Table of contents

1. General description

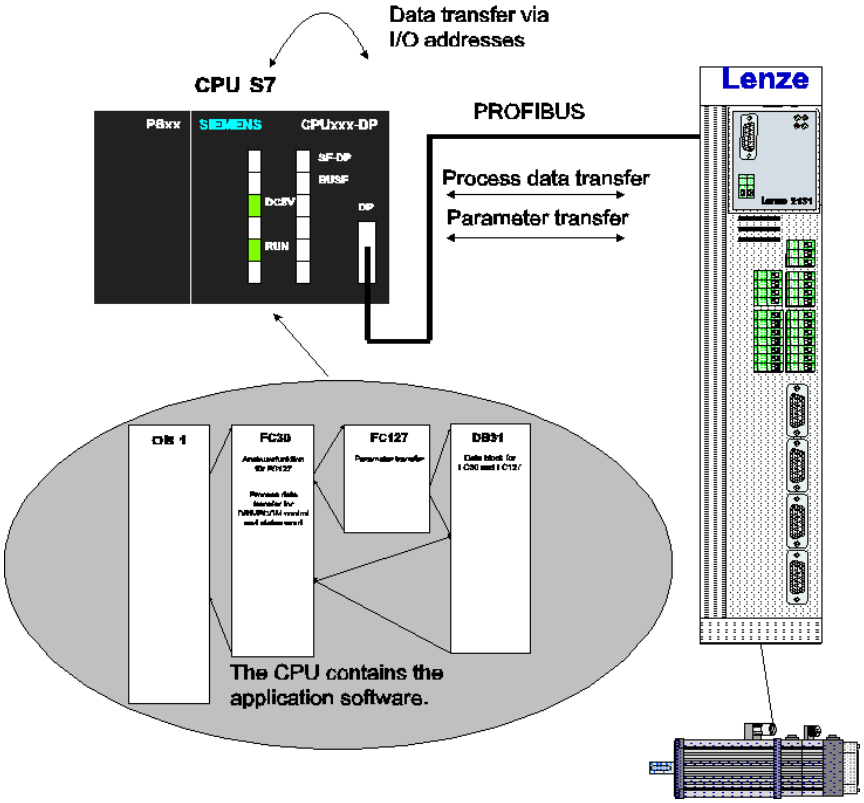
In conjunction with PROFIBUS, the blocks for S7 are generally applicable application examples from Lenze. The blocks provide programming and project planning support for PROFIBUS DP applications with Lenze products and suggest possible solutions. They are a free of charge Lenze product. Customers are not entitled to claim any warranty or updates. We do not accept any responsibility or liability for any damage that may occur through the use of these blocks.

Purpose:

The PROFIBUS interface has a parameter channel and a process data channel. The supplied function blocks support parameter setting and the sending/receiving of process data.

Parameter channel: The parameter channel is used to read and write parameter data. This data is not time-critical. Parameter data may, for example, be operating parameters (e.g. C11 – n_{max}), diagnostics information (e.g. C161 – current error) or motor data (e.g. C15 – V/f rated frequency).

Process data channel: Process data must be very quickly cyclically exchanged. Data such as the control word, status word, setpoint and actual values is transmitted. The size of the process data channel can be selected via the hardware configuration.



Hardware requirements:

- Drive controllers with PROFIBUS interface (e.g. 2131, 2133, PROFIBUS function block)
- S7 CPU 3xx/4xx with a PROFIBUS interface

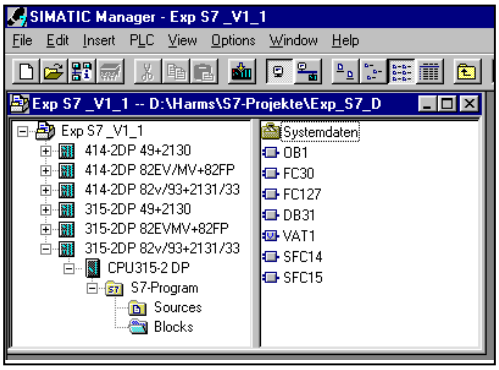
2. Installing the S7 program / blocks

2.1 Integrating the project file

1. Select ⇒ **File** ⇒ **Retrieve** in the S7 Manager.
2. Select the supplied file **E_S7V1_2.zip**.
3. Select the target directory (e.g. sub-directory "\Step7\s7proj").

