDR	STAT	WORD	PROFIBUS address/device number of slave (from HW configuration, not evaluated by blocks)
DADDR	STAT	WORD	Diagnostic address of slave (from HW configuration)
SLOT_m	STAT	SLOT_UDT	Slot-specific configuration data Slot m (UDT 31, see below)
...			
SLOT_m+y	STAT	SLOT_UDT	Slot-specific configuration data Slot m + y (UDT 31, see below)

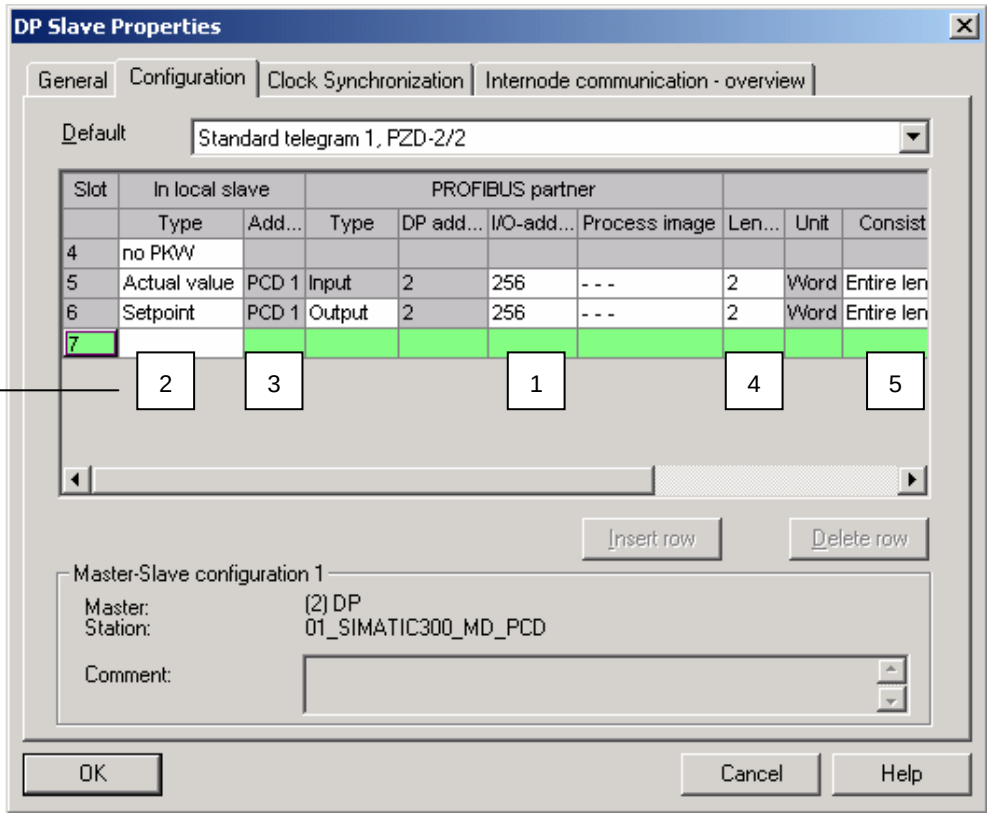
The data block utilizes the user-defined data type SLOT_UDT:

UDT 31 (data type for slot-specific configuration data)
The number of the UDT must **not** be changed.

Parameter	Data type	Description
LADDR 1	WORD	I/O basic address of slot (from HW configuration > PROFIBUS/PROFINET IO partner > I/O address)
SLOT_ID 2	BYTE	Identifier for slot type (indirectly from HW configuration > Drive > Type) 0 = Slot not assigned 1 = Setpoint slot 2 = Actual value slot 3 = Combined setpoint/actual value slot 4 = PIV slot
PCD_ADDR [1] 3	BYTE	Process data offset address for setpoint and/or actual value slot (from HW configuration > Drive > Address; Value range 1 ... 10h).
LENGTH [1] 4	BYTE	Length in words of slot data to be transferred (from HW Configuration > Length; Value range 1...10h).
CONSIST [1] 5	BOOL	Consistency (from HW configuration > Consistency): 0 (FALSE) = Unit 1 (TRUE) = Total length

1) Irrelevant for slot ID = 3, 4

The number of slaves for which configuration data can be stored in one DRIVDBx is dependent on the slave configuration (number of slots per slave) and on the CPU used.



3.4.2 Easy generation of data block DRIVDBx

The tool "Drive ES - Generate DRIVDBx" is a simple method for program-supported generation of the DRIVDBx data block. Repeated entry of the same data is avoided and the risk of errors is reduced. In addition, the programmer needs much less time for implementation.

Function

The tool "Drive ES - Generate DRIVDBx" reads the configuration data of all the drives of a DP or IO subnetwork of a SIMATIC station and uses the data to produce one or more STL sources in the source container of the corresponding project. These sources are then compiled and one or more data blocks of type DRIVDBx are available, which is necessary for use of the basic blocks from Section 3.1.

Installation

The tool is installed during the framework setup of Drive ES SIMATIC.

When setup is being carried out, a message can appear saying that the file IXLocalGlobal.tlb already exists in another version. This is the case if an application has already been installed which also needs this file. You must then confirm that you want to keep the existing version.

Preconditions

An S7 project (STEP 7, from Version 5.0 / Service Pack 3) has already been created with the following conditions:

- Hardware configuration with drive slaves has been carried out and translated.
- Blocks from the library Drive ES - DRVDPS7 have been inserted into the S7 program, specially the UDT31 (preconditions for translation of the DB source)
- Version V5.1 SP3 of STEP 7 will be needed to be able to generate a DRIVDBx from the hardware configuration of a PC station.

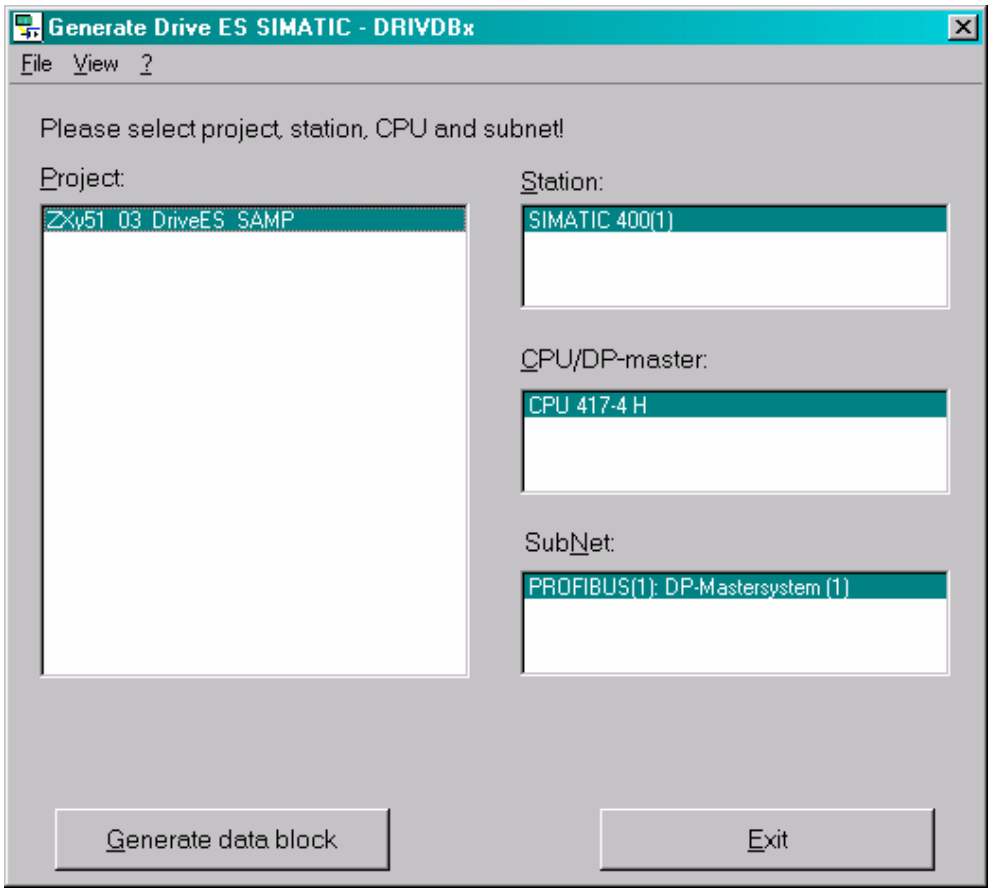
Starting

The tool is started by means of the start bar "SIMATIC\STEP 7\Drive ES - Generate DRIVDBx". After the tool has been started, the main window appears.

The starting screen can be shown or hidden by means of the menu View -> Display start screen when the tool is started.

Main window

After the tool has been started, the following window appears.



Project: In this field, all the available S7 projects are displayed.

Station: In this field, all the available stations of the selected project are displayed.

CPU/DP-Master: In this field, all the DP masters / IO controllers of the selected station are indicated.

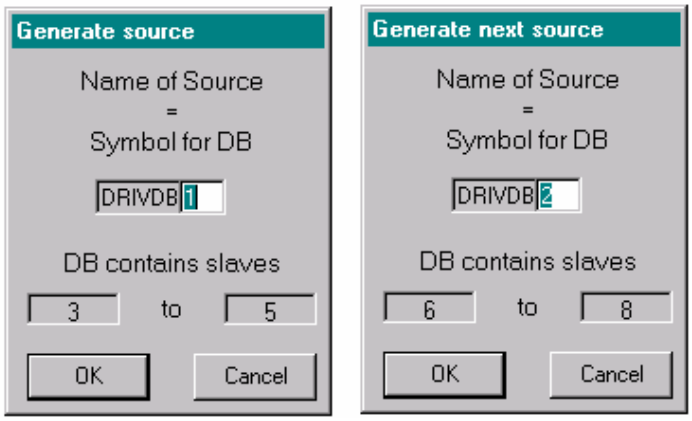
SubNet: Here, all the subnetworks are shown which are available in the selected DP master / IO controller.

The DB generator is started via menu options File → Generate Data Block or the Generate Data Block button if the project name, station name, CPU and subnet are selected.

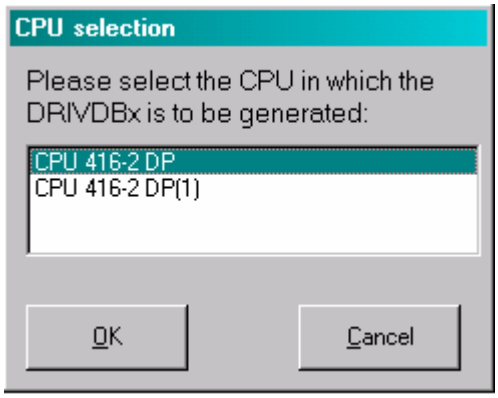
If there are no drive slaves in the selected subnetwork, a source is not generated. The tool must be started once per station, CPU and subnetwork.

If there are drive slaves, the name of the STL source is requested in the next screen form. The name of the STL source matches the data block symbol entered in the symbol table. In addition, which slaves will be retained in the generated data block is shown.

In the case of S7-300, the length of the data block(s) to be generated is limited to 8KB.

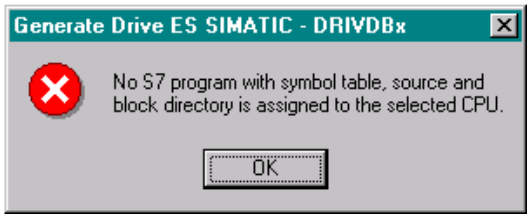


If there are an IM 467, CP 443-5, CP443-1 and/or CP343-1 and at least two CPUs in the selected station, the user must decide which CPU the DrivDBx is to be generated in.

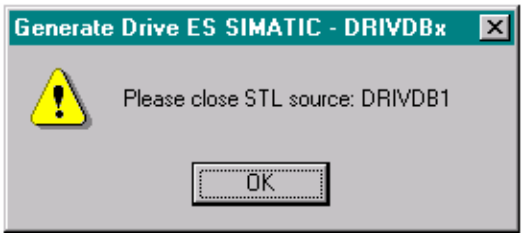


A check is made to see whether the source already exists. You can overwrite the existing source or enter a new name for the source.

Before the source is generated, a check is made to determine whether an S7 program with source, block container and symbol table is assigned to the selected CPU…

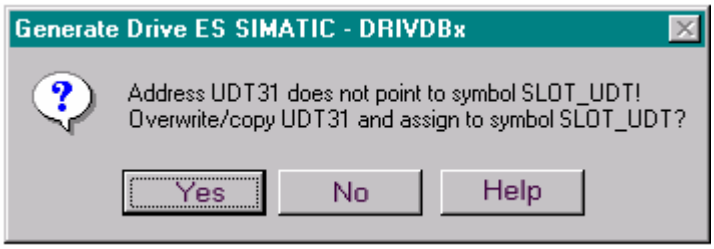


…or whether the source to be overwritten is still open.

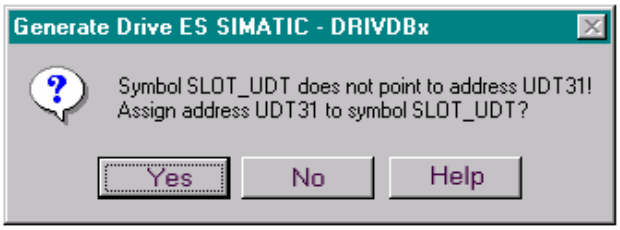


After the source has been generated, the program checks whether the block folder of the selected CPU contains UDT31 (SLOT_UDT). If not, it is copied to the folder from library DRVDPS7.

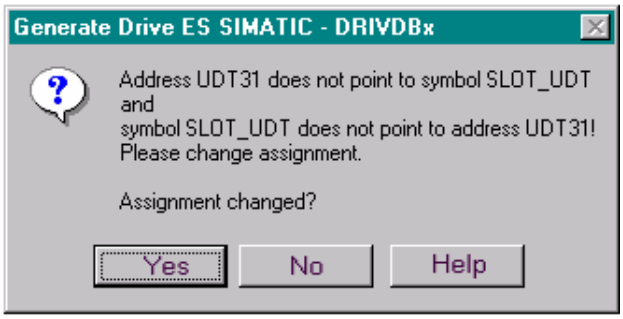
If the project contains UDT31, but it is not assigned to symbol SLOT_UDT, the following message box appears:



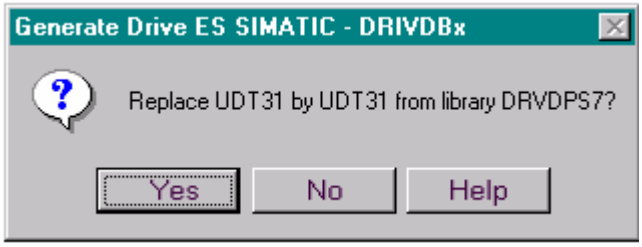
If symbol SLOT_UDT is already assigned to another address in the symbol table, the following message box appears.



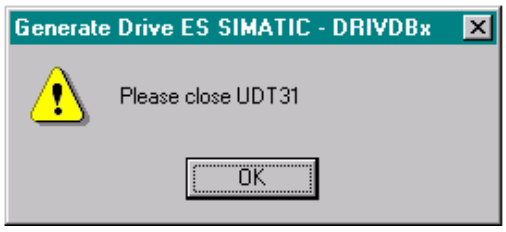
If both UDT31 and the SLOT_UDT symbol are already assigned in the project, you must correct the assignment or delete the entries from the symbol table. The following message box is displayed to remind you:



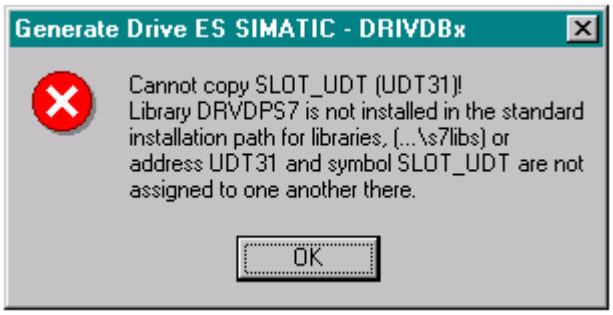
If the SLOT_UDT symbol is assigned to address UDT31 in the symbol table after the above message is displayed, a message indicating that UDT31 will be overwritten after the symbol table has been checked again:

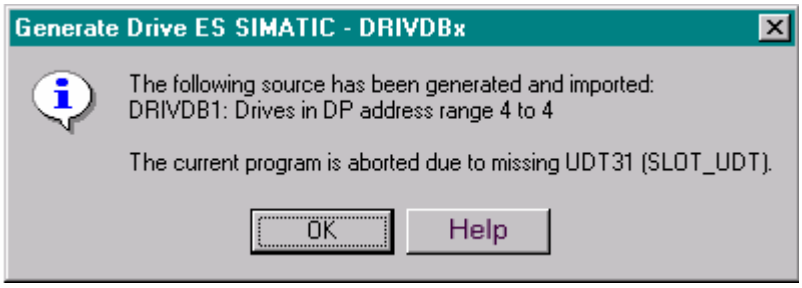


If UDT31 is open in the LAD, STL or CSF editor, the following message appears:

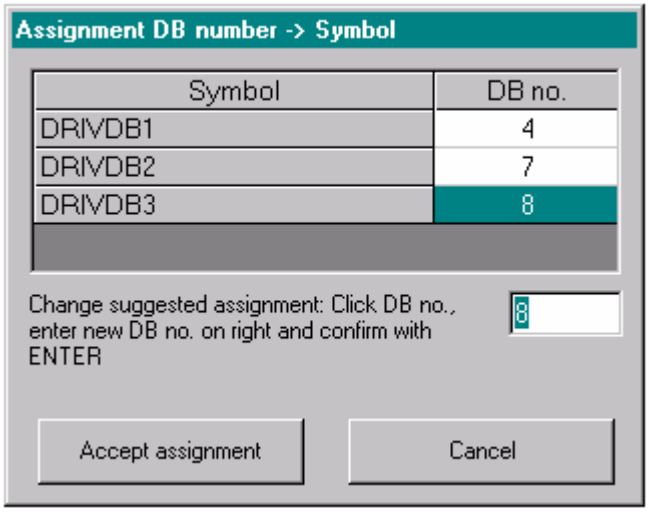


If library DRVDPS7 is not installed in the standard installation path for libraries (...\s7libs), or it does not contain UDT31 (SLOT_UDT), the following window appears and the data block(s) is (are) not generated. The program is then terminated.

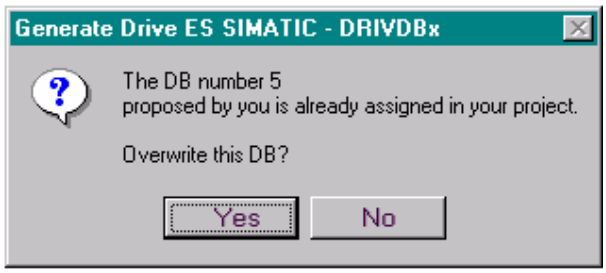




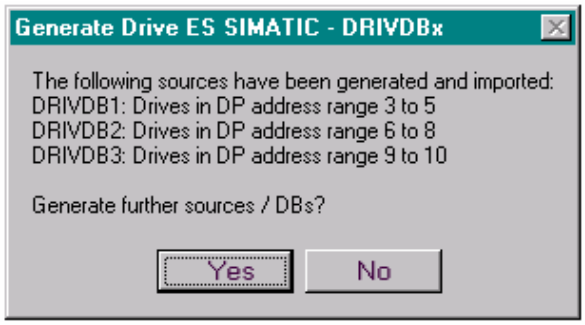
When the generation and copying processes are finished, a window appears showing the free data block number assigned to the new sources.