A project consists of several components and may contain one or more stations (S7 300/400). Each station has a CPU (e.g. 315-2 DP) and one or more S7 programs. Each program has its own block folder, source folder and a symbol table.

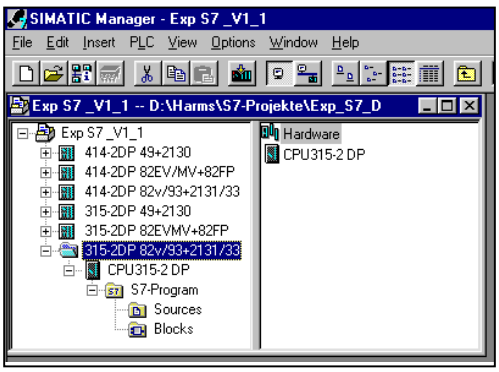
Now, the project can be selected by clicking the menu item ⇒ **Open** in the "SIMATIC Manager". A new project can be created under the menu item ⇒ **New**.



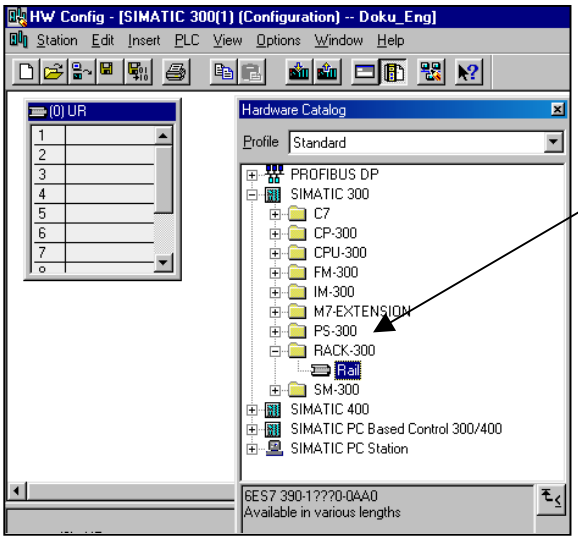
This project has sub-projects that consist of CPU 315-2 DP and CPU 414-2 DP and the different PROFIBUS modules (2130, 2131, etc.). All projects use the same blocks. These blocks have different process data lengths. The below application example describes operation with only one PROFIBUS slave. If more than one participant is involved, each block may be used once per participant. As the blocks must not have the same block numbers, they must be renamed for each participant.

2.2 Hardware configuration

The below example explains how to create a new hardware configuration. In this example, Step 7 version 5.1 + Hotfix 2, version K 5.1.0.2 are used.

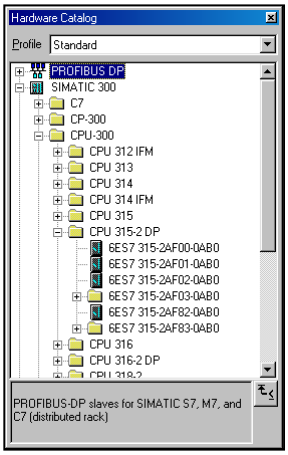


Once you have created a new project and have added a new station (S7-300/400) by selecting the menu item ⇒ **Insert** ⇒ **Station**, double-click **Hardware** in the right-hand window to open the hardware configuration.



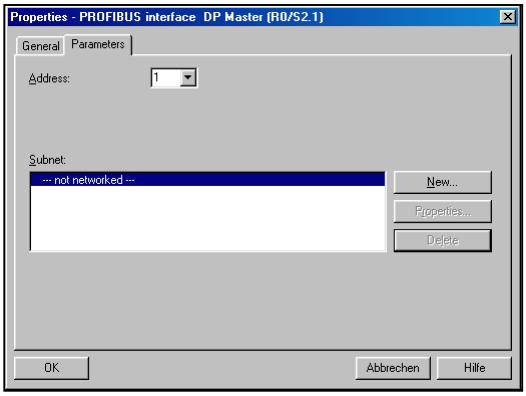
Unless the Hardware Catalog is displayed, select it by clicking the menu items ⇒ **View** ⇒ **Catalog**.

Select “Rail” (mounting rail) from the catalog folder **Rack** and drag and drop it to the left-hand window.

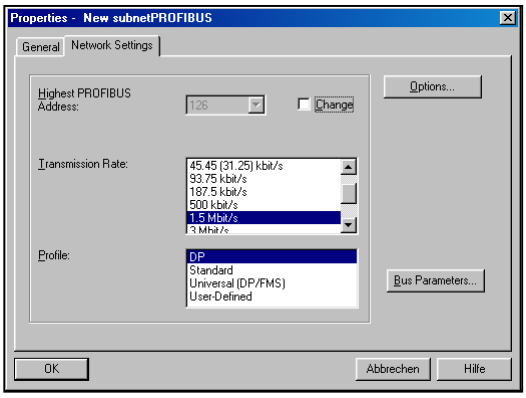


Select the power supply unit and the CPU via drag and drop from the catalog.

Note: ⇒ Under CPU 315-2 DP, the catalog lists several device types to choose from. Always compare the version number of your CPU with the version number stated in the catalog to avoid hardware conflicts.
The block version number is indicated on the side walls.



If you use a CPU with a DP Master, the “Properties – PROFIBUS Interface DP Master” dialog box opens automatically.
In this dialog box, the DP Master address is set and a PROFIBUS network can be created by clicking the button ⇒ **New**.



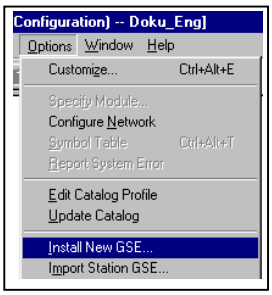
In the dialog box that opens the baud rate can be set in the Network Settings register.
Once the baud rate has been set, confirm all windows by clicking **OK**. Now, your hardware configuration consists of a power supply, a DP master CPU and a PROFIBUS network.

2.3 GSE files

If you are for the first time working with Lenze drives in conjunction with PROFIBUS or want to insert new GSE files, you first have to import them to the hardware catalog.

⇒ Note: The latest GSE files are available free of charge for downloading on the Lenze homepage in the Internet.

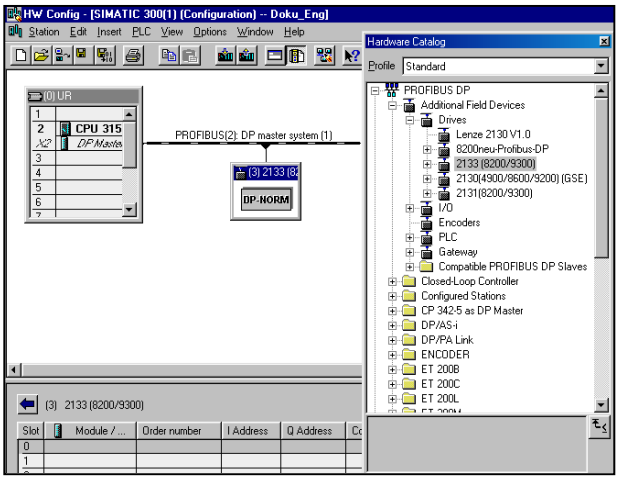
www.lenze.com (⇒ Service ⇒ Download ⇒ PROFIBUS)



Installing the GSE files:

Make sure that there is no configuration loaded in the hardware configuration tool. Select ⇒ **Options** ⇒ **Install New GSE**. Select the new GSE files from the dialog box that opens and confirm by clicking OK.

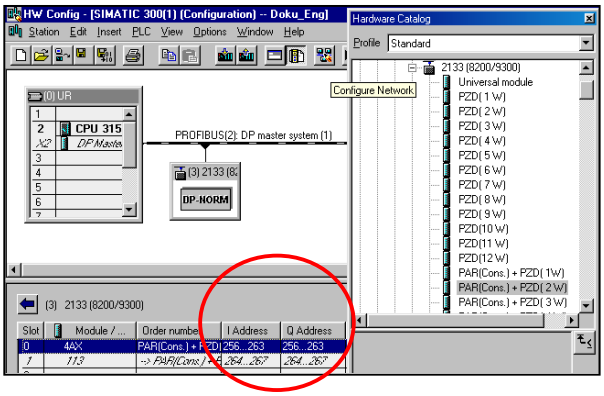
As an alternative, you may also copy all new GSE files to the ⇒ **Siemens** ⇒ **Step7** ⇒ **S7data** ⇒ **gse** folder and click the menu item ⇒ **Options** ⇒ **Update Catalog** in the hardware configuration tool.



Generating the DP slave:

The Lenze DP slave modules can be found in the Hardware Catalog under ⇒ **PROFIBUS DP** ⇒ **Additional Field Devices** ⇒ **Drives**.