You can accept the assignment, enter your own free data block number or cancel the generation of a data block (Cancel). The assignment you specify is checked again for plausibility. If the check determines that the specified DB number is already assigned, the following message box appears:

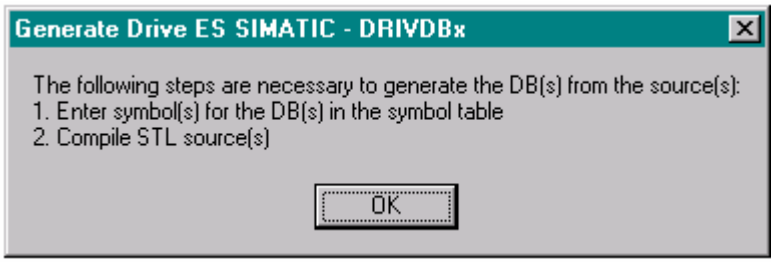


If you do not want to create a data block, a window containing a list of all generated and imported sources is displayed:



If you want to create further sources, the program branches back to the main window.

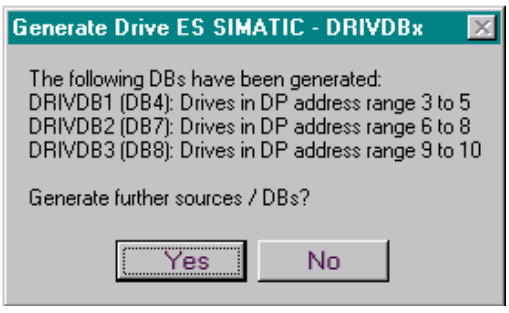
If you choose to exit the program, a box appears showing what steps you need to take next.



Further steps:

- Enter symbol(s) for data block(s) in the symbol table
- Compile STL source(s)

If you have selected the creation of a data block, the LAD, STL or CSF editor is opened and the source compiled. Once it has been compiled, a window listing all generated data blocks is displayed.



Whenever a symbol is entered in the symbol table or a source is compiled, a log file showing the result of the entry or the compilation is created.

This log file is named "DB_GEN_LOG.txt“ and is archived in the installation directory of the DRIVDBx generator.

NOTES

- You can abort the display of all available projects as it is building at any time by pressing the "ESC" key.
- The name of the CPU must contain the following sequence of characters: "CPU", "WinLC" or "WinAC"(ensure correct upper case/lower case letters).
- If an IM 467 is used, its name must also contain the sequence of characters "IM 467". The same applies to CP 443-5, CP 443-1 or CP 343-1 except that here the sequence of characters "CP 443-5", "CP 443-1" or "CP 343-1" must be contained in the name.
- The names of the integrated interface modules of the CPUs must contain the sequence of characters "DP", "CP 5" or "PN-IO" if they are used as the DP master/IO controller.
- The tool does not evaluate
  - Multi-axis slaves if no axis separator is configured between axes
  - Norm slaves with different I/O addresses per slot for PROFIBUS
  - Drive slaves managed by a CP342-5
- If "No" default was selected during hardware configuration and if "1 word" or "2 words" was entered in the parameter "Consistency = Total length" for a setpoint slot or actual-value slot, a consistency "Unit" is stored in the generated DRIVDBx. This apparently incorrect behavior of the tool has no effect on the function of the blocks. It behaves in the same way when PPO types 1 and 3 are selected.

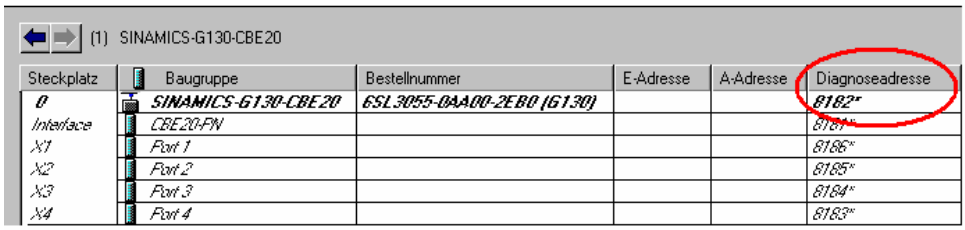
3.5 Using the blocks under PROFINET IO

The blocks described in the previous chapters can also be used with PROFINET IO.

3.5.1 Continuing to use S7 programs for PROFIBUS in PROFINET IO

The points listed below must be observed in order to convert an existing PROFIBUS configuration to PROFINET IO, without having to modify the S7 program.

- Save the existing S7 program; in particular, the DrivDBx with the configuration of the drive slave
- The existing DrivDBx in the PROFIBUS configuration must not be overwritten with a new DB based on the PROFINET configuration generated using the DrivDBx generator. In this case, the old DrivDBx DB would be used unaltered.
- Make a note of the PROFIBUS configuration (diagnostics, IO address and telegram) of the drives
- Delete drive slaves on the PROFIBUS system or remove the DP master system
- Insert the PROFINET IO system
- Insert drives into PROFINET IO system and configure them
- The diagnostics address from the PROFIBUS configuration (also to be found in the DrivDBx for the relevant slave) must be entered in the field marked in red



- The IO address and the telegram selection must be identical to the previous PROFIBUS configuration

3.5.2 Converting to PROFINET IO

The procedure below describes how to configure blocks for use with a PROFINET IO system.

- Create a PROFINET IO configuration
- Save the HW Config
- Use the DrivDBx generator to create the DriveDBx for the PROFINET IO system
- Use the blocks described in the previous sections to program the user program.
- Supply the blocks with data from the DrivDBx DB:

Diagnostics address for the blocks with acyclic communications

Data for input CFG_DATA for FB32 (PCD_RECV)

Data for input CFG_DATA for FB31 (PCD_SEND)

Data for input CFG_DATA for FB32 (PCD_RECV)

Data for input CFG_DATA for FB31 (PCD_SEND)

SLAVE_3.DPADDR	WORD	W#16#0	W#16#3	PROFIBUS-Adresse / Gerätenummer des Slaves
SLAVE_3.DADDR	WORD	W#16#0	W#16#1FED	Diagnoseadresse des Slaves
SLAVE_3.SLOT_1R.LADDR	WORD	W#16#0	W#16#116	I/O base address of slot
SLAVE_3.SLOT_1R.SLOT_ID	BYTE	B#16#0	B#16#3	Slot type ID
SLAVE_3.SLOT_1R.PCD_ADDR	BYTE	B#16#0	B#16#5	Process data address
SLAVE_3.SLOT_1R.LENGTH	BYTE	B#16#0	B#16#9	Slot data length
SLAVE_3.SLOT_1R.CONSIST	BOOL	FALSE	FALSE	Consistency condition
SLAVE_3.SLOT_1S.LADDR	WORD	W#16#0	W#16#116	I/O base address of slot
SLAVE_3.SLOT_1S.SLOT_ID	BYTE	B#16#0	B#16#3	Slot type ID
SLAVE_3.SLOT_1S.PCD_ADDR	BYTE	B#16#0	B#16#5	Process data address
SLAVE_3.SLOT_1S.LENGTH	BYTE	B#16#0	B#16#5	Slot data length
SLAVE_3.SLOT_1S.CONSIST	BOOL	FALSE	FALSE	Consistency condition
SLAVE_3.SLOT_1R_A1.LADDR	WORD	W#16#0	W#16#110	I/O base address of slot
SLAVE_3.SLOT_1R_A1.SLOT_ID	BYTE	B#16#0	B#16#3	Slot type ID
SLAVE_3.SLOT_1R_A1.PCD_ADDR	BYTE	B#16#0	B#16#E	Process data address
SLAVE_3.SLOT_1R_A1.LENGTH	BYTE	B#16#0	B#16#2	Slot data length
SLAVE_3.SLOT_1R_A1.CONSIST	BOOL	FALSE	FALSE	Consistency condition
SLAVE_3.SLOT_1S_A1.LADDR	WORD	W#16#0	W#16#110	I/O base address of slot
SLAVE_3.SLOT_1S_A1.SLOT_ID	BYTE	B#16#0	B#16#3	Slot type ID
SLAVE_3.SLOT_1S_A1.PCD_ADDR	BYTE	B#16#0	B#16#A	Process data address
SLAVE_3.SLOT_1S_A1.LENGTH	BYTE	B#16#0	B#16#2	Slot data length
SLAVE_3.SLOT_1S_A1.CONSIST	BOOL	FALSE	FALSE	Consistency condition

Diagnostics address for the blocks with acyclic communi-cations

- If you want to continue to use an existing PROFIBUS program for PROFINET IO, in which a new DrivDBx has been generated based on the PROFINET IO, you only need to adapt the pointers to the SLOT-UDTs in the manner described
- Load the program to the CPU

4 Technical data and processing times

4.1 Technical block data

Block	Name	Version	Language	Length				Nesting depth	Blocks called
				Local data	MC7 code	Load memory	User memory		
FB31	PCD_SEND	5.3	AWL	24	392	562	428	1	DPWR_DAT
FB32	PCD_RECV	5.3	AWL	34	638	842	674	1	DPRD_DAT, FILL
FB33	PDAT_CY	5.3	SCL	22	2110	2524	2146	1	DPRD_DAT, DPWR_DAT
FB34	PDAT_AC	5.5	SCL	38	2778	3476	2814	1	BLKMOV, FILL, WRREC , RDREC
FB35	DEV_FLT1	5.6	SCL	66	7170	7998	7206	1	WRREC , RDREC
FB36	PDAT_AC2	5.5	SCL	48	2146	2516	2182	1	FILL, TEST_DB, WRREC , RDREC
FB37	DEV_FLT2	5.4	SCL	64	6898	7694	6934	1	WRREC , RDREC
FB38	DEV_FLT3	5.5	SCL	18	1530	1938	1566	1	WRREC , RDREC
FB39	DEV_FLT4	5.5	SCL	20	1774	2212	1810	1	WRREC , RDREC
FB40	PDAT_DL	5.8	SCL	78	5654	6408	5690	1	BLKMOV, FILL, TEST_DB, WRREC , RDREC
FB41	PDAT_UD	5.6	SCL	82	7386	8314	7422	1	BLKMOV, FILL, TEST_DB, WRREC , RDREC
FB42	PDAT_UD2	5.7	SCL	120	9656	10740	9692	1	BLKMOV, FILL, TEST_DB, WRREC , RDREC, TIME_TCK
FB50	SHELL_MMC_ PDAT_DL	5.4	AWL	30	390	790	426	2	READ_DBL, PDAT_DL
FB51	SHELL_MMC_ PDAT_UD	5.4	AWL	30	406	1332	442	2	READ_DBL, PDAT_UD
FB52	SHELL_MMC_ PDAT_UD2	5.4	AWL	30	406	1464	442	2	READ_DBL, PDAT_UD2
FC60	COM_STAT	5.4	SCL	44	484	628	520	1	RDSYSST

4.2 Processing times

The block runtimes or job processing times were measured on a CPU 416-2DP (6ES7 416-2XL01-0AB0) or 315-2DP (6ES7 315-2AF00-0AB0) and a MASTERDRIVES VC with CBP2 (MD).

Block	Name	Function	Drive	Runtime in ms CPU 416-2DP	Runtime in ms CPU 315-2DP
FB31	PCD_SEND	PPO1	MD	0.14	0.44
		PPO2	MD	0.32	0.59
		PPO3	MD	0.14	0.44
		PPO4	MD	0.32	0.59
		PPO5	MD	0.32	0.60
		4 PZD words (consistency: total)	MD	0.33	0.57
		4 PZD words (consistency: word)	MD	0.15	0.50
		8 PZD words (consistency: total)	MD	0.32	0.59
		8 PZD words (consistency: word)	MD	0.16	0.60
		10 PZD words (consistency: total)	MD	0.32	0.60
		10 PZD words (consistency: word)	MD	0.17	0.66
		12 PZD words (consistency: total)	MD	0.33	0.60
		12 PZD words (consistency: word)	MD	0.17	0.71
		14 PZD words (consistency: total)	MD	0.33	0.62
		14 PZD words (consistency: word)	MD	0.18	0.77
		16 PZD words (consistency: total)	MD	0.35	0.63
		16 PZD words (consistency: word)	MD	0.19	0.82

Block	Name	Function	Drive	Runtime in ms CPU 416-2DP	Runtime in ms CPU 315-2DP
FB32	PCD_RECV	PPO1	MD	0.15	0.59
		PPO2	MD	0.32	0.73
		PPO3	MD	0.14	0.60
		PPO4	MD	0.32	0.72
		PPO5	MD	0.33	0.75
		4 PZD words (consistency: total)	MD	0.33	0.72
		4 PZD words (consistency: word)	MD	0.15	0.65
		8 PZD words (consistency: total)	MD	0.32	0.74
		8 PZD words (consistency: word)	MD	0.16	0.75
		10 PZD words (consistency: total)	MD	0.33	0.75
		10 PZD words (consistency: word)	MD	0.17	0.80
		12 PZD words (consistency: total)	MD	0.33	0.76
		12 PZD words (consistency: word)	MD	0.18	0.86
		14 PZD words (consistency: total)	MD	0.34	0.77
		14 PZD words (consistency: word)	MD	0.18	0.91
		16 PZD words (consistency: total)	MD	0.34	0.78
		16 PZD words (consistency: word)	MD	0.19	0.96
FB33	PDAT_CY	Zero job	MD	2.5	7.5
		Job identifier 1-14 (AK 10 not measured)	MD	On average: 28 (22 min.; 29 max.)	On average: 37 (30 min.; 47 max.)
		Average cycle time during job processing	MD	0.58	2.3
FB34	PDAT_AC	Idling	MD	0.01	—
		Zero job	MD	38	47
		Job identifier 1-15	MD	On average: 42 (39 min.; 45 max.)	On average: 44 (40 min.; 47 max.)
		Average cycle time during job processing	MD	0.60	2.4

Block	Name	Function	Drive	Runtime in ms CPU 416-2DP	Runtime in ms CPU 315-2DP
FB35	DEV_FLT1	Idling	MD	0.01	0.01
		Read out fault information	MD	527	600
		Average cycle time during job processing	MD	1.4	2.8
FB36	PDAT_AC2	----			
FB37	DEV_FLT2	----			
FB38	DEV_FLT3	----			
FB39	DEV_FLT4	----			
FB40	PDAT_DL	The runtime of the block depends on – the type and number of jobs – the data kept in the working or load memory – Reaction to REQ_ERR (logging!)			
FB41	PDAT_UD	The runtime of the block depends on – the type and number of jobs – the data kept in the working or load memory – Reaction to REQ_ERR (logging!)			
FB42	PDAT_UD2	The runtime of the block depends on – the type and number of jobs – the data kept in the working or load memory – Reaction to REQ_ERR (logging!)			
FC60	COM_STAT		MD	0.65	0.74

5 Diagnostics

5.1 Drive diagnostics

Block PCD_RECV evaluates the "Group fault" bit in the status word of the drive (bit 3) and signals a drive fault in parameter DEV_FLT (it can do this only if the status word is configured as word 1 in the drive-to-master telegram!). You will then be able to read out the fault memory of the drive to find out more about the cause(s) of the disturbance.

You can read out the complete current fault event (fault number, fault text and possibly fault number) on a MASTERDRIVES, DC-MASTER, MICROMASTER 4xx or SINAMICS device from the fault memory using blocks DEV_FLT1, DEV_FLT2, DEV_FLT3 or DEV_FLT4.

5.2 DP diagnostics

The COM_STAT function can be applied to monitor slaves for station failure or deactivation. If the display COM_FLT or DEACTIV equals TRUE, the receive data must not be accepted.

5.3 S7 system diagnostics

The SIMATIC S7 signals the following slave errors:

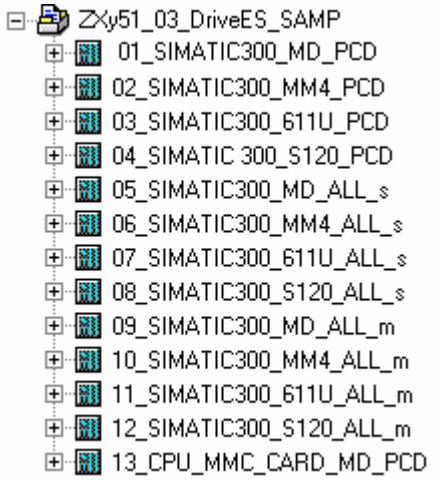
- OB86 (module rack failure) is called when a DP station fails
- OB122 is called in response to an I/O access error.

A selective reaction to the error can thus be programmed in these organization blocks. The corresponding block must always be stored as at least an "empty" block in the PLC. Otherwise the CPU will switch to the STOP state in the event of a fault.

6 Sample programs

After installation of Drive ES SIMATIC-DRVDPS7, the "Examples" directory of the STEP 7 installation contains an example of a project, namely "ZXy51_03_DriveES_SAMP". This project can be changed to 5 SIMATIC languages and contains several SIMATIC 300 stations:

- Each of the stations (..._PCD) contains an executable example of a cyclic process data communication link with a MASTERDRIVES MC Plus, MICROMASTER 420, SINAMICS S120 and a two-axis SIMODRIVE 611U with the PROFIBUS option module. The blocks are called in a multi-instance.
- Each of the stations (..._All_s) contains an executable example of a cyclic and an acyclic communication link with a MASTERDRIVES MC Plus, MICROMASTER 420, SINAMICS S120 and a two-axis SIMODRIVE 611U with the PROFIBUS option module. The blocks are called as a single instance.
- Each of the stations (..._ALL_m) contains an executable example of a cyclic and an acyclic communication link with a MASTERDRIVES MC Plus, MICROMASTER 420, SINAMICS S120 and a two-axis SIMODRIVE 611U with the PROFIBUS option module. The blocks are called in a multi-instance.
- In functional terms, station "13_CPU_MMC_CARD_MD_PCD" contains the same program as station "01_SIMATIC300_MD_PCD". The difference between the two stations is that, in station "13_CPU_MMC_CARD_MD_PCD", an MMC-CPU is used instead of the 315-3DP CPU and the download DB is contained in the loading memory. In the case of this CPU, the FB50 shell block FB50 "SHELL_MMC_PDAT_DL" for calling block FB40 "PDAT_DL" must be used for the download.



Procedure for changing the language

1. Go to the Options menu and select > Manage Multilingual Texts > Change Language.
2. In the language dialog box which appears, select the desired language for the text types.
3. Confirm by clicking "OK".

For more detailed information, please go to the online help system for STEP 7 and look under the index "Manage Multilingual Texts".

6.1 Examples for SIMATIC 300 (...PCD)

These stations contain the HW configuration and the executable STEP 7 program solely for cyclic process data communication between a SIMATIC S7 CPU 315-2DP and a MASTERDRIVES MC Plus with CBP2, MICROMASTER 420, a two-axis SIMODRIVE 611U with the PROFIBUS option module or SINAMICS S120. The blocks are called in a multi-instance.

Each program contains the 'read and write process data' function (PCD_SEND and PCD_RECV). In this program, selected bits of the control and status words are linked to the block envelope. As an alternative, the control word can be specified complete, in which case the inputs with the selected control word bits are inoperative. The main setpoint can also be specified. The complete status word and the main actual value are displayed at the block outputs. Standard telegram 1 (two words per transmit and receive direction) is parameterized for process data communication in the examples.

If the hardware required by the example is already installed, the appropriate program need only be loaded to the CPU and the drive parameterized for data exchange. The easiest method of parameterizing the drive is to start block PDAT_DL or PDAT_UD2 (this interconnects the control and status words and the main setpoint and actual value for PROFIBUS communication).

The stations include a variable table (VAT_...) for each function provided by the examples. The relevant function can be controlled via the table.

VAT_DOWNLOAD_...:	Control download of drive parameters
VAT_PCD_...:	Operator control and process monitoring of process data communication

If the hardware does not meet the example's requirements, it must be adapted.

NOTE:

In the case of SIMODRIVE 611U, a change in the parameters is applied to standard telegram 1 only after a power-on reset.

6.2 Examples for SIMATIC 300 (...All_s)

These stations contain the HW configuration and the executable STEP 7 program for communication between a SIMATIC S7 CPU 315-2DP and a MASTERDRIVES MC Plus with CBP2, MICROMASTER 420, a two-axis SIMODRIVE 611U with the PROFIBUS option module or SINAMICS S120. The blocks are called as a single instance.

Each program contains the functions

- Read and write process data (PCD_SEND and PCD_RECV)
- Process parameters acyclically (PDAT_AC or PDAT_AC2)
- Download drive parameters (PDAT_DL or PDAT_UD2)
- Read out fault buffer (DEV_FLTx)

For the SIMODRIVE 611U, only the Read and Write Process Data functions (PCD_SEND and PCD_RECV) and the Download Drive Parameters function (PDAT_UD2) have been implemented.