To integrate a slave the whole folder (2133 (8200/9300) must be dragged and dropped to the PROFIBUS. A dialog box opens in which the PROFIBUS address is set. In this example, the module 2133 has the address 3.



A click on the plus sign in front of the selected drive module opens a list from which you can choose the supported process data formats. Highlight the DP slave and drag and drop it to the table below the planned hardware.

Two columns were completed in the appropriate table for the slave.
The first column shows the parameter channel and the second the process data channel.
The I/O address columns (red circle) are important.

⇒ Note: Various process data formats are available for communication.

<u>Extract from the Hardware Catalog:</u>

Hardware Catalog
" PROFIBUS DP
:
" Additional Field Devices
" Drives
Lenze 2130 V1.0
! 8200neu-Profibus DP
" 2133 (8200/9300)
:
Par(cons.) + PCD(1W) **AR**
:
PPO1
:
Par(cons.) + PCD(1W)
:
! 2130 (4900/8600/9200) (GSE)
! 2131 (8200/9300)

Profile	Name	Comment
AR	Lenze device control	direct access to AIF CTRL
PPO	Profidrive	Profidrive status machine
No identifier	Drivecom	Drivecom status machine

For the control and status word assignment, please refer to the Operating Instructions for the module concerned.

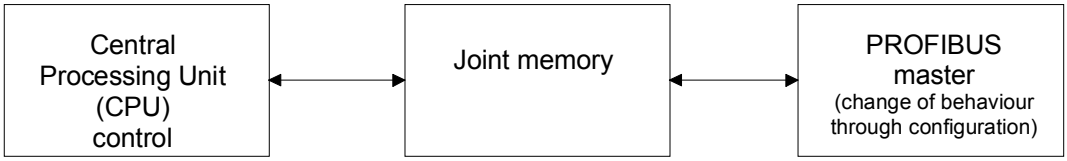
The designation (cons.) means that data (parameters and/or process data) is consistently transmitted.

⇒ Note: If parameters with the FC127 function are transmitted, data consistency is important!

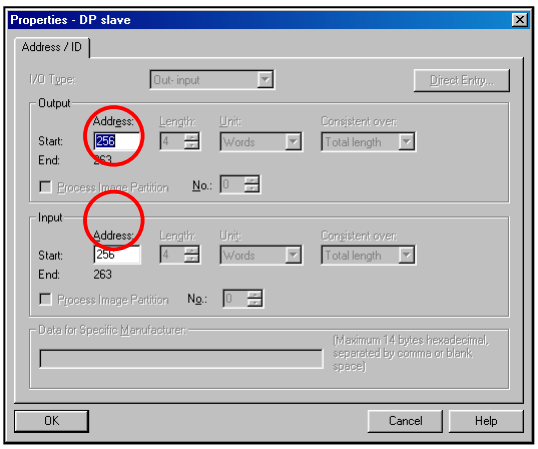
What does “consistency” mean?

The term “consistency” is used, if a range of more than one word or byte contains related data. In this case, enabling “consistency” ensures faultless data exchange between the central processing unit (CPU) of the control and the PROFIBUS master via the joint memory (dual port memory). This means, unless “consistency” is enabled, non related data might be entered in the CPU in a read request. This can lead to faulty information.

Example: a parameter is to be read



Waits until all data has been read. Then, the complete data set is written to the joint memory. Only writes, if the CPU does <u>not</u> read.