Standard telegram 1 (two words per transmit and receive direction) is parameterized for process data communication in the examples.

If the hardware required by the example is already installed, the appropriate program need only be loaded to the CPU and the drive parameterized for data exchange. The easiest method of parameterizing the drive is to start block PDAT_DL or PDAT_UD2 (this interconnects the control and status words and the main setpoint and actual value for PROFIBUS communication).

The stations include a variable table (VAT_...) for each function provided by the examples. The relevant function can be controlled via the table.

VAT_DEV_FLT...:	Read out fault memory of drive
VAT_PCD_DATA...:	Operator control and process monitoring of process data communication
VAT_PDAT_AC...:	Process parameters acyclically
VAT_PDAT_UD...:	Control download of drive parameters

If the hardware does not meet the example's requirements, it must be adapted.

NOTE:

In the case of SIMODRIVE 611U, a change in the parameters is applied to standard telegram 1 only after a power ON reset.

6.3 Examples for SIMATIC 300 (...ALL_m)

These stations contain the HW configuration and the executable STEP 7 program for communication between a SIMATIC S7 CPU 315-2DP and a MASTERDRIVES MC Plus with CBP2, MICROMASTER 420, a two-axis SIMODRIVE 611U with the PROFIBUS option module or SINAMICS S120. The blocks are called in a multi-instance.

Each program contains the functions

- Read and write process data (PCD_SEND and PCD_RECV)
- Upload/download of drive parameters (PDAT_UD or PDAT_UD2)
- Read out fault buffer (DEV_FLTx)

For SIMODRIVE 611U there is no DEV_FLTx block for the function “Read out fault buffer”.

Standard telegram 1 (two words per transmit and receive direction) is parameterized for process data communication in the examples.

If the hardware required by the example is already installed, the appropriate program need only be loaded to the CPU and the drive parameterized for data exchange. The easiest method of parameterizing the drive is to start block PDAT_UD or PDAT_UD2 (this interconnects the control and status words and the main setpoint and actual value for PROFIBUS communication).

The stations include a variable table (VAT_...) for each function provided by the examples. The relevant function can be controlled via the table.

VAT_DEV_FLT...:	Read out fault memory of drive
VAT_PCD_DATA...:	Operator control and process monitoring of process data communication
VAT_PDAT_UD...:	Control download of drive parameters

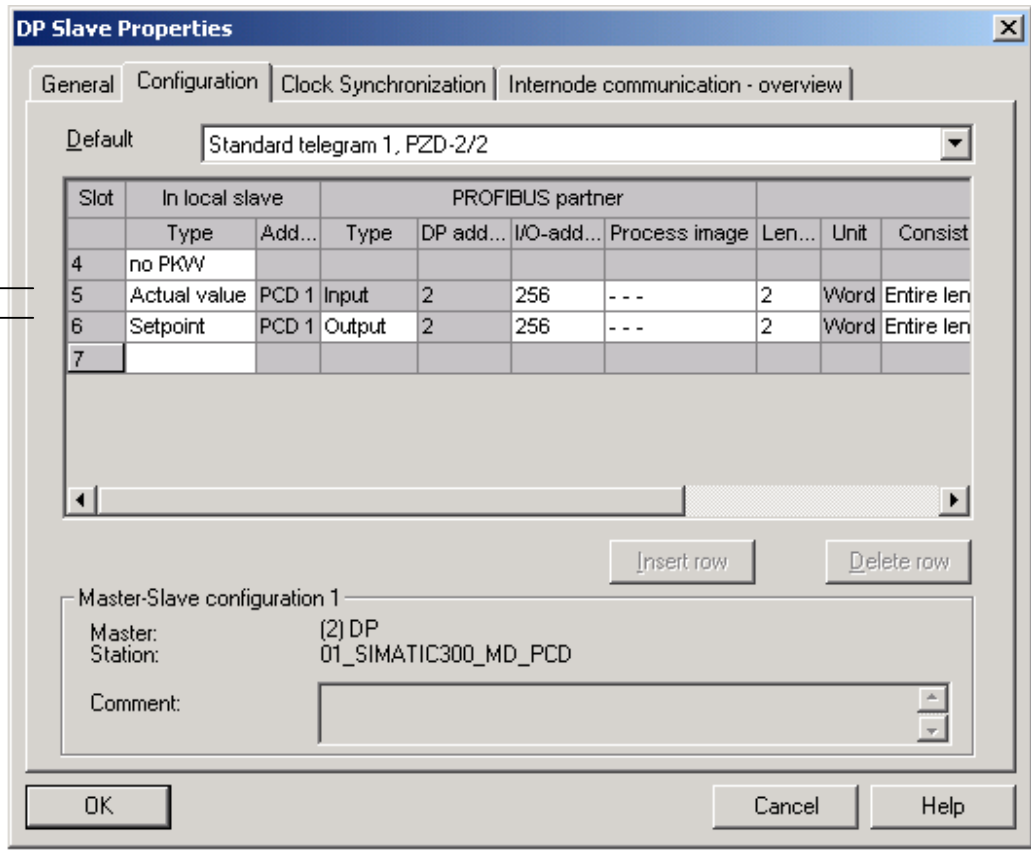
If the hardware does not meet the example's requirements, it must be adapted.

NOTE:

In the case of SIMODRIVE 611U, a change in the parameters is applied to standard telegram 1 only after a power ON reset.

7 Appendix

7.1 Parameter settings for PCD_SEND for several setpoint slots



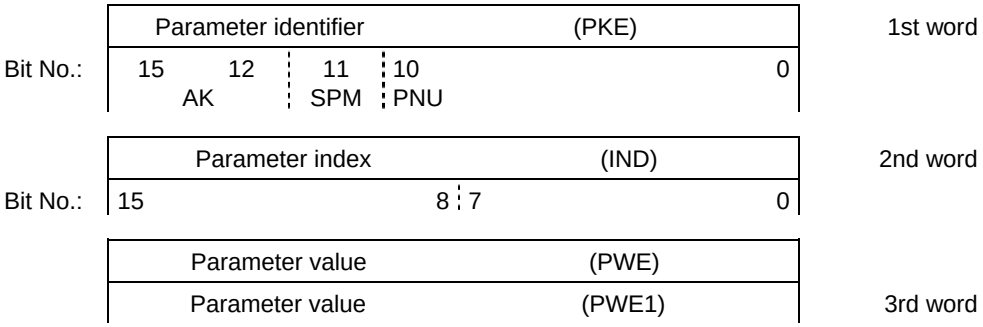

```
CALL  PCD_SEND, DB_PCD_SEND(
      CFG_DATA  := DRIVDB1.SLAVE_1.SLOT_5,
      PCD_1     := MW0, //PZD 1
      PCD_2     := MW2, //PZD 2
      .....
      SFC_ERR   := M30.0,
      CFG_ERR   := M30.1);

CALL  PCD_SEND, DB_PCD_SEND(
      CFG_DATA  := DRIVDB1.SLAVE_1.SLOT_6,
      PCD_1     := MW0, //Data is not transferred to the drive
      PCD_2     := MW2, //Data is not transferred to the drive
      PCD_3     := MW4, //PZD 3
      PCD_4     := MW6, //PZD 4
      PCD_5     := MW8, //PZD 5
      .....
      SFC_ERR   := M30.0,
      CFG_ERR   := M30.1);
```

7.2 Formulating parameter jobs (data record 100)

Parameter jobs processed with the PIV mechanism (DS 100) must be created as follows:

Parameter area (PIV)

The parameter area always includes at least three words.



AK: Job identifier or response identifier

SPM: Toggle bit for spontaneous processing of signals

PNU: Parameter number

A 32-bit parameter value is composed of PWEn (higher-value word) and PWEn+1 (lower-value word).

Parameter identifier (PKE), 1st word

The parameter identifier (PKE) is always a 16-bit value.

Bits 0 to 10 (PNU) contain the number of the desired parameter.

Bit 11 (SPM) is the toggle bit for spontaneous signals.

Bits 12 to 15 (AK) contain the job or response identifier.

The following table shows the meaning of the job identifier. Job identifiers 10 to 15 are specific to MASTERDRIVES. For a download job, only the job identifiers with a grey background are supported.

Job identifier	Meaning	Response identifier Positive	Response identifier Negative
0	No job	0	7 or 8
1	Request for parameter value	1 or 2	↑
2	Change parameter value (word)	1	|
3	Change parameter value (double word)	2	|
4	Request for description element [1]	3	|
5	Change description element (not with CBP)	3	|
6	Request for parameter value (array) [1]	4 or 5	|
7	Change parameter value (array, word) [2]	4	|
8	Change parameter value (array, double word) [2]	5	|
9	Request for number of array elements	6	|
10	Reserved	-	|
11	Change parameter value (array, double word), store in EEPROM [2]	5	|
12	Change parameter value (array, word) and store in EEPROM [2]	4	|
13	Change parameter value (double word) and store in EEPROM	2	|
14	Change parameter value (word) and store in EEPROM	1	↓
15	Read or change text (not with CBP)	15	7 or 8

Job identifiers (Master -> Converter)

[1] The desired element of the parameter description is given in IND (2nd word)

[2] The desired element of the indexed parameter is given in IND (2nd word)

The following table shows the meaning of the response identifier. Response identifiers 11 to 15 are specific to MASTERDRIVES. Depending on the job identifier, only certain response identifiers are possible. For a download job, only the response identifiers on a grey background are relevant.

Response identifier	Meaning
0	No response
1	Transmit parameter value (word)
2	Transmit parameter value (double word)
3	Transmit description element[1]
4	Transmit parameter value (array, word) [2]
5	Transmit parameter value (array, double word) [2]
6	Transmit number of array elements
7	Job cannot be executed (with error number)
8	No parameter change rights for PIV interface
9	Spontaneous signal (word)
10	Spontaneous signal (double word)
11	Spontaneous signal (array, word) [2]
12	Spontaneous signal (array, double word) [2]
13	Reserved
14	Reserved
15	Transmit text (not with CBP)

Response identifiers (Converter ->Master)

[1] The desired element of the parameter description is given in IND (2nd word)

[2] The desired element of the indexed parameter is given in IND (2nd word)

If the response identifier has the value 7 (job cannot be executed), an error number is then stored in parameter value 1 (PWE 1) in accordance with the following table. The error number is shown at the block output, ERR_NO.

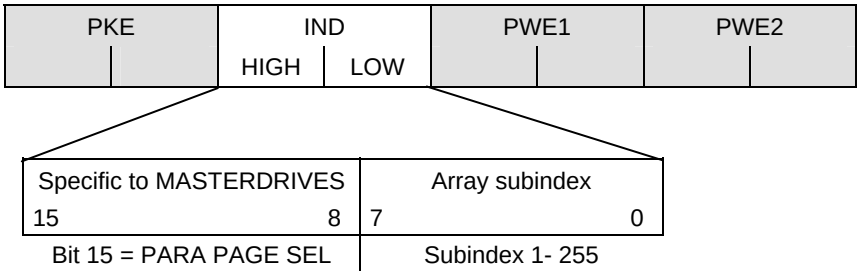
No.	Meaning	
0	Non-permissible parameter number (PNU)	When the PNU does not exist
1	Parameter value cannot be changed	When the parameter is a monitoring parameter
2	Bottom or top value limit exceeded	–
3	Incorrect subindex	–
4	No array	–
5	Incorrect data type	–
6	Setting not allowed (can only be reset)	–
7	Description element cannot be changed	Not possible for MASTERDRIVES
11	No parameter change rights	–
12	Key word missing	Device parameter: ‘Access key’ and/or ‘Par. special access' not set appropriately
15	There is no text array	–
17	Job cannot be executed due to operating state	Converter status does not permit the job at the present time
101	Parameter number deactivated at the present time	Specific to MASTERDRIVES
102	Channel width too small	Specific to MASTERDRIVES: Only for short channels
103	PIV number incorrect	Specific to MASTERDRIVES: Only for G-SST1/2 and SCB interface (USS)
104	Parameter value not permissible	Specific to MASTERDRIVES
105	The parameter has been indexed	e.g. job: ‘change PWE, word' for indexed parameter
106	Job not implemented	

Error numbers in the case of response "Job cannot be executed" (device parameter)

Parameter index (IND) 2nd word

The array subindex is an eight-bit value and is always transmitted in the lower-value byte (bits 0 to 7) of the parameter index (IND) in the case of acyclical data traffic. The task of parameter page selection for additional technology parameters or parameters of the free blocks in the MASTERDRIVES is performed here by the higher-value byte (bits 8 to 15) of the parameter index. This structure corresponds to the statements in the USS specification.

Structure of IND in the case of acyclical communication



Task of the IND

If the subindex in a job is transferred with values between 1 and 254, the desired index of the parameter is transmitted if the parameter is an indexed one. The meaning of the individual indices of a parameter is given in the "Parameter list" in the operating instructions for the converter.

The value 255 for the array subindex has a special significance. If the array index with 255 is transmitted, all the indices of an indexed parameter are transferred at the same time in a data block.

The structure of the transmitted data block is in accordance with the USS specification. The maximum data-block size is 206 bytes.

The bit for parameter page selection has the following effect (example: MASTERDRIVES):

If this bit = 1, the parameter number (PNU) transmitted in the PIV job is given an offset of 2000 in the CBP and then passed on.

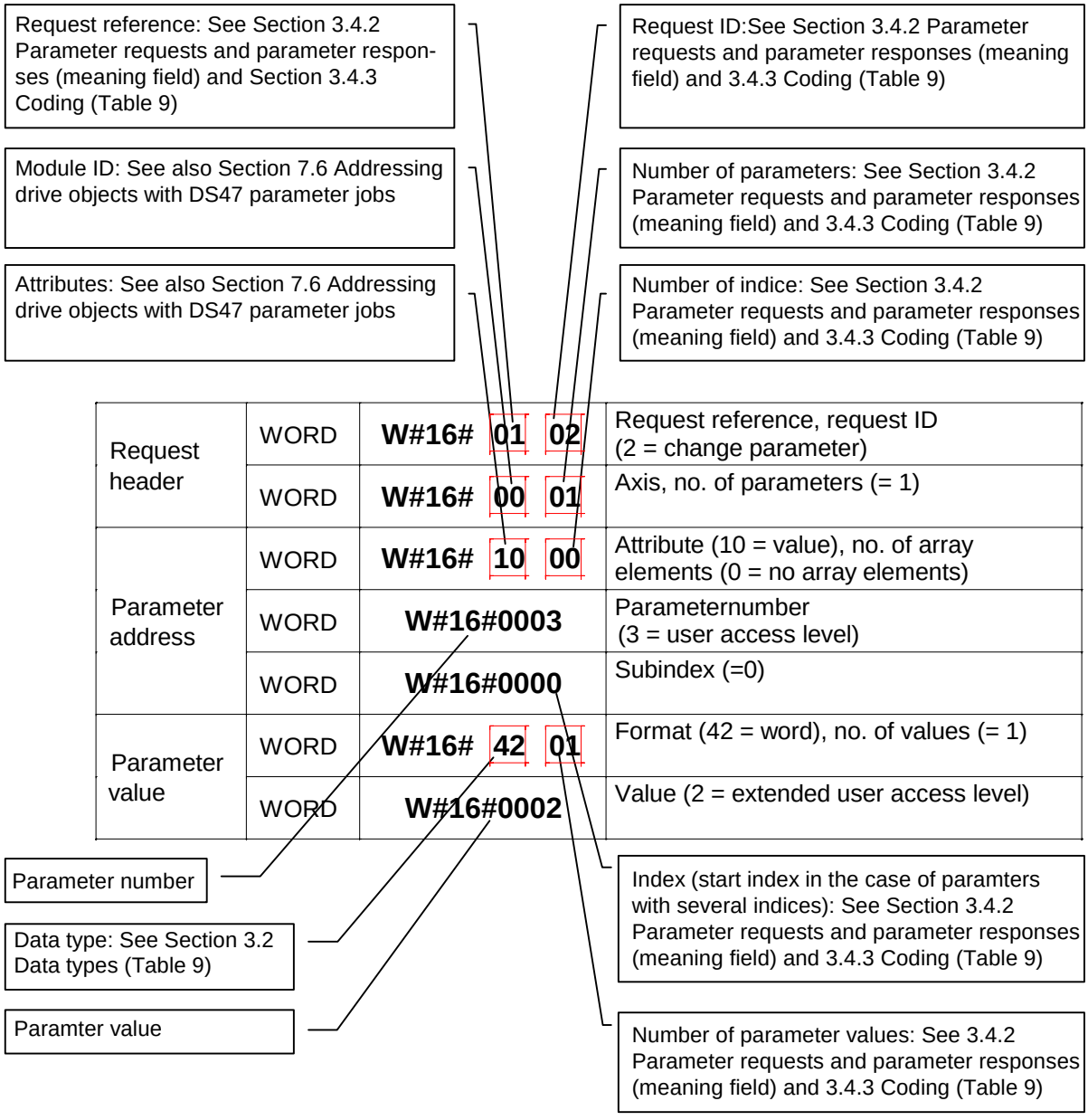
Parameter designation (acc. to parameter list)	Serial parameter number	Necessary addressing of the parameter via PROFIBUS		
		PNU [decimal]	PNU [hex.]	Bit *)
P000 - P999 (r000 - r999)	0 - 999	0 - 999	0 - 3E7	= 0
H000 - H999 (d000 - d999)	1000 - 1999	1000 -1999	3E8 - 7CF	= 0
U000 - U999 (n000 - n999)	2000 - 2999	0 - 999	0 - 3E7	= 1
L000 - L999 (c000 - c999)	3000 - 3999	1000 - 1999	3E8 - 7CF	= 1

***)** Parameter-page selection

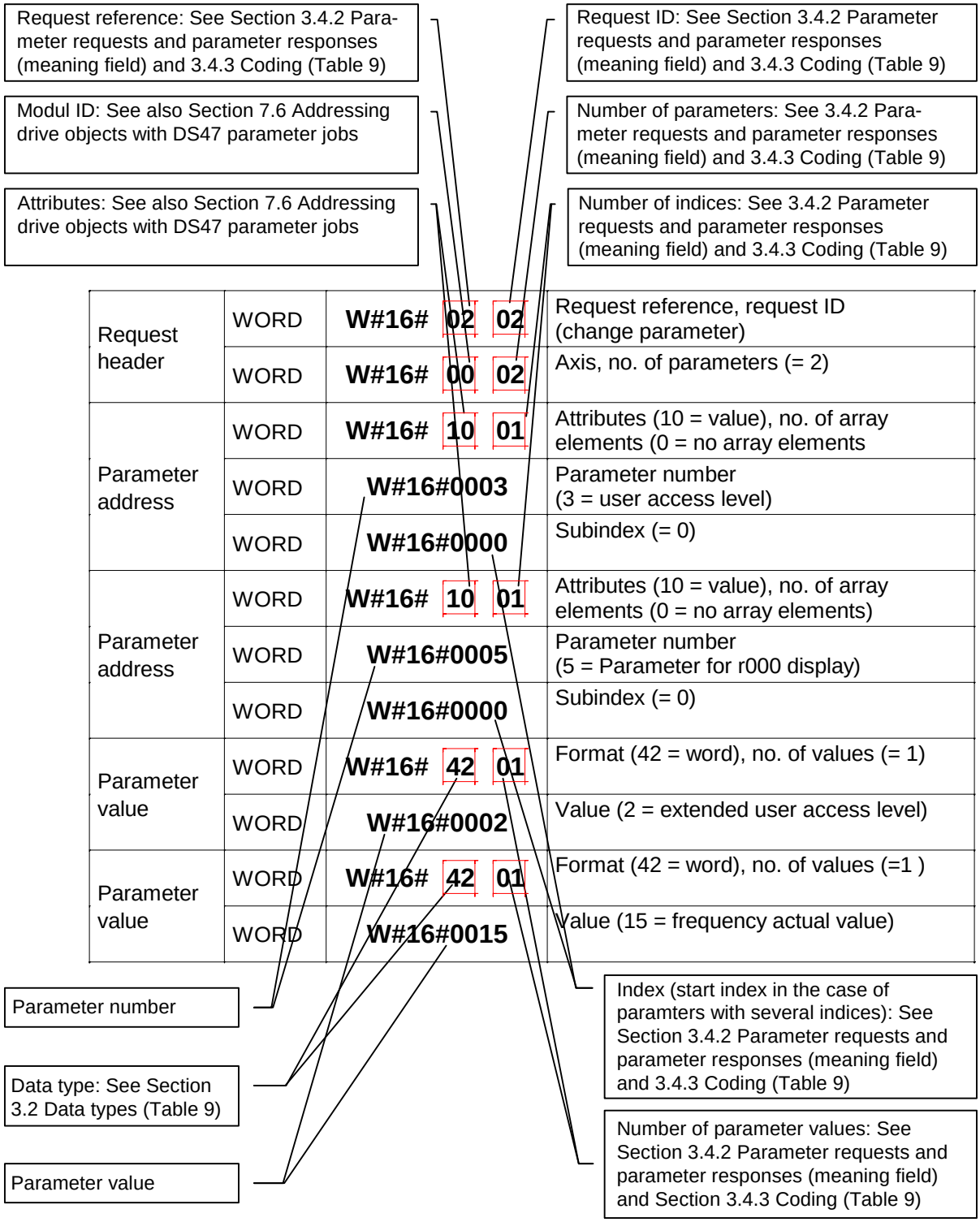
7.3 Formulating parameter jobs (data record 47)

The tables below are intended as a brief explanation of data record 47 jobs and refer to the extract from "PROFIBUS Profile Drive Technology, V3.1, November 2002" which follows this explanation.

Job requesting change of single parameter value:



Job requesting change of several parameter values:



PROFIBUS Profile Drive Technology

The following description is an extract from the "PROFIBUS Profile Drive Technology, V3.1, November 2002", published with kind permission of the PROFIBUS User Organization (Germany).

3. Parameter model

3.1. Parameter definition

A parameter represents an information memory that consists of the following elements:

Table 2 Parameter definition

Parameter value (PWE) (refer to 3.1.1)	Includes the information variable(s)
Parameter description (PBE) (refer to 3.1.2)	Specifies a parameter
Text (refer to 3.1.3)	Is used to support visualization and contains a general description on the parameter function or on the parameter value

The total of all parameters of a drive uniquely describes its behavior or characteristics.

A parameter number is assigned to each parameter. The number range of the parameters is specified for decimal 1-65535. The parameter 0 is not permitted. Profile-specific parameters are specified or reserved for the ranges decimal 900-999 and decimal 60000 - 65535(refer to Appendix **A.1**).

Access to the parameters (parameter value, parameter description or text) is explained in Chapter 3.4 and Appendix A.10.

3.1.1. Parameter value

The parameter value contains a single (simple variable) or several similar (array) information variables.
An array consists of n elements of the same data type which can be individually addressed with sub-indices from 0 to n-1.

3.1.2. Parameter description

The parameter description contains relevant information about the respective parameter. The table below shows the structure of the parameter description which will be discussed in the course of this chapter.

Table 3 Parameter description elements

Subindex	Meaning	Data type (refer to 3.2)
1	Identifier (ID)	V2
2	Number of array elements or length of string	Unsigned 16
3	Standardization factor	Floating Point
4	Variable attribute	OctetString 2
5	Reserved	OctetString 4
6	Name	VisibleString 16
7	Low limit	OctetString 4
8	High limit	OctetString 4
9	Reserved	OctetString 2
10	ID extension	V2
11	PZD reference parameter	Unsigned 16
12	PZD normalization	V2
0	Complete description	OctetString 46

Identifier (ID) (subindex 1)

Additional characteristics of the parameter are stored in the ID. Bit value = “0” means: “parameter does not possess this attribute”. Bit value = “1” means: “parameter possesses this attribute ".

Table 4 Parameter description element "Identifier (ID)"

Bit	Meaning	Explanation
15	Reserved	
14	Array	
13	Parameter value can be reset only	If this bit is set, the associated parameter value is increased exclusively by internal processing while externally it can only be set to “0” (for example, “time differences “).
12	Parameter was changed with respect to the factory setting	This bit is set, if the parameter value is unequal to the factory setting. It is reset, if the parameter value is equal to the factory setting.
11	Reserved	
10	Additional text array available	
9	Parameter not writeable	
8	Standardization factor and variable attribute not relevant	This bit is set, if parameters have data types to which physical values cannot be calculated, for example the data type string.
0 - 7	Data type of the parameter value (index from Table 9)	

Number of array elements res. string length (subindex 2)

In the case of array parameters, the number of elements is entered here.
In the case of parameters of the data type “String”, the length of the string is entered here. The data types OctetString or VisibleString correspond to an array of bytes. It is not possible to form an array of the data type “String”.

Standardization factor (subindex 3)

Factor that converts the (internal) value into an (external) standardized variable which, together with the unit, corresponds to the physical representation of the parameter. The standardization factor is of the data type Floating Point.

Variable attribute (subindex 4)

A variable index and a conversion index is stored in the variable attribute (Table 44 and Table 45, refer to Appendix A.5):

Table 5 Parameter description element "variable attribute“

Octet 1	**Octet 2**
variable index	Conversion index (Factor A, Offset B)

The variable index represents the fixed coding of the physical variable (and therefore the base unit) of the parameter value. The variable index is of the data type Unsigned 8.

The conversion index represents the fixed coding of the conversion factor (A) and the offset (B) for a parameter value. With the conversion index, the unit can be converted into the base unit. The conversion index is of the data type Integer 8.

Example: variable attribute = 13 / -3 (0x0DFD)
- Variable index = 13 -> physical variable “speed”, basic unit “meter/second”
- Conversion Index = -3 -> unit “millimeter/second” (factor A=0.001, offset B=0)

Examples: External representation (by means of standardization factor, Variable attribute)

The following applies:

Physical value (in the unit)	= transmitted value * standardization factor * unit
or	
Physical value (in the base unit)	= (transmitted value * standardization factor * A + B) * base unit

Coding for the variable index and the conversion index (Factor A, Offset B): refer to Appendix **A.5**.

Example 1:
- Data Type Integer16
- Transmission Value: 500
- Standardization Factor: 1.0
- Variable Index: 21 -> physical variable "Electrical Voltage", base unit “Volt“
- Conversion Index: -3 -> unit "Millivolt" (Factor A=0.001, Offset B=0)

-> Physical Value (in the unit) = 500 * 1.0 mV = 500 mV
-> Physical Value (in the base unit) = (500 * 1.0 * 0.001 + 0) V = 0.5 V

Example 2:
- Data Type Unsigned16
- Transmission Value: 1234
- Standardization Factor: 0.01
- Variable Index: 13 -> physical variable "Speed“, base unit “Meter/Second“
- Conversion Index: 73 -> unit "Kilometer/Hour" (Factor A=1000/3600, Offset B=0)

-> Physical Value (in the unit) = 1234 * 0.01 km/h = 12.34 km/h
-> Physical Value (in the base unit) = (1234 * 0.01 * 1000/3600 + 0) m/s = 3.428 m/s

Example 3:
- Data Type N2 -> 100 % corresponds to 2^{14} (refer to Appendix **A.4.2**)
- Transmission Value: 8043
- Standardization Factor: 0.0061 (100 / 2^{14})
- Variable Index: 24 -> physical variable "Ratio", base unit “Percent“
- Conversion Index: 0 -> unit = base unit "Percent" (Factor A=1, Offset B=0)

-> Physical Value = 8043 * 0.0061 % = 49.1 %

Note:
In the case of the data types N2, N4 / X2, X4 / optional Integer16, Integer32, Floating Point (normalized variables), the unit % can be converted to another physical unit by assigning a physical reference parameter (see below, Description Element PZD Reference Parameter).

Name (subindex 6)

Symbolic name of the parameter. The name is of the data type VisibleString with a length of 16.

Low/high limit (subindices 7 and 8)

Defines the valid value range of the parameter value. The drive rejects an attempt to assign a value outside of the parameter’s value range.
The low and the high limit are of the same data type as the parameter value, but the length of the description elements is always 4 bytes (file format: right justified, big endian).

For parameters whose data types permit no value range (for example, VisibleString), the contents of these description elements are of no importance.

ID extension (subindex 10)
The ID extension is reserved.

PZD reference parameter / PZD normalization (subindices 11 and 12)
Parameter values can also be transmitted as process data (refer to Chapter 4.4.3). If normalized variables (data types N2, N4 / X2, X4 / optional Integer16, Integer32, Floating Point) are transmitted, the following is necessary for calculating the physical value: the physical reference value (process data reference value), and the bit (refer to Process Data Normalization) to which the physical reference value refers. For a calculation example, refer to Chapter 4.4.5.

For parameters of the data type X2, X4, the description elements “PZD reference parameter” and “PZD normalization” must be available.
For parameters of the data type N2, N4, the description elements “PZD reference parameter” must be available, and the description element “PZD normalization” is optional as it is defined by the data type.
If parameters of the data type Integer16, Integer32, Floating Point, are transmitted as normalized process data (with unit "%"), the description elements “PZD reference parameter” and “PZD normalization” must be available. If they are transmitted as non-normalized, these description elements must not be available.
In the case of all other data types these description elements are not relevant.

Table 6 Parameter Description Elements “Process Data Reference Value/Process Data Normalization “

Description Element	Content	
PZD reference parameter	0 1-65535	No reference value available Parameter number of the reference value
PZD normalization	Bit 0-5 Bit 6-14 Bit 15	Normalization bit 0-31 (32-63 is reserved) reserved Normalization valid

Notes:
- For the parameters with data type N2, N4, the coding of the normalization bit is fixed (14 and 30)
- For the normalized parameters with data type Floating Point, the coding of the normalization bit is not relevant (= 0).
- The combination “no reference value exists”/ “ normalization valid” is permissible.
- Parameters that are used for reference values are not to be normalized.
- If the complete parameter description is read out with one access, the description elements must be included (see below).

Complete description (subindex 0)
The “complete description” includes a total field of 46 bytes (corresponding to the complete parameter description structure). This length is the constant for each parameter (regardless of the data type, etc.).

3.1.3. Text
Text from a text array may be assigned to a parameter as an additional explanation or description. An indexed text line has a length of 16 bytes.

Subindex text array	Text
0	Text 0 (16 bytes)
1	Text 1 (16 bytes)
2...n	Text 2...n (16 bytes each)

The existence of a text array is marked within the parameter description (ID: additional text array available). The text is stored in the object type “array” of the data type “VisibleString 16” assigned to the parameter.

Text arrays may be assigned to parameters of the object type “array” (with any data type), or to parameters of the object type “simple variable” (with data type “Unsigned8/16/32”, “Boolean”, or “V2”). The individual texts of a text array are assigned to the array elements for parameters of the type “array”, and assigned to the values for parameters of the type “simple variable”.

Array parameter - text array

Subindex text array == Subindex array parameter

Unsigned8/16/32 – text array

Subindex text array == parameter value

$0 \leq$ Parameter value ≤ 65535

Boolean - text array
Number of texts = 2

Table 7 Text array for the data type Boolean

Subindex text array	Parameter value
0	"false"
1	"true"

V2 - text array

Number of texts = 32
To each bit of the bit sequence two texts are assigned, one each to the bit value “0” and “1”.

Subindex Text Array == Bit Position * 2 + Bit Value
0 (LSB) $\leq$ Bit Position ≤ 15 (MSB), $0 \leq$ Bit Value ≤ 1;

Table 8 Text array for data type V2 (bit sequence)

Subindex text array	Parameter value
0	- - - - - - - - - - - - - - - 0
1	- - - - - - - - - - - - - - - 1
2	- - - - - - - - - - - - - - 0 -
3	- - - - - - - - - - - - - - 1 -
4	- - - - - - - - - - - - - 0 - -
:	:
30	0 - - - - - - - - - - - - - - -
31	1 - - - - - - - - - - - - - - -

3.2. Data types

Profile-specific data types are defined corresponding to the particular drive requirements. The profile-specific data types are individually defined in Appendix A.4.2. The standard types are defined in Appendix **A.4.1**. An overview of all of the permissible data types (standard data types and profile-specific data types) and their coding is given in Table 9.

Table 9 Data types

Coding (decimal)	**Data type**	**Comment**
1	Boolean	Standard data type
2	Integer8	Standard data type
3	Integer16	Standard data type
4	Integer32	Standard data type
5	Unsigned8	Standard data type
6	Unsigned16	Standard data type
7	Unsigned32	Standard data type
8	FloatingPoint	Standard data type
9	VisibleString	Standard data type
10	OctetString	Standard data type
12	TimeOfDay (with date indication)	Standard data type
13	TimeDifference	Standard data type
...		
33	N2 Normalized value (16 bit)	Profile-specific data type
34	N4 Normalized value (32 bit)	Profile-specific data type
35	V2 Bit sequence	Profile-specific data type
36	L2 Nibble	Profile-specific data type
37	R2 Reciprocal time constant	Profile-specific data type
38	T2 Time constant (16 bit)	Profile-specific data type
39	T4 Time constant (32 bit)	Profile-specific data type
40	D2 Time constant	Profile-specific data type
41	E2 Fixed point value (16 bit)	Profile-specific data type
42	C4 Fixed point value (32 bit)	Profile-specific data type
43	X2 Normalized value, variable (16 bit)	Profile-specific data type
44	X4 Normalized value, variable (32 bit)	Profile-specific data type
...		
50	Date	Standard data type
52	TimeOfDay without date indication	Standard data type
53	TimeDifference with date indication	Standard data type
54	TimeDifference without date indication	Standard data type
...		

Note: The standard data types specified in Appendix **A.4.1** and their coding are up-to-date according to this profile edition. However, they will still be revised. The latest version of [4] should be consulted to get the most up-to-date status.

3.3. Multi-axis Drive Units

As defined in Chapter 4.4.3, a drive unit is a modular device and consists of the drive device itself and one or several drive axes.
For multi-axis drives, each drive axis has a dedicated parameter number space.
Two types of parameters with different ranges of values are defined in the profile:

- global parameter:
  Global parameters are valid for the complete device independent of the axis (for example parameter 918 "Node address"). If addressing different axes of a drive unit, a global parameter will always supply the same value.
- axis-specific parameters:

These parameters exist in the drive axes, but not on the drive device. The axis-specific parameters can have different values in every drive axis (for example parameter 967 "control word1").

The subdivision of the parameters into global and axis-specific parameters is in Appendix A.1.

The following figure shows an example with global parameter 918 "node address" and the axis-specific parameter 944 "fault message counter".

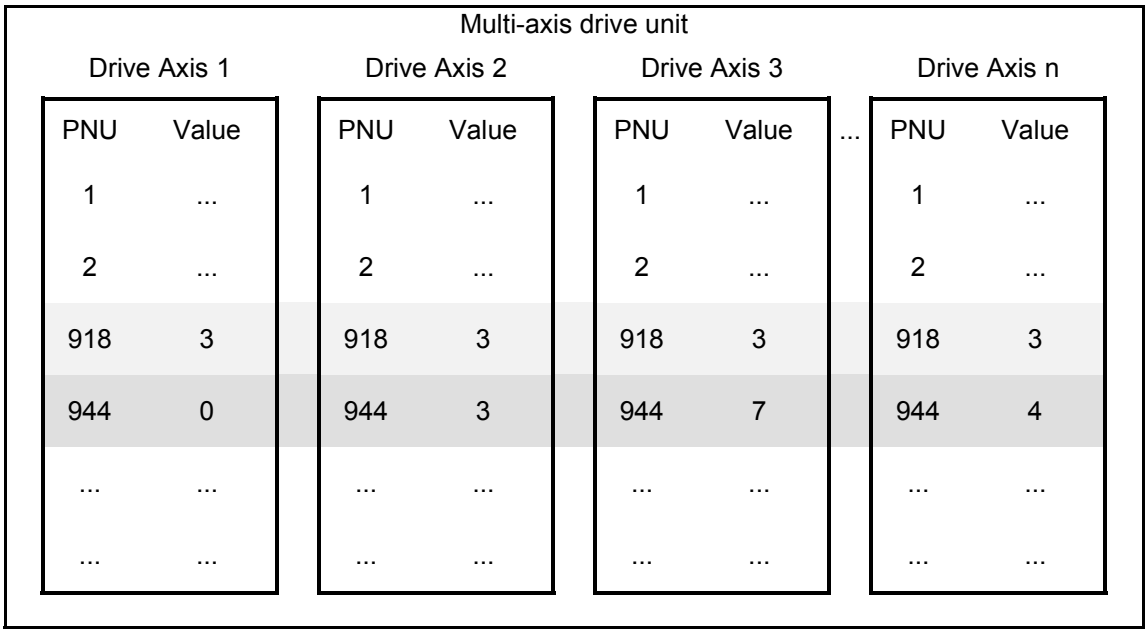


Fig.13 Example overview of global and local parameters of a multi-axis drive unit

The value range of the drive axis numbers range from 1-255. With the address 0 the drive device itself can be addressed and the global parameters can be read. The assignment of the drive axis numbers to the axis is device-specific and can be read out of parameter P978" list of module IDs" (refer to Chapter 4.4.3).
The addressing is described in Chapter 3.4.

3.4. Parameter access with DPV1

In this chapter the access to drive parameters via DPV1 is defined. A request language will be defined for the access. The requests and the replies are transmitted acyclically with DPV1 data blocks.

In the Initiate telegram of a Master Class 2 connection the attribute “Profile Ident Number” shall have the value 0x3A00 for the PROFIdrive Profile.

3.4.1. General characteristics

- Compatibility with PKW (parameter ID/parameter value) requests according to Profile Version 2 (refer to Appendix A.10.2)
- 16-bit wide address each for parameter number and subindex.
- Transmission of complete arrays or parts of them, or the entire parameter description.
- Transmission of different parameters in one access (multi-parameter requests).
- Always just one parameter request is being processed at a time (no pipelining).
- A parameter request/parameter response has to fit in a data block (240 bytes max.) The requests/replies are not split-up over several data blocks. The maximum length of the data blocks may be less than 240 bytes depending on slave characteristics or bus configuration.
- No spontaneous messages will be transmitted.

- For optimized simultaneous access to different parameters (for example, operator interface screen contents), "multi-parameter" requests will be defined.
- There are no cyclic parameter requests.
- After run-up, the profile-specific parameters must be at least readable in every state.

3.4.2. Parameter requests and parameter responses

A parameter request consists of three segments:

- Request header: ID for the request and number of parameters which are accessed. Multi-axis drives, Addressing of one axis.
- Parameter address: Addressing of a parameter. If several parameters are accessed, there are correspondingly many parameter addresses. The parameter address appears only in the request, not in the response.
- Parameter value: Per addressed parameter, there is a segment for the parameter values. Depending on the request ID, parameter values appear only either in the request or in the reply.

Words and double words:
The following telegram contents are displayed in words (a word or 2 bytes per line). Words or double words will have the most significant byte being transmitted first (big endian).