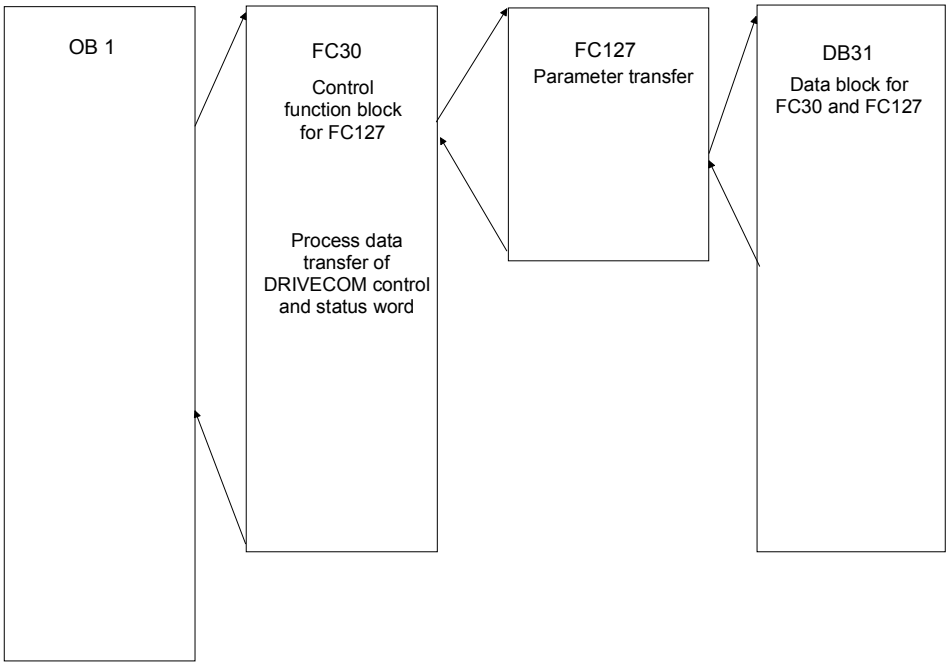


The I/O start addresses are required in the PLC program. They can also be changed.

Double-clicking the corresponding column opens a dialog box in which the address range can be changed.
The address range for one column (input and output address) should always be the same (red circles). This is clearer and makes working with the application program easier.

2.4 Subsequent call of blocks

The OB1 organisation block cyclically calls the FC30 function block.



The FC30 function is an example and shows the basic requirements for parameter and process data transfer.

In addition, the FC30 function block writes the DRIVECOM control word via the process data channel and reads the DRIVECOM status word. The function block operates in accordance with the DRIVECOM status machine and, in this way, initiates controller enable via the PROFIBUS provided that the DRIVECOM status machine has been activated. Furthermore, the FC30 block shows how the FC127 function is used and how PROFIBUS parameters can be read or written.
The codes to be read or written are entered in data block DB31.

The system blocks SFC 14 and SFC 15 have to be loaded for parameter transfer. They are required for consistent data transfer.

3. Function block description

Below, the individual blocks are described. The blocks are written in STL (Statement List).

3.1 DRIVECOM profile

The FC30 block supports the DRIVECOM profile. It assumes that the first word of the process data channel is the DRIVECOM control or status word. According to this profile, a certain sequence has to be observed upon controller enable. The FC30 processes this sequence automatically. Detailed information about the functionality of the DRIVECOM control word can be found in the next chapter.

3.1.1 DRIVECOM control word

The FC30 network numbers (NW) in which the bits are processed are written in brackets.

Bit	0 := Switch on	**<- addressed by the function block (NW 6)** (should not be addressed by the programmer)
	1 := Inhibit voltage	**<- addressed by the function block (NW 6)** (should not be addressed by the programmer)
	2 := Quick stop	**<- addressed by the function block (NW 6)** (should not be addressed by the programmer)
	3 := Enable operation	=> controller inhibit: = 0; controller enable: = 1, **(NW 3)**
	4 := Inhibit ramp generator	=> QSP := 0; QSP not active: = 1, **(NW 5)**
	5 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133) **IMPORTANT! Default setting for most controllers:** **Stop RFG : = 0; stop RFG not active: = 1 (NW 4)**
	6 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133) **IMPORTANT! Default setting for most controllers:** **RFG zero: = 0; RFG zero not active: = 1 (NW 4)**
	7 := Reset fault	Bit change from 0 to 1
	8 := DRIVECOM reserved	
	9 := DRIVECOM reserved	
	10 := DRIVECOM reserved	
	11 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	12 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	13 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	14 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	15 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)

3.1.2 DRIVECOM status word

The DRIVECOM status word shows the current device status.

Bit	0 : = Ready to start	**-> Information for the function block**
	1 := Switched on	**-> Information for the function block**
	2 := Operation enabled	**-> Information for the function block**
	3 := Fault	<= no fault (TRIP) := 0; fault (TRIP) := 1
	4 := Voltage inhibited	**-> Information for the function block**
	5 := Quick stop	DRIVECOM quick stop := 0; no quick stop := 1
	6 := Switch-on inhibit	**-> Information for the function block**
	7 := Warning	no warning := 0; warning := 1
	8 := Message	no message := 0; message := 1
	9 := Remote	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	10 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	11 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	12 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	13 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	14 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)
	15 := FREE DRIVECOM	see PROFIBUS Operating Instructions (e.g. 2131, 2133)

The free bits of the control or status word can be freely assigned. Before assignment, please read the relevant PROFIBUS Operating Instructions since the bit assignment may be different from controller to controller.