Word:	Byte 1	Byte 2

Double word:	Byte 1	Byte 2
	Byte 3	Byte 4

DPV1 parameter request:

Request Header	Request Reference	Request ID	0
	Axis	No. of Parameters = n	2
1st Parameter Address	Attribute	No. of Elements	4
	Parameter Number		
	Subindex		
nth Parameter Address	...		4 + 6 * (n-1)
1st Parameter Value(s) (only for request "Modify")	Format	No. of Values	4 + 6 * n
	Values		
	...		
nth Parameter Values	...		
			4 + 6 * n +...+ (Format_n * Qty_n)

DPV1 parameter response:

Response Header	Request Ref. mirrored	Response ID	0
	Axis mirrored	No. of Parameters = n	2
1st Parameter Value(s) (only after request "Request")	Format	No. of Values	4
	Values or Error Values		
	...		
nth Parameter Values	...		
			4 + ... + (Format_n * Qty_n)

Meaning of the fields:

Request Header:

- Request Reference:
  Unique identification of the request/response pair for the master. The master changes the request reference with each new request (for example, modulo 255). The slave mirrors the request reference in the response.
- Request ID:
  Two IDs are defined:
  - Request parameter
  - Change parameter

  A parameter change can be stored either in volatile or non-volatile RAM according to the device. A changed parameter that is stored in volatile RAM first can be stored in ROM with parameter P971.
  The differentiation Value/Description/Text known from PKW requests is added to the address as attribute. The differentiation Word/Double Word is added to the parameter values as format. For the differentiation Single/ Array Parameter, refer to “No. Of Elements” in the parameter address.
- Response ID:
  Mirroring of the request ID with supplement information whether the request was executed positively or negatively.
  - Request parameter positive
  - Request parameter negative (it was not possible to execute the request, entirely or partially)
  - Change parameter positive
  - Change parameter negative (it was not possible to execute the request, entirely or partially)

  If the response is negative, error numbers are entered per partial response instead of values.
- Axis:
  Axis addressing for multi-axis drives. This enables various axes to be able to be accessed each with a dedicated parameter number space in the drive via the same DPV1 connection.
- No. of Parameters:
  In the case of multi-parameter requests, specifying the number of the following Parameter Address and/or Parameter Value areas. For single requests the No. of parameters = 1.
  Value range 1 .. 39 (limitation because of DPV1 telegram length)

Parameter Address:

- Attribute:
  Type of object which is being accessed. Value range:
  - Value
  - Description
  - Text
- Number of Elements:
  Number of array elements that are accessed or length of string which is accessed.
  Value range: 0, 1..234
  Limitation because of DPV1 telegram length.
  Special Case Number of Elements = 0:
  If values are accessed: recommended for non-indexed parameters in achieving a compatible conversion of the parameter request into a PKW request according to the PROFIdrive Profile, Version 2 (differentiation “request/change parameter value” and “request/change parameter value (array)”).
- Parameter Number:
  Addresses the parameter that is being accessed. Value range: 1..65535.
- Subindex:
  Addresses the first array element of the parameter or the beginning of an string access or the text array, or the description element that is being accessed. Value range: 0.. 65535.

Parameter Value:

- Format:
  Format and number specify the location in the telegram to which subsequent values are assigned.
  Value range:

- Zero (without values as positive partial response to a change request)
- Data type (refer to Appendix A.4)
- Error (as negative partial response)
  Instead of a data type, the following are possible:
- Byte (for description and texts)
- Word
- Double word

The distinction Word/Double Word is a precondition for a compatible conversion of the parameter request into a PKW (Parameter ID/ Parameter Value) request according to Profile Version 2.

- Number of Values:
  Number of the following values
- Values:
  The values of the parameter
  If the values consist of an odd number of bytes, a zero byte is appended in order to secure the word structure of the telegrams.

In the case of a positive partial response, the parameter value contains the following:
- Format = (Data Type or Byte, Word, Double Word)
- Number of values
- the values

In the case of a negative partial response, the parameter value contains the following:
- Format = error
- No. of values = 1
- Value = error value = error number

In the case of a negative response, the parameter value can contain the following:
- Format = error
- No. of values = 2
- Value 1 = Error Value 1: error number
- Value 2 = Error Value 2: subindex of the first array element where the error occurs
  (Purpose: after a faulty write access to an array, not all values must be repeated).

In the case of a positive partial response without values, the parameter value contains the following:
- Format = zero
- Number of values = 0
- (no values)

Not all combinations consisting of attribute, number of elements, and subindex are permitted (refer to Table 10).

A parameter which is not indexed in the profile, can be realized with indices in the drive unit, if the response to a parameter access is profile specific.

Table 10 Permissible combinations consisting of attribute, number of elements and subindex (see also A.10.2)

Attribute	No. of Elements	Subindex	=> Data	Comment
Value (single parameter)	0 *)	0	The value	Compatible with PKW => no array *)
	1	0	The value	Caution: Improved harmonization to index parameters, but PKW incompatible
(Indexed parameter)	1 *)	0...n	One value, under subindex	Compatible with PKW => Array *)
	2...n **)	0...n	Several values, starting with subindex	
Description	0 (irrelevant)	0	The entire description	
	1	1...n	One description element	Compatible with PKW
Text (from text array)	1	0...n	One text (16bytes), under subindex	
	2...n	0...n	Several texts, starting with subindex	

*) Regarding compatibility to PKW: Differentiation between PKW task and single parameter/array parameter: Single parameter: No. of elements = 0 (Subindex irrelevant, preferred 0); Array parameters: No. of elements > 0

**) If the number of elements available in the device does not match with the number of elements which are requested or shall be changed, an error should be output.

3.4.3. Coding

Table 11 Coding of the fields in parameter request/parameter response

Field	Data Type	Values		Comment
Request Reference	Unsigned8	0x00 0x01..0xFF	reserved	
Request ID	Unsigned8	0x00 **0x01** **0x02** 0x03..0x3F 0x40..0x7F 0x80..0xFF	reserved **Request parameter** **Change parameter** reserved manufacturer-specific reserved	
Response ID	Unsigned8	0x00 **0x01** **0x02** 0x03..0x3F 0x40..0x7F 0x80 **0x81** **0x82** 0x83..0xBF 0xC0..0xFF	reserved **Request parameter(+)** **Change parameter(+)** reserved manufacturer-specific reserved **Request parameter(-)** **Change parameter(-)** reserved manufacturer-specific	
Axis	Unsigned8	0x00..0xFF	Number 0..255	
No. of Parameters	Unsigned8	0x00 **0x01..0x27** 0x26..0xFF	reserved **Quantity 1..39** reserved	Limitation through DPV1 telegram length
Attribute	Unsigned8	0x00 **0x10** **0x20** **0x30** 0x40..0x70 0x80..0xF0	reserved **Value** **Description** **Text** reserved manufacturer-specific	The four less significant bits are reserved for (future) expansion of “No. of elements” to 12bits
No. of Elements	Unsigned8	**0x00** **0x01..0xEA** 0xEB..0xFF	**Special Function** **Quantity 1..234** reserved	Limitation through DPV1 telegram length
Parameter Number	Unsigned16	0x0000 **0x0001...** **0xFFFF**	reserved **Number 1.. 65535**	
Subindex	Unsigned16	**0x0000...** **0xFFFF**	**Number 0..65535**	
Format	Unsigned8	0x00 **0x01..0x36** 0x37..0x3F **0x40** **0x41** **0x42** **0x43** **0x44** 0x45.. 0xFF	Reserved **Data types** Reserved **Zero** **Byte** **Word** **Double word** **Error** Reserved	The one which is writing preferably enters “correct” data types (refer to Chapter 3.2).As substitute, Byte, Word, or Double Word are also possible. The one who is reading has to be able to interpret all values.
No. of Values	Unsigned8	**0x00..0xEA** 0xEB..0xFF	**Quantity 0..234** reserved	Limitation through DPV1 telegram length
Error Number	Unsigned16	0x0000... 0x00FF	Error Numbers (see table below)	The more significant byte is reserved

The device shall output an error, if reserved values are accessed.

Table 12Error numbers in DPV1 parameter responses

Error No.	Meaning	Used at	Supplem. Info
0x00	Impermissible parameter number	Access to unavailable parameter	0
0x01	Parameter value cannot be changed	Change access to a parameter value that cannot be changed	Subindex
0x02	Low or high limit exceeded	Change access with value outside the value limits	Subindex
0x03	Faulty subindex	Access to unavailable subindex	Subindex
0x04	No array	Access with subindex to non-indexed parameter	0
0x05	Incorrect data type	Change access with value that does not match the data type of the parameter	0
0x06	Setting not permitted (can only be reset)	Change access with value unequal to 0 where this is not permitted	Subindex
0x07	Description element cannot be changed	Change access to a description element that cannot be changed	Subindex
0x08	reserved	*(PROFIdrive Profile V2: PPO-Write requested in IR not available)*	-
0x09	No description data available	Access to unavailable description (parameter value is available)	0
0x0A	reserved	*(PROFIdrive Profile V2: Access group wrong)*	-
0x0B	No operation priority	Change access without rights to change parameters	0
0x0C	reserved	*(PROFIdrive Profile V2: wrong password)*	-
0x0D	reserved	*(PROFIdrive Profile V2: Text cannot be read in cyclic data transfer)*	-
0x0E	reserved	*(PROFIdrive Profile V2: Name cannot be read in cyclic data transfer)*	-
0x0F	No text array available	Access to text array that is not available (parameter value is available)	0
0x10	reserved	*(PROFIdrive Profile V2: No PPO-Write)*	-
0x11	Request cannot be executed because of operating state	Access is temporarily not possible for reasons that are not specified in detail	0
0x12	reserved	*(PROFIdrive Profile V2: other error)*	-
0x13	reserved	*(PROFIdrive Profile V2: Data cannot be read in cyclic interchange)*	-
0x14	Value impermissible	Change access with a value that is within the value limits but is not permissible for other long-term reasons (parameter with defined single values)	Subindex
0x15	Response too long	The length of the current response exceeds the maximum transmittable length	0
0x16	Parameter address impermissible	Illegal value or value which is not supported for the attribute, number of elements, parameter number or subindex or a combination	0
0x17	Illegal format	Write request: Illegal format or format of the parameter data which is not supported	0
0x18	Number of values are not consistent	Write request: Number of the values of the parameter data do not match the number of elements in the parameter address	0
0x19	axis nonexistent	Access to an axis which does not exist	0
...			
up to 0x64	reserved	-	-
0x65..0xFF	Manufacturer-specific	-	-

The error numbers 0x00 .. 0x13 are taken from PROFIdrive Profile, Version 2. Values that cannot be assigned are reserved for future use.

3.4.4. DPV1 telegram sequences

The data in the write request corresponds to the parameter request. The data in the read response corresponds to the parameter response.
A write request is first sent with the parameter request. The master has to send a read request so that the slave answers with a read response containing the parameter response.

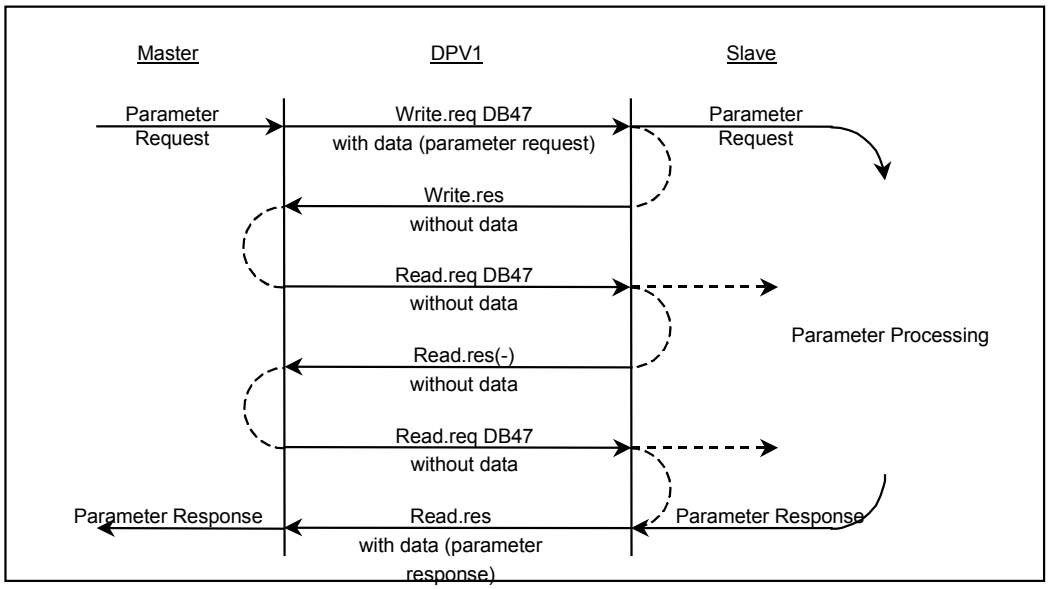


Fig.14 Telegram sequence via DPV1

Table 13 State machine for slave processing

Significance: The columns specify the state.
The rows explain an event. Every row is subdivided into two fields. One describes the action and the other the subsequent state.

State → Event ↓		Connection disconnected	Idle	Request being processed	Response available
Connection being established	Action		Reset processing		
	Subsequent state	Idle	Idle		
Connection being disconnected	Action	(ignore)	Reset processing		
	Subsequent state	-	Connection disconnected		
Write.req	Action	(protocol error) Write.res(-)	Start processing Write.res(+)	Write.res(-) "state conflict"	Reject response Start processing Write.res(+)
	Subsequent state	-	Request being processed	-	Request being processed
Read.req	Action	(protocol error) Read.res(-)	Read.res(-) "state conflict"		Read.res(+)
	Subsequent state	-	-		Idle
Processing completed	Action	Reject	Reject		(internal error) reject
	Subsequent state	-	-	Response available	-

Note:
This state machine is valid for a DPV1 connection. If several connections have been set-up, then the appropriate number of state machines must be available.
The order of a DPV1 telegram sequence is always: first write request, then read request.

3.4.5. DPV1 telegram frame

Standard case:
The following four DPV1 telegrams are used for transmitting a parameter request/parameter response pair:

1) Transmission of the parameter request in a DPV1 Write request:

DPV1 Write Header	Function_Num = 0x5F (Write)	Slot_Number = ...
	Index = 47	Length = (Data)
DPV1 Data (Length)	Parameter request ...	
	...	

2) Short acknowledgement of the parameter request with DPV1 Write response (without data):

DPV1 Write Header	Function_Num = 0x5F (Write)	Slot_Number = (mirrored)
	Index = (mirrored)	Length = (mirrored)

3) Request of the parameter response in a DPV1 Read request (without data):

DPV1 Read Header	Function_Num = 0x5E (Read)	Slot_Number = ...
	Index = 47	Length = MAX

4) Transmission of the parameter response in the DPV1 Read Response:

DPV1 Read Header	Function_Num = 0x5E (Read)	Slot_Number = (mirrored)
	Index = (mirrored)	Length = (Data)
DPV1 Data (Length)	Parameter response ...	
	...	

Meaning and utilization of the elements in the DPV1 Header:

- Function_Num:
  ID for DPV1 service (Read, Write, Error)
- Slot_Number:
  DPV1: In the request: addressing of a real or virtual module of the slave, in the response: mirrored.
  PROFIdrive: no evaluation.
- Index (Data Block):
  DPV1: In the request: addressing of a data block at the slave, in the response: mirrored.
  PROFIdrive: data block number 47 defined for parameter requests and for parameter response.
- Length:
  DPV1: In the Write request and Read response, the length of the transmitted data in bytes.
  PROFIdrive: Length of parameter request/parameter response.
  DPV1: In the Read request, the requested length of a data block.
  PROFIdrive: Maximum length possible
  DPV1: In the Write response, the data length accepted by the slave.
  PROFIdrive: Mirroring of the length from the Write request.

Error case:
If there is an error, the reply to a DPV1 Read or Write request is an error response.

5) DPV1 Error Response:

DPV1 Error	Function_Num = 0xDF (Error Write) = 0xDE (Error Read)	Error_Decode = 128 (DPV1)
	Error_Code_1	Error_Code_2 = (don't care⇒ always = 0)

- Error_Decode:
  DPV1: ID as to how Error_Code_1/2 are to be interpreted.
  PROFIdrive: always 128 (DPV1 codes)
- Error_Code_1:
  DPV1: Breakdown into error class (4 bits) and error code (4 bits); refer to Table 14.

PROFIdrive: Utilizing certain numbers

- Error_Code_2:
  DPV1: Application-specific
  PROFIdrive: not used (always = 0)

Table 14 DPV1 Error class and code for PROFIdrive

Error_Class (from DPV1 Spec)	Error_Code (from DPV1 Spec)	Application PROFIdrive
0x0..0x9 = reserved		
0xA = application	0x0 = read error 0x1 = write error 0x2 = module failure 0x3 to 7 = reserved 0x8 = version conflict 0x9 = feature not supported 0xA to 0xF = user specific	-
0xB = access	0x0 = invalid index 0x1 = write length error 0x2 = invalid slot 0x3 = type conflict 0x4 = invalid area 0x5 = state conflict 0x6 = access denied 0x7 = invalid range 0x8 = invalid parameter 0x9 = invalid type 0xA to 0xF = user specific	0xB0 = No data block DB47: parameter requests are not supported 0xB5 = Access to DB47 temporarily not possible due to internal processing status 0xB7 = Write DB47 with error in the DB47 header
0xC = resource	0x0 = read constraint conflict 0x1 = write constraint conflict 0x2 = resource busy 0x3 = resource unavailable 0x4..0x7 = reserved 0x8..0xF = user specific	-
0xD...0xF = user specific		

3.4.6. Data block lengths

If the data block length limitation of DPV1 (also refer to [1]) is used, then the following settings of the transferred data quantity are recommended which help to optimize the DP cycle.

Data quantities which can be transferred as a function of the maximum data block length.

Numerical data used as basis:

(Length in bytes)	Complete	Half	Quarter
Max. Profibus length	255	127	63
FDL user data	244	116	52
DPV1 user data	240	112	48
Parameter address and value	236	108	44
FDL header	11		
DPV1 header	4		
Parameter header	4		
Parameter address	6		
Parameter value header	2		

Formulas:

1) DPV1 user data length required as a function of the number of parameters and number of values
DPV1 user data length =
Parameter header (=4) +
Number of parameters * (Parameter address (=6) + Parameter value header (=2) +
Format(=1,2,4) * number of values)

2) Max. number of parameters for a specified DPV1 user data length (number of values = 1)
Number of parameters =
= (DPV1 user data length - Parameter header (=4))
/ (Parameter address(=6) + Parameter value header(=2) + Format(=1,2,4))

3) Maximum number of values for a specified DPV1 user data length (number of parameters = 1)
Number of elements =
= (DPV1 user data length - Parameter header(=4)
- Parameter address(=6) - Parameter value header(=2))
/ Format(=1,2,4)

Table 15 Limits due to the DPV1 data block length

Object	Task	Limited by	Data block length 240 byte		Data block length 112 byte		Data block length 48 byte	
			Word	D Word	Word	D Word	Word	D Word
One parameter, several elements	Request	Number of elements	117	58	53	26	21	10
	Change		114	57	50	25	18	9
Multi-parameters, each one element	Request	Number of parameters	39	39	18	18	7	7
	Change		23	19	10	9	4	3
Description complete or string	Request	Lengths in bytes	234		106		42	
	Change		228		100		36	
Text (length=16)	Request	Number of requests	14		6		2	
	Change		14		6		2	

3.4.7. Scalability of the functionality

- Multi-parameter accesses: yes/no
- Parameter description available: yes/no
  If no parameter description is available, the parameter description must be added as a file to the drive.
- DPV1 data block length: limitation of the data block length for the purpose of shorter bus cycles in the isochronous mode.

3.4.8. GSD parameters for DPV1

The following GSD parameters are relevant for the DPV1 parameter access:

Table 16 GSD parameters for the DPV1 parameter access

Parameter	Designation
DPV1_Slave	DPV1 support
C1_Max_Data_Len	Max. DPV1 data block length for the communication channel with the Class 1 master
C2_Max_Data_Len	Max. DPV1 data block length for the communication channel with the Class 2 master
C1_Read_Write_supp	Supports the read and write services of the Class 1 master
C2_Read_Write_supp	Supports the read and write services of the Class 2 master
C2_Max_Count_Channels	Maximum account of active C2 channels of the DPV1 slave

Note: A summarization of all relevant GSD entries for isochronous mode, data eXchange broadcast and DPV1 parameter access is given in Appendix A.3. For further information see [8].

A.9.4. Conclusion

Table 60 Examples of Configuration Elements in PROFIdrive Context

PROFIdrive_ Object	**Special_Cfg _ Identifier**	**Length_Bytes**		**Manufacturer_ Specific_Data**			**Comment**
		Output	**Input**	**PROFIdrive_ Code**	**Number**		
				Byte 1	**Byte 2**	**Byte 3**	
Axes Separator	0x01 / 1	-	-	0xFE / 254	-		Empty slot
Standard Telegram 1	0xC3 /195	0xC1 193	0xC1 193	0xFD / 253	0x0001 / 1		2 words output 2 words input
Standard Telegram 2	0xC3 /195	0xC3 195	0xC3 195	0xFD / 253	0x0002 / 2		4 words output 4 words input
Standard Telegram 3	0xC3 /195	0xC4 196	0xC8 200	0xFD / 253	0x0003 / 3		5 words output 9 words input
Standard Telegram 4	0xC3 /195	0xC5 197	0xCD 205	0xFD / 253	0x0004 / 4		6 words output 14 words input
Standard Telegram 5	0xC3 /195	0xC8 200	0xC8 200	0xFD / 253	0x0005 / 5		9 words output 9 words input
Standard Telegram 6	0xC3 /195	0xC9 201	0xCD 205	0xFD / 253	0x0006 / 6		10 words output 14 words input
Standard Telegram 7	0xC3 /195	0xC1 193	0xC1 293	0xFD / 253	0x0007 / 7		2 words output 2 words input
Standard Telegram 8	0xC3 /195	0xC4 196	0xC4 196	0xFD / 253	0x0008 / 8		5 words output 5 words input
Standard Telegram 9	0xC3 /195	0xC5 197	0xC4 196	0xFD / 253	0x0009 / 9		6 words output 5 words input
Standard Telegram 20	0xC3 /195	0xC1 193	0xC5 197	0xFD / 253	0x0014 / 20		2 words output 6 words input
Standard Telegram **X**	0xC3 /195	0xC**x-1** 191**+x**	0xC**y-1** 191**+y**	0xFD / 253	0x**X** / **X**		**x** words output **y** words input

A.10. DPV1 parameter channel

A.10.1. Examples for telegram sequences

Request parameter value, single

Parameter request:

Request header	Request reference	Request ID = Request parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Value	No. of elements = 0[1]	4
	Parameter number		
	Subindex = 0 (irrelevant)		
			10

[1] No. of elements = 0 for PKW compatibility. If No. of elements = 0, the Subindex is irrelevant. The recommended value of the Subindex is 0 in this case.

Parameter response positive with data of data type Word:

Response header	Request ref. Mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Word	No. of values = 1	4
	Value		6
			8

Parameter response positive with data of data type Double word:

Response header	Request ref. Mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Double word	No. of values = 1	4
	Value		6
			10

Parameter response, negative:

Response header	Request ref. Mirrored	Response ID = Request parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Change parameter value, single

Parameter request:

Request header	Request reference	Request ID = Change parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Value	No. of elements = 0[1]	4
	Parameter number		
	Subindex = 0 (irrelevant)		
Parameter value	Format = Word	No. of values = 1	10
	Value		12
			14

Parameter response, positive:

Response header	Request ref. mirrored	Response ID = Change parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
			4

[1] No. of elements = 0 for PKW compatibility. If No. of elements = 0, the Subindex is irrelevant. The recommended value of the Subindex is 0 in this case.

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Change parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Request parameter value, several array elements

Parameter request:

Request header	Request reference	Request ID = Request parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Value	No. of elements = 5	4
	Parameter number		
	Subindex = 0		
			10

Parameter response, positive:

Response header	Request ref. mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Word	No. of values = 5	4
	Value 1		6
	Value 2		
	Value 3		
	Value 4		
	Value 5		
			16

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Request parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Change parameter value, several array elements

Parameter request:

Request header	Request reference	Request ID = Change parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Value	No. of elements = 5	4
	Parameter number		
	Subindex = 125		
Parameter value	Format = Word	No. of values = 5	10
	Value 1		12
	Value 2		
	Value 3		
	Value 4		
	Value 5		
			22

Parameter response, positive:

Response header	Request ref. Mirrored	Response ID = Change parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
			4

Parameter response, negative:

Response header	Request ref. Mirrored	Response ID = Change parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Change parameter value, several array elements, Format Byte

Parameter request:

Request header	Request reference	Request ID = Change parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Value	No. of elements = 7	4
	Parameter number		
	Subindex = 110		
Parameter value	Format = Byte	No. of values = 7	10
	Value 1	Value 2	12
	Value 3	Value 4	
	Value 5	Value 6	
	Value 7	Dummy Byte	18

Parameter response, positive:

Response header	Request ref. Mirrored	Response ID = Change parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
			4

Parameter response, negative:

Response header	Request ref. Mirrored	Response ID = Change parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Request parameter value, multi-parameter

Parameter request:

Request header	Request reference	Request ID = Request parameter	0
	Axis = 0	No. of parameters = 3	2
1st parameter address	Attribute = Value	No. of elements = 1	4
	Parameter number		
	Subindex = 7		
2nd parameter address	Attribute = Value	No. of elements = 100	10
	Parameter number		
	Subindex = 0		
3rd parameter address	Attribute = Value	No. of elements = 2	16
	Parameter number		
	Subindex = 13		
			22

Parameter response (+): all partial accesses OK

Response header	Request ref. mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 3	2
1st parameter value(s)	Format = Word	No. of values = 1	4
	Value		6
2nd parameter value(s)	Format = Word	No. of values = 100	8
	Value 1		10
	Value 2		
	...		
	Value 100		
3rd parameter value(s)	Format = Double word	No. of values = 2	210
	Value 1		212
	Value 2		
			220

Parameter response (-): first and third partial access OK, second partial access erroneous

Response header	Request ref. mirrored	Response ID = Request parameter (-)	0
	Axis mirrored	No. of parameters = 3	2
1st parameter value(s)	Format = Word	No. of values = 1	4
	Value		6
2nd parameter value(s)	Format = Error	No. of values = 1	8
	Error value		10
3rd parameter value(s)	Format = Double word	No. of values = 2	12
	Value 1		14
	Value 2		
			22

Change parameter value, multi-parameter

Parameter request:

Request header	Request reference	Request ID = Change parameter	0
	Axis = 0	No. of parameters = 3	2
1st parameter address	Attribute = Value	No. of elements = 1	4
	Parameter number		
	Subindex = 7		
2nd parameter address	Attribute = Value	No. of elements = 100	10
	Parameter number		
	Subindex = 0		
3rd parameter address	Attribute = Value	No. of elements = 2	16
	Parameter number		
	Subindex = 13		
1st parameter value(s)	Format = Word	No. of values = 1	22
	Value		24
2nd parameter value(s)	Format = Word	No. of values = 100	26
	Value 1		28
	Value 2		
	...		
	Value 100		
3rd parameter value(s)	Format = Double word	No. of values = 2	228
	Value 1		230
	Value 2		
			238

Parameter response (+): all partial accesses OK

Response header	Request ref. mirrored	Response ID = Change parameter (+)	0
	Axis mirrored	No. of parameters = 3	2
			4

Parameter response (-):first and third partial access OK, second partial access erroneous

Response header	Request ref. mirrored	Response ID = Change parameter (-)	0
	Axis mirrored	No. of parameters = 3	2
1st parameter value(s)	Format = Zero	No. of values = 0	4
2nd parameter value(s)	Format = Error	No. of values = 2	6
	Error value		8
	Erroneous subindex		10
3rd parameter value(s)	Format = Zero	No. of values = 0	12
			14

Request description, single

Parameter request:

Request header	Request reference	Request ID = Request parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Description	No. of elements = 1	4
	Parameter number		
	Subindex = n		
			10

Parameter response positive with data of the data type word (e.g. ID):

Response header	Request ref. mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Word	No. of values = 1	4
	Value		6
			8

Parameter response positive with text:

Response header	Request ref. mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Byte	No. of values = 16	4
	Byte 1	Byte 2	6
	...	...	
	Byte 15	Byte 16	
			22

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Request parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Change description, single

Parameter request:

Request header	Request reference	Request ID = Change parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Description	No. of elements = 1	4
	Parameter number		
	Subindex = n		
Parameter value	Format = Word	No. of values = 1	10
	Value		12
			14

Parameter response, positive:

Response header	Request ref. mirrored	Response ID = Change parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
			4

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Change parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Request description, complete

Parameter request:

Request header	Request reference	Request ID = Request parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Description	No. of elements = 0 (irrelevant)	4
	Parameter number		
	Subindex = 0 (!)		
			10

Parameter response, positive:

Response header	Request ref. mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Byte	No. of values = (Bytes)	4
	ID		6
	(etc.)		
	...		
	...	...	
			6 + Description

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Request parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Change description, complete

Parameter request:

Request header	Request reference	Request ID = Change parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Description	No. of elements = 0 (irrelevant)	4
	Parameter number		
	Subindex = 0 (!)		
Parameter value	Format = Byte	No. of values = (in bytes)	10
	ID		12
	(etc.)		
	...		
	...	...	
			12 + Description

Parameter response, positive:

Response header	Request ref. mirrored	Response ID = Change parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
			4

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Change parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Request text, single

Parameter request:

Request header	Request reference	Request ID = Request parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Text	No. of elements = 1	4
	Parameter number		
	Subindex = n		
			10

Parameter response, positive:

Response header	Request ref. mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Byte	No. of values = 16	4
	Byte 1	Byte 2	6
	...	...	
	Byte 15	Byte 16	
			22

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Request description (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Change text, single

Parameter request:

Request header	Request reference	Request ID = Change parameter	0
	Axis = 0	No. of parameters = 1	2
Parameter address	Attribute = Text	No. of elements = 1	4
	Parameter number		
	Subindex = n		
Parameter value	Format = Byte	No. of values = 16	10
	Byte 1	Byte 2	12
	...	...	
	Byte 15	Byte 16	
			28

Parameter response, positive:

Response header	Request ref. mirrored	Response ID = Change parameter (+)	0
	Axis mirrored	No. of parameters = 1	2
			4

Parameter response, negative:

Response header	Request ref. mirrored	Response ID = Change parameter (-)	0
	Axis mirrored	No. of parameters = 1	2
Parameter value	Format = Error	No. of values = 1	4
	Error value		6
			8

Request parameter, multi-parameter, various attributes

Request of values, description and text in one request
Parameter request:

Request header	Request reference	Request ID = Request parameter	0
	Axis = 0	No. of parameters = 3	2
1st parameter address	Attribute = Value	No. of elements = 3	4
	Parameter number		
	Subindex = 0		
2nd parameter address	Attribute = Description	No. of elements = 0	10
	Parameter number		
	Subindex = 0		
3rd parameter address	Attribute = Text	No. of elements = 3	16
	Parameter number		
	Subindex = 0		
			22

Parameter response (+): all partial accesses OK

Response header	Request ref. mirrored	Response ID = Request parameter (+)	0
	Axis mirrored	No. of parameters = 3	2
1st parameter values (three values)	Format = Word	No. of values = 3	4
	Value 1		
	Value 2		
	Value 3		
2nd parameter values (complete description)	Format = Byte	No. of values = (Bytes)	12
	ID		
	(etc.)		
	...	...	
3rd parameter values (three texts)	Format = Byte	No. of values = 48	12 + Description
	Byte 1	Byte 2	
	...	...	
	Byte 47	Byte 48	
			62 + Description

A.10.2. Parameter access with DPV0

In IEC 61158, the definition of an array has been changed with respect to EN 50170.

The PROFIdrive Profile Version 2 is in conformance with EN 50 170. The subindex of a parameter with an index or an array respectively starts with index 1. In the actual IEC standard 61158, an access to a parameter with an index or an array respectively starts with index 0.

This means, in order to be in conformance with the IEC standard, the parameter model and the DPV1 parameter channel had to be adapted in the PROFIdrive Profile Version 3.

Devices which are implemented according to the new PROFIdrive Profile Version 3, shall have the following behavior regarding the characteristics at interfaces to a DPV1 client according to profile version 3 and a PKW client according to Profile version 2.

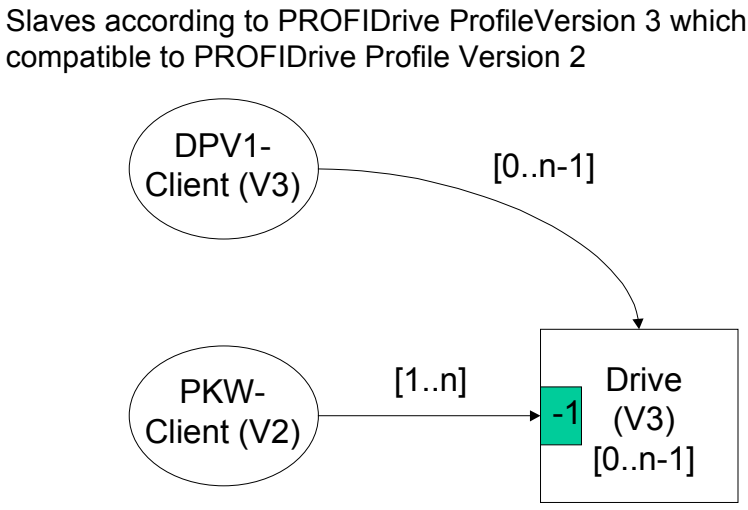


Fig. 65 Interface characteristics of the devices according to profile version 3

Note:
A device which was implemented in accordance with PROFIdrive Profile Version 3 must subtract or add an offset at the interface to a PKW client in accordance with the PROFIdrive Profile Version 2. Subtraction or addition respectively depends on whether the subindex of a parameter is in a request or response from the slave's perspective.

The following interface characteristics must to apply for devices according to profile version 2 which wish to use the DPV1 specifications according to profile version 3.

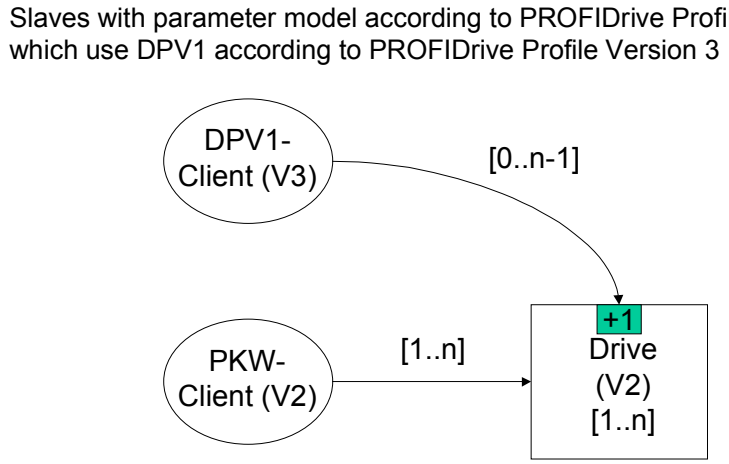


Fig. 66 Interface characteristics of devices according to profile version 2

Note:
A device which was implemented according to PROFIdrive Profile Version 2 and which uses DPV1 according to PROFIdrive Profile Version 3 must add or subtract an offset at the interface to a DPV1 client. Addition or subtraction respectively depends on whether the subindex of a parameter is in a request or response from the slave's perspective.

Compatibility to the PKW mechanism in PROFIdrive Profile version 2

In many existing drives, parameters are accessed via PKW (and possibly also internal interfaces are defined in the PKW format). For this reason, a table has been provided showing the relation between the DPV1 parameter requests to PKW (and vice versa).

Table 61 Comparison parameter requests DPV0/DPV1

	PKW accord. to PROFIdrive Profile V2	DPV1 Parameter Requests	Comment
Request reference	-	New! 8 bits	Identification of request/response
Request ID	Request/change value/descr./text 4 bits	Request/Change 8 bits	Differentiation value/description/text as additional attribute
No. of Parameters	-	New! 8 bits	Multi-parameter requests
Parameter Number	0..1999 (11 bits)	Content as PKW 16 bits	Parameter number = 0 is not permitted
Subindex	1..255 (8 bits)	Content as PKW -1 16 bits	The subindex is shifted due to a modified array definition: DPV1 subindex = PKW subindex - 1
No. of Elements	- (always "1")	New! 8 bits	
Attribute	-	New! 8 bits	Differentiation value/description/text
Total Length	2 words	5 words	

DPV1 parameter requests -> PKW

For “old” applications that can handle “new” DPV1 parameter requests.
External interface: DPV1 parameter requests
Internal interface: PKW

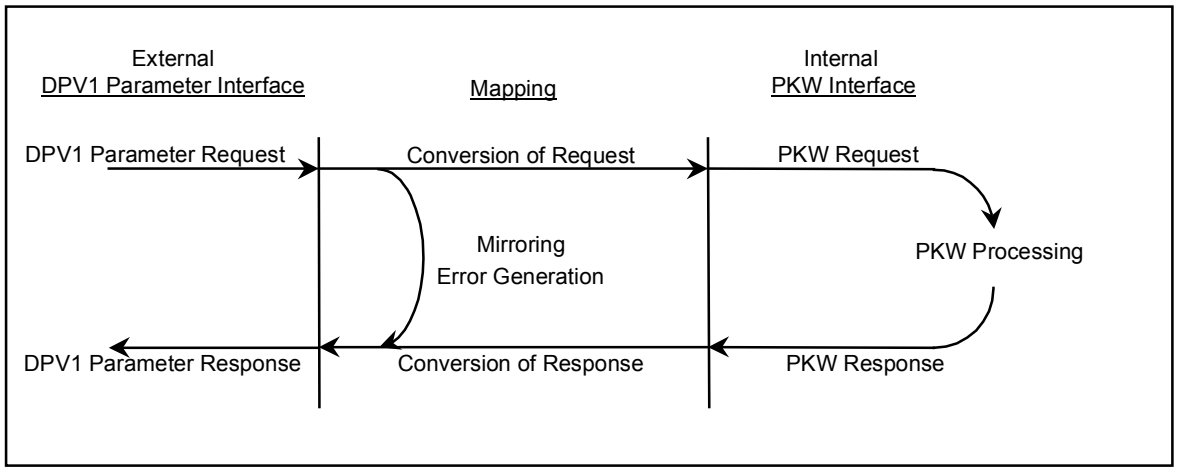


Fig. 67 DPV1 parameter requests -> PKW

In the following it is assumed that no multi-parameter requests are handled. However if they are handled, the procedure can be expanded correspondingly (execute PKW request n-times).

1) Evaluate DPV1 parameter request: Limit Range of values: since the PKW interface is narrower, the DPV1 parameter request is checked first for range of values limits that are not compatible with PKW. If exceeded, an error response is generated.

- No. of Parameters = 1
  Parameter Number ≤ 1999
  Subindex ≤ 116
  No. of Elements = 0 or 1

2) Generate PKW request:

- Request ID:
  To generate the different PKW requests (word, double word, array), the fields Attribute, No. of Elements, and Format of the DPV1 parameter request are also to be evaluated according to Table 62.
- Accept parameter number and subindex +1 directly

3) Evaluate PKW response:

- Generate a DPV1 error response if the PKW response does not match (internal error)

4) Generate DPV1 parameter response:

- Mirroring of the following fields from the DPV1 parameter request:
  Request reference, Request ID, No. of Parameters
- Generate the fields Response ID, Format, and Value according to Table 62.