Which Operating Instructions describe what?

Fieldbus module Operating Instructions	PROFIBUS module for controllers:
2130	4800, 4900, 8600 and 9200
2131, 2133	8200 and 9300
“Fieldbus function modules for frequency inverters”	8200vector and 8200motec

3.2 FC30 function

The FC30 function is either called in OB1 or in another block. Below, the function is explained step-by-step using FC30 with the function module 2131 as an example.

Note: If consistent data transfer is required, the system blocks SFC 14 and SFC 15 have to be loaded in the PLC.

The block uses two temporary data structures in the variables table.

data_receive (receiving data)
data_send (sending data)

In the below example, the FC30 block is designed for communication with 3 process data words (PCD). In the variables table, each structure comprises 6 bytes or 3 words.

Note: Please refer to the Operating Instructions for your PROFIBUS master to see if data are transferred in byte or word mode.

The variables table must be adjusted to the number of process data words.

Example:

Address	Declaration	Name	Type	Comment
+2.0	temp	processdataword_1	INT	process data word 1 <- controller
+4.0	temp	processdataword_2	INT	process data word 2 <- controller
+6.0	temp	processdataword_3	INT	process data word 3 <- controller
+8.0	temp	processdataword_4	INT	process data word 4 <- controller
=8.0			END_STRUCT	

The table shows the data_receive structure with 4 PCDs. The same has to be repeated for the data_send structure.

FC30 consists of a total of 9 networks and supports the DRIVECOM profile. The individual networks are explained below.

3.2.1 Network 1, process data ⇐ controller

Network 1
```
// 1.################################################
// Consistent transfer via 1 process data word
// Process data word 0 (status word)
// L PEW 264        // 1st I/O address for process data transfer
// T LW 0           // temp. local variable
// Process data word 1 (actual value)
// L PEW  266
// T #data_receive.processdataword_1  // temp. local variable

// 2.################################################
// Consistent transfer via 2 process data words
// L PED  264       // 1st I/O address for process data transfer
// T LD 0           // temp. local variable

// 3.#################################################################
// Consistent transfer via 2 and more process data words
CALL  "DPRD_DAT"
LADDR  :=W#16#108      // 1st I/O address for process data transfer
RET_VAL:=#data_receive_ret_val
RECORD :=#data_receive     // process data <- controller
```

Network 1 suggests three consistent PCD transfer methods to the controller.

1. consistent transfer via 1 PCD
2. consistent transfer via 2 PCDs
3. consistent transfer via 2 and more PCDs

Select the transfer method selected in the hardware configuration and delete the other transfer methods or change the code by means of a comment.

⇒ Important: The I/O addresses must correspond to those in the hardware configuration and are entered as a hexadecimal code.

3.2.2 Network 2, controller on, mains contactor on

This network can be used for your own link to the application program.

3.2.3 Network 3, 4 and 5, ctrl. enable, RFG (ramp function generator) and QSP ⇒ controller

Network 3
```
SET                //replaces link of the application program
= #data_send.control_word.operation_enable_controller
```

Network 4
```
//"RFG stop/RFG enable"
SET                // replaces link of the application program
= #data_send.control_word.RFG_no_stop

//"RFG zerol/RFG enable "
= #data_send.control_word.RFG_not_zero
```

Network 5
```
SET // replaces link of the application program
= #data_send.control_word.NO_QSP_controller
```

In these networks, controller enable, QSP (quick stop) and RFG (ramp function generator) are activated or not activated. The SET commands can be replaced by the application program.

3.2.4 Network 6, DRIVECOM ⇒ controller

Network 6
```
//## Device status (Drivecom) -> controller
// => "switched on"
SET
S #data_send.control_word.DRIVECOM_switch_on
S #data_send.control_word.DRIVECOM_inhibit_voltage
S #data_send.control_word.DRIVECOM_quick_stop
// => "Ready to switch on"
UN #data_receive.statusword.DRIVECOM_ready_to_switch
UN #data_receive.status_word.fault_TRIP
R #data_send.control_word.DRIVECOM_switch_on
```

The network must not be changed. After enabling in networks 3, 4 and 5 and selecting a setpoint speed, the controller changes from the DRIVECOM status “operational” to “switched-on”. The motor will then rotate at the specified speed provided that the controller was enabled via the control terminals (terminal 28).

3.2.5 Network 7, process data words ⇒ controller

Network 7
```
// Process data word 1 (setpoint 1)
L 0                    // replaces user date
T #data_send.processdataword_1
```

In this network, the process data for the controller are defined.
⇒ Note: The variables table must be adapted if you want to work with more than 3 process data words.

3.2.6 Network 8, PCD ⇒ controller

Network 8
```
// 1.##############################################
// Consistent transfer via 1 process data word
// Process data word 0 (control word)
L LW 4                 // temp. local variable
T "PAW 264"        // 1st I/O address for process data transfer
// Process data word 1 (setpoint)
L #data_send.processdataword_1   // temp. local variable
T "PAW266"

// 2.##############################################
// Consistent transfer via 2 process data words
// l LD 4                 // temp. local variable
// t PAD 264          // 1st I/O address for process data transfer

// 3.##############################################
// Consistent transfer via 2 and more process data words
CALL  "DPWR_DAT"
LADDR  :=W#16#108                  // 1st I/O address for process data
transfer
RECORD :=#data_send              // process data <- controller
RET_VAL:=#data_send_ret_val   // error code from the SFCs
```

Here, the process data (according to network 7) for the controller are written.

As for network 1, the user may choose from three data transfer methods. The transfer methods that are not required have to be deleted or the codes have to be changed by means of a comment.

The I/O addresses also have to be adapted to the hardware configuration.

Summary – process data:

If 3 PCDs (consistent over one word) are to be used, you only have to adjust networks 1 and 8. The remaining networks (NW 2, 3, 4, 5 and 7) have to be linked to the application program. If you are working with more or less PCDs, you also have to:

a.) adjust the variables table
b.) adjust network 7

3.2.7 Network 9, DP parameter S7 ⇒ controller

```
//## Activate request =>
//#  01) C 011 -> controller
//- Setpoint -> request list
L L#50              // replaces selection of the application program
L L#10000           // 4 decimal positions
*D                  // => 50 Hz
T "DB31".telegram_001.data
//- Activate request
SET                 // replaces selection of the application program
= DB31.DBX 8.6

//# 02) C 050 <- controller
//- ? request ready without errors
O DB31.DBX 16.6  // activate request
O DB31.DBX 16.7  // request with error
SPB nOK2
//- Actual value <- request list
L "DB31".telegram_002.data
L L#10000           // 4 decimal positions
/D                  // => in min-1
//T xx              // replaces transfer target of the application program
nOK2: NOP 0
//- Activate request
SET                 // replaces selection of the application program
= DB31.DBX 16.6

//# 03) C 012 -> controller
//- Setpoint -> request list
L L#123             // replaces selection of the application program
L L#1000            // 4 decimal positions
*D                  // => 12.3sec
T "DB31".telegram_003.data
//- Activate request
SET                 // replaces selection of the application program
= DB31.DBX 24.6

//# 04) C 013 -> controller
//- Setpoint -> request list
L L#123             // replaces selection of the application program
L L#100             // 4 decimal positions
*D                  // => 1.23sec
T "DB31".telegram_004.data
//- Activate request
SET                 // replaces selection of the application program
= DB31.DBX 32.6

//## Process request
CALL FC 127
peripherieadr_1st_byte:=256     // 1st I/O address
DB_transferlist          :="DB31" // DB with request list
begin_DB_transferlist    :=0        // beginning of the request list in DB
enable_transfer          :=TRUE  // enable parameter transfer
timeout_timer            :=T35      // time monitoring
```

Network 9 comprises a request list in which four parameter requests are activated for the codes:
C011 (fdmax),
C051 (fdactual,
C012 (acceleration time) and
C013 (deceleration time).

A parameter request comprises 8 bytes or 4 words. The individual requests are written to data block 31. At the end of the request list, FC127 is called which processes the requests one after another.

Each request must be separately activated by means of the Lenze service byte. For more details, please refer to the description for DB31 (chapter 5, on page 15).

4. FC127 function

This function processes the individual parameter requests. It requires consistent parameter configuration because parameters should always be consistently transmitted.

Important ⇒ In the hardware configuration, consistency must be activated throughout the whole I/O address range for the parameter channel

CALL FC127

peripherieadr_1st_byte	:=256	decimal I/O address
DB_transferlist	:=”DB31“	DB with request list
begin_DB_transferlist	:=0	beginning of the request list in DB
enable_transfer	:=TRUE	enable parameter transfer, reset to FALSE after correct feedback
timeout_timer	:=T35	time monitoring

begin_DB_transferlist: Defines from which DB31 request number onwards requests are to be processed. Preferably, always start with 0.

enable_transfer: This bit activates parameter transfer.

timeout_timer: The timer monitors the run time of the parameter requests being processed. Unless the request is correctly processed within 5 s, bit 7 is set to one in the service byte. Then, the next request is processed.