Table 62 Convert DPV1 parameter request into a PKW request and generate a DPV1 parameter response

Evaluate DPV1 Parameter Request			Generate PKW Request ID	Check PKW Response ID	Generate DPV1 Parameter Response		
Request ID/ Attribute	No. of Ele-ments	Format			Response ID	Format	Value
Request / value	0	-	Request parameter value	Transmit parameter value (word)	positive	Word	From PKW Response
				Transmit parameter value (Double word)		Double word	
	1	-	Request parameter value (Array)	Transmit parameter value (Array word)		Word	
				Transmit parameter value (Array Double word)		Double word	
Change / value	0	Word	Change parameter value (Word)	Transmit parameter value (word)	positive	-	-
		Double word	Change parameter value (Double word)	Transmit parameter value (Double word)			
	1	Word	Change parameter value (Array word)	Transmit parameter value (Array word)			
		Double word	Change parameter value (Array Double word)	Transmit parameter value (Array Double word)			
Request / description	0	-	-	-	negative	Error	Error value
	1	-	Request description element	Transmit description element	positive	Byte	From PKW Response
Change / description	0	-	-	-	negative	Error	Error value
	1	-	Change description element	Transmit description element	positive	-	-
Request / text	-	-	-	-	negative	Error	Error value
Change / text	-	-	-	-	negative		
(any)			(any)	Error Response	negative		
(any)			(any)	Response does not match	negative		

PKW -> DPV1 parameter requests

For “new” applications that handle “old” PKW requests
External Interface: PKW
Internal Interface: DPV1 parameter requests

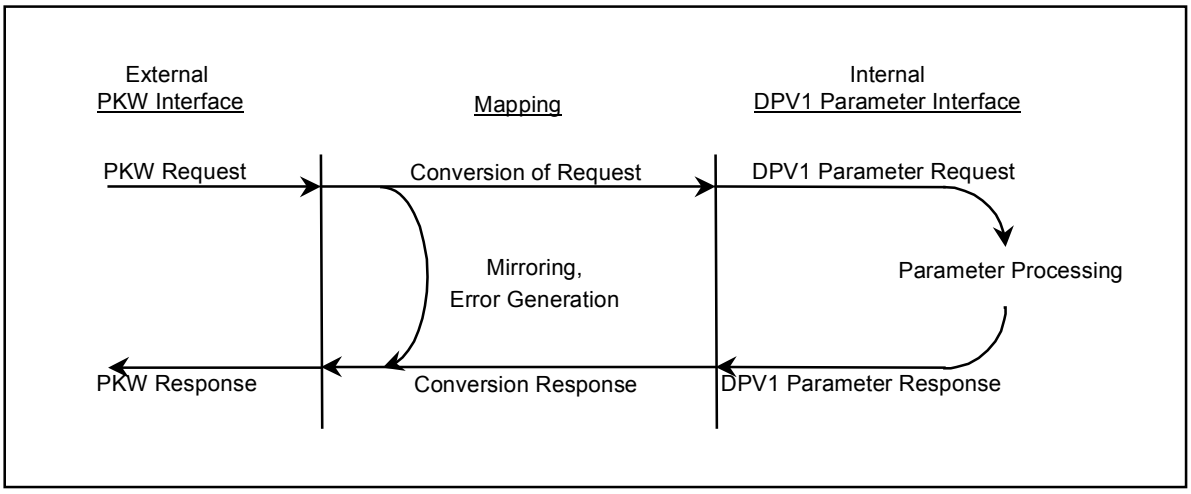


Fig. 68 PKW-> DPV1 parameter requests

1) Evaluate PKW request
- Short route: Respond to "no request" with "no response"

2) Generate DPV1 parameter request
- Generate request reference (counter, toggle, or constant only)
- Generate Request ID, Attribute, No. of Elements, and Format according to Table 63 from the PKW request ID
- No. of parameters = No. of values = 1
- Accept parameter number and subindex –1 (see Table 63) from PKW request

3) Evaluate DPV1 parameter response
- Generate a PKW error response if the DPV1 parameter response does not match

4) Generate PKW response
- Generate PKW response ID according to Table 63
- Mirror parameter number and subindex from PKW request
- Parameter value:
  If PKW request ID is “change”, then mirror from PKW request.
  If PKW request ID is “request”, then accept parameter response from DPV1.

Table 63 Convert PKW Request into a DPV1 request and Generating PKW response

Evaluate PKW Request ID	Generate DPV1 Parameter Request			Check DPV1 Parameter Response (Format)	Generate PKW Response ID
	Request ID/ Attribute	No. of Ele-ments	Format		
0: No request	-	-	-	-	No response
Request parameter value	Request / value	0	-	Word	Transmit parameter value (Word)
				Double word	Transmit parameter value (Double word)
Request parameter value (Array)		1	-	Word	Transmit parameter value (Array word)
				Double word	Transmit parameter value (Array double word)
Change parameter value (Word)	Change / value	0	Word	-	Transmit parameter value (Word)
Change parameter value (Double word)			Double word		Transmit parameter value (Double word)
Change parameter value (Array word)		1	Word		Transmit parameter value (Array word)
Change parameter value (Array double word)			Double word		Transmit parameter value (Array double word)
Request description element	Request/ descr.	1	-	-	Transmit description element
Change description element	Change/ descr.	1	Byte	-	
Request no. of array elements	-	-	-	-	Request not executable
(any)	(any)			(negative response)	Request not executable
(any)	(any)			(Response does not match)	Request not executable

7.4 Structure of the parameter job DB

7.4.1 Structure of the parameter job DB for FB PDAT_DL / PDAT_UD

7.4.1.1 Example of complete DB transfer

Start address

Separator between two jobs must not be changed

Structure of the DS 100 job (see Subsection 7.4.1.3)

DBW0	0001		Version identifier of DB	
DBW2	AA64	Job 1:	Identifier DS 100	
DBW4	AA64		Identifier DS 100	
DBW6			Parameter identifier (PKE) 1)	
DBW8			Index	
DBW10			PWE1	
DBW12			PWE2	
DBW14	AA64	Job 2:	Identifier DS 100	
DBW16	AA64		Identifier DS 100	
DBW18		Several indices	Parameter identifier (PKE) 1)	
DBW20			Index	
DBW22			PWE1	
DBW24			PWE2	
DBW26			PWE3	
DBW28			PWE4	
DBW30	EEEE		End identifier	
DBW32	EEEE		End identifier	
DBW34	001F 2)		Number of next DB: 31	

1) When the parameter identifier (PKE) is entered, the parameter number for "Page turning" (setting of Para_Page_Select-Bits in the index word) must be taken into account.

2) If no further data block follows, "0000" is to be entered here.

7.4.1.2 Example of partial DB transfer

Structure of the DS 100 job (see Section 7.2)
Structure of the DS47 job (see Section 7.3)

Start of first partial job (start address)

Separator between two jobs must not be changed

End of second partial job

Start of second partial job (start address)

Separator between two jobs must not be changed

End of second partial job

DBW0	0001		Version identifier of DB/partial job	
DBW2	AA64	Job 1:	Identifier DS 100	
DBW4	AA64		Identifier DS 100	
DBW6			Parameter identifier (PKE) 1)	
DBW8			Index	
DBW10			PWE1	
DBW12			PWE2	
DBW14	AA64	Job 2:	Identifier DS 100	
DBW16	AA64		Identifier DS 100	
DBW18		Several indices	Parameter identifier (PKE) 1)	
DBW20			Index	
DBW22			PWE1	
DBW24			PWE2	
DBW26			PWE3	
DBW28			PWE4	
DBW30	EEEE		End identifier	
DBW32	EEEE		End identifier	
DBW34	001F 2)		Number of next DB: 31	
DBW36	0001		Version identifier of DB/partial job	
DBW38	AA64	Job 1:	Identifier DS 100	
DBW40	AA64		Identifier DS 100	
DBW42			Parameter identifier (PKE) 1)	
DBW44			Index	
DBW46			PWE1	
DBW48			PWE2	
DBW50	AA64	Job 2:	Identifier DS 100	
DBW52	AA64		Identifier DS 100	
DBW54		Several indices	Parameter identifier (PKE) 1)	
DBW56			Index	
DBW58			PWE1	
DBW60			PWE2	
DBW62			PWE3	
DBW64			PWE4	
DBW66	EEEE		End identifier	
DBW68	EEEE		End identifier	
DBW70	001F 2)		Number of next DB: 31	

1) When the parameter identifier (PKE) is entered, the parameter number for "Page turning" (setting of Para_Page_Select-Bits in the index word) must be taken into account.

2) If no further data block follows, "0000" is to be entered here.

7.4.1.3 Note on the transfer of parameter value 16#AA64 / 16#AA64AA64 to the converter

If parameter value 16#AA64 is to be transferred to the converter, it may happen that the value cannot be transferred. The reason for this is that identifier 16#AA64AA64 is defined as a separator between two jobs. The table below contains a list of cases where problems may occur with parameter value 16#AA64 and how these may be avoided.

Case	Job identifier (PKE)	Index (IND)	Parameter value	Solution
1	2/14 Change parameter value (word)		PWE1 = 16#AA64	Enter another word with the value 0 between PWE1 and the separator
2	7/12 Change parameter value (array, word)	<>255	PWE1 = 16#AA64	Enter another word with the value 0 between PWE1 and the separator
3	3/13 Change parameter value (doubleword)		PWE1 <> 16#AA64 PWE2 = 16#AA64	Enter another word with the value 0 between PWE2 and the separator
4	3/13 Change parameter value (doubleword)		PWE1 = 16#AA64 PWE2 <> 16#AA64	Always possible
5	3/13 Change parameter value (doubleword)		PWE1 = 16#AA64 PWE2 = 16#AA64	This value cannot be transferred. The only possible remedy is to change the value.
6	8/11 Change parameter value (array, doubleword)	<>255	PWE1 <> 16#AA64 PWE2 = 16#AA64	Enter another word with the value 0 between PWE2 and the separator
7	8/11 Change parameter value (array, doubleword)	<>255	PWE1 = 16#AA64 PWE2 <> 16#AA64	Always possible
8	8/11 Change parameter value (array, doubleword)	<>255	PWE1 = 16#AA64 PWE2 = 16#AA64	This value cannot be transferred. The only possible remedy is to change the value.

Table 7-1 Changing a single parameter

Changing all the indices with indexed parameters (PKE = 7/12, IND = 255)

- If the value 16#AA64 is to be written to two consecutive indices, a value <> 16#AA64 must be written to the second index. A job must then be formulated in order to write 16#AA64 to this parameter.
- If 16#AA64 is to be written to the last index, proceed as described in Table 7-1 Case 1/2.

Changing all the indices with indexed parameters (PKE = 8/11, IND = 255)

- If 16#AA64AA64 is to be written to an index, proceed as described in Table 7-1 Case 8.
- If the values 16#xxxxAA64 and 16#AA64xxx are to be written to two consecutive indices, a value <> 16#AA64xxxx must be written to the second index. A job must then be formulated in order to write 16#AA64xxxx to this index.
- If 16#xxxxAA64 is to be written to the last index, proceed as described in Table 7-1 Case 6.

7.4.2 Structure of the parameter job DB for FB PDAT_UD2

7.4.2.1 Example of complete DB transfer

Start address →

DBB0	Version identifier 1 Converter	Main version	
DBB1		Subversion	
DBB2	Version identifier 2 Converter	Service pack	
DBB3		Hotfix	
Bit 4.0	Control information	= 1: US relevant; =0 US not relevant	
Bit 4.1		= 1: US relevant; =0 US not relevant	
Bit 4.2		No meaning	
Bit 4.3		No meaning	
Bit 4.4		No meaning	
Bit 4.5		No meaning	
Bit 4.6		No meaning	
Bit 4.7		No meaning	
Bit 5.0		No meaning	
Bit 5.1		No meaning	
Bit 5.2		No meaning	
Bit 5.3		No meaning	
Bit 5.4		No meaning	
Bit 5.5		No meaning	
Bit 5.6		=1 comparison value (DBB34) / wait time DBB35) for parameter transfer to EEPROM stored in DBW34	
Bit 5.7		=1 Previous DB exists; =0 Previous DB does not exist	
DBW6	Detailed info about DBx4.0	Parameter value for US system (-1 if unused)	
DBW8	Detailed info about DBx4.1	Parameter value for US system (-1 if unused)	
DBW10	Detailed info about DBx4.2	No meaning	
DBW12	Detailed info about DBx4.3	No meaning	
DBW14	Detailed info about DBx4.4	No meaning	
DBW16	Detailed info about DBx4.5	No meaning	
DBW18	Detailed info about DBx4.6	No meaning	
DBW20	Detailed info about DBx4.7	No meaning	
DBW22	Detailed info about DBx5.0	No meaning	
DBW24	Detailed info about DBx5.1	No meaning	
DBW26	Detailed info about DBx5.2	No meaning	
DBW28	Detailed info about DBx5.3	No meaning	
DBW30	Detailed info about DBx5.4	No meaning	
DBW32	Detailed info about DBx5.5	No meaning	
DBB34	Detailed info about DBx5.6	Comparison value for parameter transfer to EEPROM;	
DBB35		Maximum permissible wait time for parameter transfer to EEPROM to avoid function abort with error message	
DBW36	Detailed info about DBx5.7	No. of previous DB	
DBW38	Reserve 1		

Separator between two jobs must not be changed

DBW40	AA2F	Identifier DS 47	1st Job
DBW42	AA2F	Identifier DS 47	
DBW44	Job ID[1)]	Definition see below	
DBB46	Job reference		
DBB47	Job identifier		
DBB48	Axis		
DBB49	Number of parameters		
DBB50	Attribute		
DBB51	Number of elements		
DBW50	Parameter number		
DBW52	Subindex		
DBB54	Format		
DBB55	Number of values		
DBW56	Value		

Separator between two jobs must not be changed

DBW58	AA2F		...
DBW60	AA2F		
...	...		
DBWn	AA2F	Identifier DS 47	nth Job
DBW(n+2)	AA2F	Identifier DS 47	
DBW(n+4)	Job ID[1)]	Definition see below	
DBB(n+6)	Job reference		
DBB(n+7)	Job identifier		
DBB(n+8)	Axis		
DBB(n+9)	Number of parameters		
DBB(n+10)	Attribute		
DBB(n+11)	Number of elements		
DBW(n+12)	Parameter number		
DBW(n+14)	Subindex		
DBB(n+16)	Format		
DBB(n+17)	Number of values		
DBW(n+18)	Value		
DBW(n+20)	EEEE	End identifier	
DBW(n+22)	EEEE	End identifier	
DBW(n+24)	No. of next DB[2)]		

1) See table "Meaning of job IDs"
2) "0000" must be entered here if there are no further data blocks.

7.4.2.2 Example of partial DB transfer

Start of first partial job (start address) →

DBB0	Version identifier 1 Converter	Main version	
DBB1		Subversion	
DBB2	Version identifier 2 Converter	Service pack	
DBB3		Hotfix	
Bit 4.0	Control information	= 1: US relevant; =0 US not relevant	
Bit 4.1		= 1: US relevant; =0 US not relevant	
Bit 4.2		No meaning	
Bit 4.3		No meaning	
Bit 4.4		No meaning	
Bit 4.5		No meaning	
Bit 4.6		No meaning	
Bit 4.7		No meaning	
Bit 5.0		No meaning	
Bit 5.1		No meaning	
Bit 5.2		No meaning	
Bit 5.3		No meaning	
Bit 5.4		No meaning	
Bit 5.5		No meaning	
Bit 5.6		=1 comparison value (DBB34) / wait time DBB35) for parameter transfer to EEPROM stored in DBW34	
Bit 5.7		=1 Previous DB exists; =0 Previous DB does not exist	
DBW6	Detailed info about DBx4.0	Parameter value for US system (-1 if unused)	
DBW8	Detailed info about DBx4.1	Parameter value for US system (-1 if unused)	
DBW10	Detailed info about DBx4.2	No meaning	
DBW12	Detailed info about DBx4.3	No meaning	
DBW14	Detailed info about DBx4.4	No meaning	
DBW16	Detailed info about DBx4.5	No meaning	
DBW18	Detailed info about DBx4.6	No meaning	
DBW20	Detailed info about DBx4.7	No meaning	
DBW22	Detailed info about DBx5.0	No meaning	
DBW24	Detailed info about DBx5.1	No meaning	
DBW26	Detailed info about DBx5.2	No meaning	
DBW28	Detailed info about DBx5.3	No meaning	
DBW30	Detailed info about DBx5.4	No meaning	
DBW32	Detailed info about DBx5.5	No meaning	
DBB34	Detailed info about DBx5.6	Comparison value for parameter transfer to EEPROM;	
DBB35		Maximum permissible wait time for parameter transfer to EEPROM to avoid function abort with error message	
DBW36	Detailed info about DBx5.7	No. of previous DB	
DBW38	Reserve 1		

Separator between two jobs must not be changed

DBW40	AA2F	Identifier DS 47	1th Job
DBW42	AA2F	Identifier DS 47	
DBW44	Job ID[1)]	Definition see below	
DBB46	Job reference		
DBB47	Job identifier		
DBB48	Axis		
DBB49	Number of parameters		
DBB50	Attribute		
DBB51	Number of elements		
DBW50	Parameter number		
DBW52	Subindex		
DBB54	Format		
DBB55	Number of values		
DBW56	Value		
DBW58	AA2F		...
DBW60	AA2F		
...	...		

Separator between two jobs must not be changed

DBWn	AA2F	Identifier DS 47	nth Job
DBW(n+2)	AA2F	Identifier DS 47	
DBW(n+4)	Job ID[1)]	Definition see below	
DBB(n+6)	Job reference		
DBB(n+7)	Job identifier		
DBB(n+8)	Axis		
DBB(n+9)	Number of parameters		
DBB(n+10)	Attribute		
DBB(n+11)	Number of elements		
DBW(n+12)	Parameter number		
DBW(n+14)	Subindex		
DBB(n+16)	Format		
DBB(n+17)	Number of values		
DBW(n+18)	Value		

End of first partial job

DBW(n+20)	EEEE	End identifier	
DBW(n+22)	EEEE	End identifier	
DBW(n+24)	No. of next DB[2)]		

Start of second partial job (start address)

DBB(n+26)	Version identifier 1 Converter	Main version	
DBB(n+27)		Subversion	
DBB(n+28)	Version identifier 2 Converter	Service pack	
DBB(n+29)		Hotfix	
Bit(n+30).0	Control information	= 1: US relevant; =0 US not relevant	
Bit(n+30).1		= 1: US relevant; =0 US not relevant	
Bit(n+30).2		No meaning	
Bit(n+30).3		No meaning	
Bit(n+30).4		No meaning	
Bit(n+30).5		No meaning	
Bit(n+30).6		No meaning	
Bit(n+30).7		No meaning	
Bit(n+31).0		No meaning	
Bit(n+31).1		No meaning	
Bit(n+31).2		No meaning	
Bit(n+31).3		No meaning	
Bit(n+31).4		No meaning	
Bit(n+31).5		No meaning	
Bit(n+31).6		=1 comparison value (DBB34) / wait time DBB35) for parameter transfer to EEPROM stored in DBW34	
Bit(n+31).7		=1 Previous DB exists; =0 Previous DB does not exist	
DBW(n+32)	Detailed info about DBx4.0	Parameter value for US system (-1 if unused)	
DBW(n+34)	Detailed info about DBx4.1	Parameter value for US system (-1 if unused)	
DBW(n+36)	Detailed info about DBx4.2	No meaning	
DBW(n+38)	Detailed info about DBx4.3	No meaning	

DBW(n+40)	Detailed info about DBx4.4	No meaning	
DBW(n+42)	Detailed info about DBx4.5	No meaning	
DBW(n+44)	Detailed info about DBx4.6	No meaning	
DBW(n+46)	Detailed info about DBx4.7	No meaning	
DBW(n+48)	Detailed info about DBx5.0	No meaning	
DBW(n+50)	Detailed info about DBx5.1	No meaning	
DBW(n+52)	Detailed info about DBx5.2	No meaning	
DBW(n+54)	Detailed info about DBx5.3	No meaning	
DBW(n+56)	Detailed info about DBx5.4	No meaning	
DBW(n+58)	Detailed info about DBx5.5	No meaning	
DBB(n+60)	Detailed info about DBx5.6	Comparison value for parameter transfer to EEPROM;	
DBB(n+61)		Maximum permissible wait time for parameter transfer to EEPROM to avoid function abort with error message	
DBW(n+62)	Detailed info about DBx5.7	No. of previous DB	
DBW(n+64)	Reserve 1		
DBW(n+66)	AA2F	Identifier DS 47	1st Job
DBW(n+68)	AA2F	Identifier DS 47	
DBW(n+70)	Job ID[1)]	Definition see below	
DBB(n+72)	Job reference		
DBB(n+73)	Job identifier		
DBB(n+74)	Axis		
DBB(n+75)	Number of parameters		
DBB(n+76)	Attribute		
DBB(n+77)	Number of elements		
DBW(n+78)	Parameter number		
DBW(n+80)	Subindex		
DBB(n+82)	Format		
DBB(n+83)	Number of values		
DBW(n+84)	Value		
DBW(n+86)	AA2F		...
DBW(n+88)	AA2F		
...	...		
DBWm	AA2F	Identifier DS 47	mth Job
DBW(m+2)	AA2F	Identifier DS 47	
DBW(m+4)	Job ID[1)]	Definition see below	
DBB(m+6)	Job reference		
DBB(m+7)	Job identifier		
DBB(m+8)	Axis		
DBB(m+9)	Number of parameters		
DBB(m+10)	Attribute		
DBB(m+11)	Number of elements		
DBW(m+12)	Parameter number		
DBW(m+14)	Subindex		
DBB(m+16)	Format		
DBB(m+17)	Number of values		
DBW(m+18)	Value		
DBW(m+20)	EEEE	End identifier	
DBW(m+22)	EEEE	End identifier	
DBW(m+24)	No. of next DB[2)]		

Separator between two jobs must not be changed

Separator between two jobs must not be changed

End of second partial job

1) See table "Meaning of job IDs"
2) "0000" must be entered here if there are no further data blocks.