5. Data block DB31

DB31 consists of 3 parameter requests and can be optionally extended. The request format is always identical. The first 4 words of a block must not be changed.

The 8 bytes of a request have the following meaning:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
service byte	subindex	high byte index	low byte index	data 4 / error 4	data 3 / error 3	data 2 / error 2	data 1 / error 1

Byte 1: Service byte => defines the data length of the parameter to be transferred and if a read or write request is to be initiated

Bits 7 6 5 4 3 2 1 0

- 0 1 => read request
- 1 0 => write request
- ➥ not assigned at present
- ➥ reserved
- 0 0 => 1-byte data length
- 0 1 => 2-byte data length
- 1 0 => 3-byte data length
- 1 1 => 4-byte data length

The data length of the parameter can be found in the attributes table in the System Manual (chapter “Code table”).

- ➥ **Handshake bit:** **This bit has to be set to activate a read or write request. Once the transfer has been completed (with/without fault), the block resets this bit.**

Error bit: This bit is set if an error occurs. If a successful request is transferred afterwards, the bit is reset once the request has been processed.

Byte 2: Subindex of the code number to be written or read.

Bytes 3-4: Index or code number of the controller to be read or written, calculation: 24575 – code no.

Bytes 5-8: Data sent to the controller upon a “read” request or, upon a “write” request, from the controller to the control.

Data:

- The data length and the code format can be found in the attributes table for the code table in the System Manual for the device used. The common data format is the fixed point format with 4 decimal positions.

  When using this format, the parameters have to be multiplied by 10000.

Data block DB31 transmits the following information for one parameter request each:

- Read or write request
- Data length
- Index (or parameter code number)
- Subindex for the parameter code number
- Real data

Note: In the event of an error

- The error bit (bit 7) in the service byte should be evaluated. If an error occurs in a parameter request an error code is displayed in the data bytes (bytes 5-8). The description of the error code can be found in the Technical Description for the PROFIBUS module used.

- If an error occurs the last error code that occurred is displayed in address 4 “error” (double word) of data block 31 (W4 and W6). This address range is designed as a memory. The contents is only overwritten if a new error occurs.

5.2 Example DB31

Task definition: Change the value of code C012 to 12.3 s.

Solution:

	telegram_001	STRUCT		##### C 012 -> controller
1.0	servicebyte	BYTE	B#16#32	Service byte -> Lenze
2.0	subindex	BYTE	B#16#0	Subindex (subcode)
4.0	index	INT	24563	Index = 24575 – code no.
8.0	data	DINT	L#123000	Job date (observe format: decimal positions)
=8.0	END_STRUCT			

Explanation:

In the first column you can see the memory locations for the data types used.

BYTE	:= one memory location
WORT/INT	:= two memory locations
DINT/DWORD	:= four memory locations

The service byte contains a 4-byte write request. The code to be written has no subindex (=0).
24575 – 12 = 24563

The acceleration time is to be set to 12.3 s. The new code value is displayed in the “data” area.

12.3 x 10000 = 123000 (4 decimal positions)

5.1 Interaction of FC30 function and data block DB31

The requests to be executed are entered in data block DB31. This can be done in two different ways.

1.) You can use the request list in FC30 and write data from there to DB31.
2.) All necessary data are created as initial values in DB31.

To initiate a request the handshake bit (bit 6) must be set in the service byte. This is done in FC30 (NW 9) which controls the FC127 function block.

The program lines marked in black set the handshake bit.

```
//# 03) C 012 -> controller
//- Setpoint -> request list
L L#123             // replaces selection of the application program
L L#1000            // 4 decimal positions
*D                  // => 12.3 sec
T "DB31".telegram_003.data
//- Activate request
SET                 // replaces selection of the application program
= DB31.DBX 24.6
```

The FC127 function changes the status of the handshake bit automatically after transfer provided that the correct response was received. If the parameters are cyclically read the handshake bit must be permanently set in the control function block (e.g. FC30) (SET).

The FC127 function runs through data block DB31 and compares if a request has been activated. If the information “30 hex” is displayed as last request in the service byte, FC127 re-starts running through the data block.