Meaning of job IDs

Job ID	Meaning
0	"Normal" job (job is executed without any restriction, whether it is a read job or a write job)
1	Job which prepares upload operation (needed to be able to read parameters from the drive). Jobs with this ID are always executed as write jobs and never converted to read jobs.
2	Read job "Units system (US) of target device" (needed to be able to compare the US of the source and target) If the target device has a different US to the source, the data are not downloaded and an error message is generated. If upload functionality is selected, the US of the target device is transferred to the DB.
3	Job which prepares download operation (needed to be able to change parameters in the drive) Jobs with this ID are positioned before "normal" parameter jobs in the DB. They are executed in the download direction only and skipped in the upload direction.
4	Job which terminates a download operation (terminates a write job) Jobs with this ID are positioned after the "normal" parameter jobs in the DB. They are executed only in the download direction and skipped in the upload direction.
5	Restore factory settings Jobs with this ID are executed only in the download direction and skipped in the upload direction.
6	Save parameters to EEPROM / Compact Flash Jobs with this ID are executed only in the download direction and skipped in the upload direction. At least two jobs are needed for this function, the first to initiate the save operation to the EEPROM and the second to check whether the save operation has been successful (depending on whether save to EEPROM operations are supported by the drive).

Meaning of control information bits and storage location of detailed information

Bit	Meaning
4.0	= 1: US relevant; =0 US not relevant; comparison value stored in DBW6
4.1	= 1: US relevant; =0 US not relevant; comparison value stored in DBW8
4.2	No meaning; comparison value stored in DBW10
4.3	No meaning; comparison value stored in DBW12
4.4	No meaning; comparison value stored in DBW14
4.5	No meaning; comparison value stored in DBW16
4.6	No meaning; comparison value stored in DBW18
4.7	No meaning; comparison value stored in DBW20
5.0	No meaning; comparison value stored in DBW22
5.1	No meaning; comparison value stored in DBW24
5.2	No meaning; comparison value stored in DBW26
5.3	No meaning; comparison value stored in DBW28
5.4	No meaning; comparison value stored in DBW30
5.5	No meaning; comparison value stored in DBW32
5.6	=1 Comparison value (DBB34)/wait time DBB35) for parameter transfer to EEPROM stored in DBW34
5.7	=1 Previous DB exists; =0 Previous DB does not exist; DB No. stored in DBW36.

Arrangement of job types in the parameter DB

1. **Jobs which prepare upload operation:** Needed to switch the drive to the upload state and to read out the units system of the target device (if relevant for the drive).
2. **Jobs to read out the units system of the target device** (if relevant for the drive): The value read out is compared with the corresponding value of the source parameter set. A download operation is started only if the US of the target and source sets are identical. With an upload operation, the US of the target system is stored in the parameter DB (see table above).
3. **Jobs which prepare download operation:** These jobs switch the drive to the download state.
4. **"Normal" jobs:** These jobs are executed in both the download and upload direction. In the upload direction, all write jobs are converted to read jobs.
5. **Jobs which terminate a download operation:** These jobs are needed to restore the drive to its original state before the download operation.
6. **Jobs which save parameters to EEPROM / Compact Flash:** This involves two separate jobs. The first is a write job which initiates transfer of the data to the EEPROM. The second is a read job to the same parameter which checks whether the data have been written successfully. The comparison value for the parameter and the wait time for the message which might be generated if the write operation has failed are stored in the detailed information for the control information bits.

7.4.2.3 Note on the transfer of parameter value 16#AA2F / 16#AA2FAA2F to the converter

If parameter value 16#AA2F is to be transferred to the converter, it may happen that the value cannot be transferred. The reason for this is that identifier 16#AA2FAA2F is defined as a separator between two jobs. The table below contains a list of cases where problems may occur with parameter value 16#AA2F and how these may be avoided.

Case	Data type	Value	Solution
1	8-bit	Last word in parameter value range = 16#AA2F	Enter a word with the value 0 between the last value and the separator
2	8-bit	Two consecutive words in parameter value range = 16#AA2F	Divide into two jobs and also enter a word with the value 0 between the last value and the separator
3	16-bit	Last word in parameter value range = 16#AA2F	Enter a word with the value 0 between the last value and the separator
4	16-bit	Two consecutive words in parameter value range = 16#AA2F	Divide into two jobs and also enter a word with the value 0 between the last value and the separator
5	32-bit	Last word in parameter value range = 16#AA2F	Enter a word with the value 0 between the last value and the separator
6	32-bit	Two consecutive words in parameter value range = 16#AA2F	This combination cannot be transferred. The only possible remedy is to change the value.

Table 7-2 Use of a data type

Combination of above data types:

- If the value 16#AA2F is being transferred as the last value, please ensure that it is separated from the separator by a word with the value 0.
- If the value 16#AA2F is to be written to two consecutive words in the parameter value range, it may happen that the value 16#AA2FAA2F does not result due to a change in the parameter sequence or division into two jobs. The previous point must still be observed, however.

7.5 Examples of download data blocks

7.5.1 Data block with DS 100 jobs

For structure of DS 100 jobs, see Section 7.2 "Formulating parameter jobs (data record 100)".

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DATA_00	WORD	W#16#100	DB version
+2.0	DATA_01	WORD	W#16#AA64	DS100 ID
+4.0	DATA_02	WORD	W#16#AA64	DS100 ID
+6.0	DATA_03	WORD	W#16#722A	Parameter ID (PKE): Control word "ON/OFF1"
+8.0	DATA_04	WORD	W#16#1	Index
+10.0	DATA_05	WORD	W#16#3100	Parameter value (PWE1)
+12.0	DATA_06	WORD	W#16#AA64	DS100 ID
+14.0	DATA_07	WORD	W#16#AA64	DS100 ID
+16.0	DATA_08	WORD	W#16#7191	Parameter ID (PKE): Fixed setpoint 1
+18.0	DATA_09	WORD	W#16#1	Index
+20.0	DATA_10	WORD	W#16#4000	Parameter value 1 (PWE1+PWE2 = 100%)
+22.0	DATA_11	WORD	W#16#0	Parameter value 2 (PWE1+PWE2 = 100%)
+24.0	DATA_12	WORD	W#16#AA64	DS100 ID
+26.0	DATA_13	WORD	W#16#AA64	DS100 ID
+28.0	DATA_14	WORD	W#16#71CE	Parameter ID (PKE): Ramp-up time
+30.0	DATA_15	WORD	W#16#FF	Index
+32.0	DATA_16	WORD	W#16#2000	Parameter value,index 1 (PWE1) = 50
+34.0	DATA_17	WORD	W#16#1000	Parameter value,index 2 (PWE2) = 25
+36.0	DATA_18	WORD	W#16#3000	Parameter value,index 3 (PWE3) = 75
+38.0	DATA_19	WORD	W#16#4000	Parameter value,index 4 (PWE4) = 100
+40.0	DATA_20	WORD	W#16#EEEE	End ID
+42.0	DATA_21	WORD	W#16#EEEE	End ID
+44.0	DATA_22	WORD	W#16#0	Number of next DB (0 = no further DB)
=46.0		END_STRUCT		

A1

A2

A3

A1: ON / OFF command from PROFIBUS interface, control word 1 bit0 = P554.1 (change parameter value (array, word))

A2: Input a fixed setpoint P401.1 = 100% (change parameter value (array, double word))

A3: Input ramp-up times P462 (change parameter value (array, word), several indices)

The two words with the value “W#16#AA64” between the parameter jobs (e.g A1 and A2) refer to an identification code for the download block to identify the individual jobs. They are not relevant to identify the data block and must not be changed.

Data block with DS 47 jobs

For structure of DS 47 jobs, see Section 7.3 "Formulating parameter jobs (data record 47)" and Subsection 7.4.1.3 for the structure of the DB.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DATA_0000	BYTE	B#16#1F	31
+1.0	DATA_0001	BYTE	B#16#0	00
+2.0	DATA_0002	BYTE	B#16#0	00
+3.0	DATA_0003	BYTE	B#16#0	0
+4.0	DATA_0004	BOOL	FALSE	= 0: ES not relevant
+4.1	DATA_0005	BOOL	FALSE	= 0: ES not relevant
+4.2	DATA_0006	BOOL	FALSE	No significance
+4.3	DATA_0007	BOOL	FALSE	No significance
+4.4	DATA_0008	BOOL	FALSE	No significance
+4.5	DATA_0009	BOOL	FALSE	No significance
+4.6	DATA_0010	BOOL	FALSE	No significance
+4.7	DATA_0011	BOOL	FALSE	No significance
+5.0	DATA_0012	BOOL	FALSE	No significance
+5.1	DATA_0013	BOOL	FALSE	No significance
+5.2	DATA_0014	BOOL	FALSE	No significance
+5.3	DATA_0015	BOOL	FALSE	No significance
+5.4	DATA_0016	BOOL	FALSE	No significance
+5.5	DATA_0017	BOOL	FALSE	No significance
+5.6	DATA_0018	BOOL	TRUE	Save to EEPROM
+5.7	DATA_0019	BOOL	FALSE	Previous DB does not exist
+6.0	DATA_0020	WORD	W#16#FFFF	Parameter value for ES (-1, if not used)
+8.0	DATA_0021	WORD	W#16#FFFF	Parameter value for ES (-1, if not used)
+10.0	DATA_0022	WORD	W#16#0	No significance
+12.0	DATA_0023	WORD	W#16#0	No significance
+14.0	DATA_0024	WORD	W#16#0	No significance
+16.0	DATA_0025	WORD	W#16#0	No significance
+18.0	DATA_0026	WORD	W#16#0	No significance
+20.0	DATA_0027	WORD	W#16#0	No significance
+22.0	DATA_0028	WORD	W#16#0	No significance
+24.0	DATA_0029	WORD	W#16#0	No significance
+26.0	DATA_0030	WORD	W#16#0	No significance
+28.0	DATA_0031	WORD	W#16#0	No significance
+30.0	DATA_0032	WORD	W#16#0	No significance
+32.0	DATA_0033	WORD	W#16#0	No significance
+34.0	DATA_0034	BYTE	B#16#0	Comparison value for Transfer parameter to EEPROM
+35.0	DATA_0035	BYTE	B#16#14	Delay time for Transfer parameter to EEPROM
+36.0	DATA_0036	WORD	W#16#0	Number of the previous DB (0 = no previous DB)
+38.0	DATA_0037	WORD	W#16#0	Reserved

+40.0	DATA_0038	WORD	W#16#AA2F	DS47 ID
+42.0	DATA_0039	WORD	W#16#AA2F	DS47 ID
+44.0	DATA_0040	INT	3	Job ID = 3: DL preparing job
+46.0	DATA_0041	WORD	W#16#102	Job reference, job ID (write parameter)
+48.0	DATA_0042	WORD	W#16#3	Axis, Number of parameters (= 3)
+50.0	DATA_0043	WORD	W#16#1001	Attribute (10 = value), Number of elements (= 1)
+52.0	DATA_0044	WORD	W#16#3	Parameter no. (p3 = User access level)
+54.0	DATA_0045	WORD	W#16#0	Subindex (= 0)
+56.0	DATA_0046	WORD	W#16#1001	Attribute (10 = value), Number of elements (= 1)
+58.0	DATA_0047	WORD	W#16#F6E	Parameter no. (p3950 = Access of hidden parameter)
+60.0	DATA_0048	WORD	W#16#0	Subindex (= 0)
+62.0	DATA_0049	WORD	W#16#1001	Attribute (10 = value), Number of elements (= 1)
+64.0	DATA_0050	WORD	W#16#A	Parameter no. (p10 = Commissioning parameter filter)
+66.0	DATA_0051	WORD	W#16#0	Subindex (= 0)
+68.0	DATA_0052	WORD	W#16#301	Format (3 = INT), Number of values (= 1)
+70.0	DATA_0053	INT	4	Value (4)
+72.0	DATA_0054	WORD	W#16#301	Format (3 = INT), Number of values (= 1)
+74.0	DATA_0055	INT	46	Value (46)
+76.0	DATA_0056	WORD	W#16#301	Format (3 = INT), Number of values (= 1)
+78.0	DATA_0057	INT	29	Value (29)
+80.0	DATA_0058	WORD	W#16#AA2F	DS47 ID
+82.0	DATA_0059	WORD	W#16#AA2F	DS47 ID
+84.0	DATA_0060	INT	0	Job ID = 0: Normal job
+86.0	DATA_0061	WORD	W#16#202	Job reference, job ID (write parameter)
+88.0	DATA_0062	WORD	W#16#1	Axis, Number of parameters (= 1)
+90.0	DATA_0063	WORD	W#16#1001	Attribute (10 = value), Number of elements (= 1)
+92.0	DATA_0064	WORD	W#16#5	Parameter no. (p5 = Display selection)
+94.0	DATA_0065	WORD	W#16#0	Subindex (= 0)
+96.0	DATA_0066	WORD	W#16#601	Format (6 = WORD), Number of values (= 1)
+98.0	DATA_0067	WORD	W#16#15	Value (21)
+100.0	DATA_0068	WORD	W#16#AA2F	DS47 ID
+102.0	DATA_0069	WORD	W#16#AA2F	DS47 ID
+104.0	DATA_0070	INT	0	Job ID = 0: Normal job
+106.0	DATA_0071	WORD	W#16#202	Job reference, job ID (write parameter)
+108.0	DATA_0072	WORD	W#16#2	Axis, Number of parameters (= 2)
+110.0	DATA_0073	WORD	W#16#1001	Attribute (10 = value), Number of elements (= 1)
+112.0	DATA_0074	WORD	W#16#2BD	Parameter no. (p701 = Selection digital inputl)
+114.0	DATA_0075	WORD	W#16#0	Subindex (= 0)
+116.0	DATA_0076	WORD	W#16#1001	Attribute (10 = value), Number of elements (= 1)
+118.0	DATA_0077	WORD	W#16#2BE	Parameter no. (p702 = Selection digital input2)
+120.0	DATA_0078	WORD	W#16#0	Subindex (= 0)
+122.0	DATA_0079	WORD	W#16#301	Format (3 = INT), Number of values (= 1)
+124.0	DATA_0080	INT	1	Value (1)
+126.0	DATA_0081	WORD	W#16#301	Format (3 = INT), Number of values (= 1)
+128.0	DATA_0082	INT	12	Value (12)

A1, A2, A3

A1: Jobs preparing for download in order to put the converter / inverter (here MICROMASTER 4xx) into a state that is capable of downloading

A2: Set display P5 to "Actual frequency"

A3: Multi-parameter job: Interconnect digital input 1 to "ON / OFF"
Interconnect digital input 2 to " Reversing"

The two words with the value “W#16#AA2F” between the parameter jobs (e.g A1 and A2) refer to an identification code for the download block to identify the individual jobs. They are not relevant to identify the data block and must not be changed.

+200.0	DATA_0118	WORD	W#16#AA2F	DS47 ID
+202.0	DATA_0119	WORD	W#16#AA2F	DS47 ID
+204.0	DATA_0120	INT	6	Job ID = 6: Save parameter to EEPROM
+206.0	DATA_0121	WORD	W#16#501	Job reference, job ID (read parameter)
+208.0	DATA_0122	WORD	W#16#1	Axis, Number of parameters (= 1)
+210.0	DATA_0123	WORD	W#16#1001	Attribute (10 = value), Number of elements (= 1)
+212.0	DATA_0124	WORD	W#16#3CB	Parameter no. (p971 = Transfer data from RAM to EEPROM)
+214.0	DATA_0125	WORD	W#16#0	Subindex (= 0)
+216.0	DATA_0126	WORD	W#16#301	Format (3 = INT), Number of values (= 1)
+218.0	DATA_0127	INT	1	Value (1)
+220.0	DATA_0128	WORD	W#16#EEEE	END ID
+222.0	DATA_0129	WORD	W#16#EEEE	END ID
+224.0	DATA_0130	WORD	W#16#0	0 = Number of the following DB (0 - no further DB)
=226.0		END_STRUCT		

A6

A4: Jobs completing download to put the converter/inverter (here MICROMASTER 4xx) into a state that is ready for operation.

A5: Store changed parameters in the EEPROM

A6: Read job in order to detect when parameters have been stored in the EEPROM

7.6 Addressing drive objects with DS 47 parameter jobs

In the request header of a parameter job formulated according to data record 47, the address of the target axis or target object of the drive must be entered in the "Axis" field. The values for these so-called module IDs are listed below for the individual drive types.

7.6.1 MICROMASTER 4xx

	Module ID / Axis
Single-axis drive	0

7.6.2 SINAMICS G

	Module ID / Axis
Control unit (CU320)	1
Drive (vector)	2
Customer terminal strip TM31 (-A60)	3
Customer terminal strip TM31 (-A61)	4

7.6.3 SINAMICS S

The module ID for individual drive objects can be found in CU parameter 101 (or 978).

7.6.4 MASTERDRIVES / DC-MASTER with CBP2, firmware version V2.20 and later

	Module ID / Axis
Single-axis device	0

7.6.5 SIMODRIVE 611U

Basic requirement for DS 47 communication:

- SIMODRIVE 611U and PROFIBUS DP2 / DP3 option module, firmware version 07.01 or later in each case

	Module ID / Axis
Axis 1	0
Axis 2	1

7.6.6 POSMO SI/CA/CD

Basic requirement for DS 47 communication:

- POSMO SI/CA/CD, firmware version 07.01 or later

	Module ID / Axis
Single-axis device	0

7.7 Tip

Example of how to generate a copy DB (8192 bytes in size) in the main memory or DB in the load memory using download function block FB40 or the upload / download function blocks FB41 / FB42.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	data	ARRAY[1..4096]		
*2.0		WORD		
=8192.0		END_STRUCT		

8 Abbreviations

ADR	Address byte in the frame
ANZ	Error byte in the pre-assignment block
CBP	Communication Board PROFIBUS
CFC	Continuous Function Chart
CP	Communications processor
CU	Basic module of MASTERDRIVES
DB	Data block
DBB	Data byte
DBW	Data word
DP	Decentralized peripheral units
Drive ES	Drive Engineering System
DS	Data record
DS47	Parameter interface compliant with PROFIBUS Profile Drive Technology, V3.1, November 2002
DS100	Parameter interface compliant with USS Protocol Specification (PIV mechanism)
FB	Function block
FC	S7 function (block in the SIMATIC S7)
FC	Frequency control (closed-loop control implementation of MASTERDRIVES)
FUP	Function plan
HIW	Main actual value
HLG	Start-up encoder in drive
HSW	Main setpoint
IND	Index in the PIV interface of the net data structure
INT	Interrupt
KSTW	Communication control word
MB	Flag byte
MD	MASTERDRIVES
MM	MICROMASTER
MW	Flag word
OB	Organization block
OM	Object manager
PAFE	Parameterizing error
PKE	Parameter ID in the net data structure
PIV	Parameter ID value
PMU	Parameterization unit for MASTERDRIVES
PPO	Parameter process data object
PWE	Parameter value in the net data structure
PZD	Process data

SC	Servo Control (control variant of MASTERDRIVES)
SCL	Structured Control Language
STW	Control word in PPO
STL	Statement list
UDT	User-defined data type
US	Units system
USS	Universal serial interface
VAT	Variables table
VC	Vector control (closed-loop control implementation of MASTERDRIVES)
WDH	Number of repeats permitted for a PIV job
ZSW	Status word in PPO

Siemens AG
Bereich Automatisierungs- und Antriebstechnik (A&D)
Geschäftsgebiet Motion Control Systems (MC)
Postfach 3180, D-91050 Erlangen