SIMATIC NET

IM 182 PROFIBUS Interface Module

User Description Date 08 July 97

Order No. 6ES7 182-0AA00-8BA0

SIEMENS

SIMATIC NET

IM 182

User Description

(PROFIBUS Interface Module
according to EN 50 170

Version:1.3

Date: 08 July 1997

Liability Exclusion
We have tested the contents of this document regarding agreement with the hardware and software described. Nevertheless, deviations can't be excluded and we can't guarantee complete agreement. The data in the document is checked regularly, however. Required corrections are included in subsequent versions. We gratefully accept suggestions for improvement.

Copyright
Copyright © Siemens AG 1997 All Rights Reserved
Unless permission has been expressly granted, passing on this document or copying it, or using and sharing its content is not allowed. Offenders will be held liable. All rights reserved, in the event a patent is granted, or a utility model or design is registered.

The trademarks SIMATIC, SINEC and SOFTNET, are protected by law through registration by Siemens.
All other product- and system names are (registered) trademarks of their respective owner and are to be treated as such.

Subject to technical change.

Copyright (C) Siemens AG 1997. All rights reserved.

Table of Contents

1. PRODUCT FEATURES 5

2. MECHANICAL CONCEPT 6

3. FUNCTION 7

3.1 Block Diagram 7

3.2 Function Description 7

4. HARDWARE INTERFACE 8

4.1 Pin Assignment for the Profibus Connector 8

4.2 Setting the Module Address in the PC/AT-Address Space 9
4.2.1 DPRAM Range 9
4.2.2 I/O Address Range 9
4.2.3 Port Assignment 10
4.2.4 Interrupt Channel 10

5. SOFTWARE INTERFACE 11

6. DPS2 12

6.1 Introduction 12

6.2 Initialization 14
6.2.1 Hardware 14

6.3 Adjustments for the PC Slaveboard IM 182 14
6.3.1 Locating the SPC 3 14
6.3.2 Hardware Mode 14
6.3.3 Activating the Indication Function 15
6.3.4 User Watchdog 16
6.3.5 Station Address 17
6.3.6 Ident Number 17
6.3.7 Response Time 18
6.3.8 Buffer Initialization 18
6.3.9 Entry of Setpoint Configuration 18
6.3.10 Fetching the First Buffer Pointers 20
6.3.11 Baudrate Control 20
6.3.12 Start of the SPC3 20

6.4 DPS2 Interface Functions 21
6.4.1 DPS2 Indication Function (dps2_ind()) 21
6.4.2 Read Out Reason for Indication 21
6.4.3 Acknowledging the Indication 23
6.4.4 Ending the Indication 23
6.4.5 Polling the Indication 23
6.4.6 Checking Parametrization 24
6.4.7 Checking Configuration Data 25
6.4.8 Transfer of Output Data 26

Copyright (C) Siemens AG 1997. All rights reserved.

6.4.9 Transfer of Input Data 27
6.4.10 Transferring Diagnostics Data 27
6.4.11 Checking Diagnostics Data Buffers 29
6.4.12 Changing the Slave Address 29
6.4.13 Signaling Control Commands 29
6.4.14 Leaving the Data Exchange State 30
6.4.15 DPS2_Reset (Go_Offline) 30
6.4.16 Response Monitoring Expired 31
6.4.17 Requesting Reparameterization 31
6.4.18 Reading Out the Baudrate 31
6.4.19 Determining Addressing Errors 31
6.4.20 Determining the Free Memory Space in the SPC3 32

7. SAMPLE PROGRAM 33

7.1 Overview 33

7.2 Main Program 34

7.3 Interrupt Program 38

8. GENERAL 40

8.1 Addresses 40

Copyright (C) Siemens AG 1997. All rights reserved.

1. Product Features

The IM182 makes a customer-specific PROFIBUS DP slave connection of a PC to a PROFIBUS DP network possible. The PROFIBUS interface is implemented by the ASIC SPC3 on the module.

The basis is the user description of the ASIC SPC3. This description is also relevant to the ASIC-specific data.

The maximum baudrate possible is 12 MBaud.

For the slave's input-, output-, diagnostic-, parameter- and configuring data, the internal memory of the SPC3 is available.

The operating temperature range for the components used is 0° ... +60° C.

Copyright (C) Siemens AG 1997. All rights reserved.

2. Mechanical Concept

The mechanical scheme of the module is shown in Figure 1.

The host interface between the SPC3 and an AT-compatible computer, as well as the connection of the ASIC to an RS485 interface is installed on the AT carrier module.

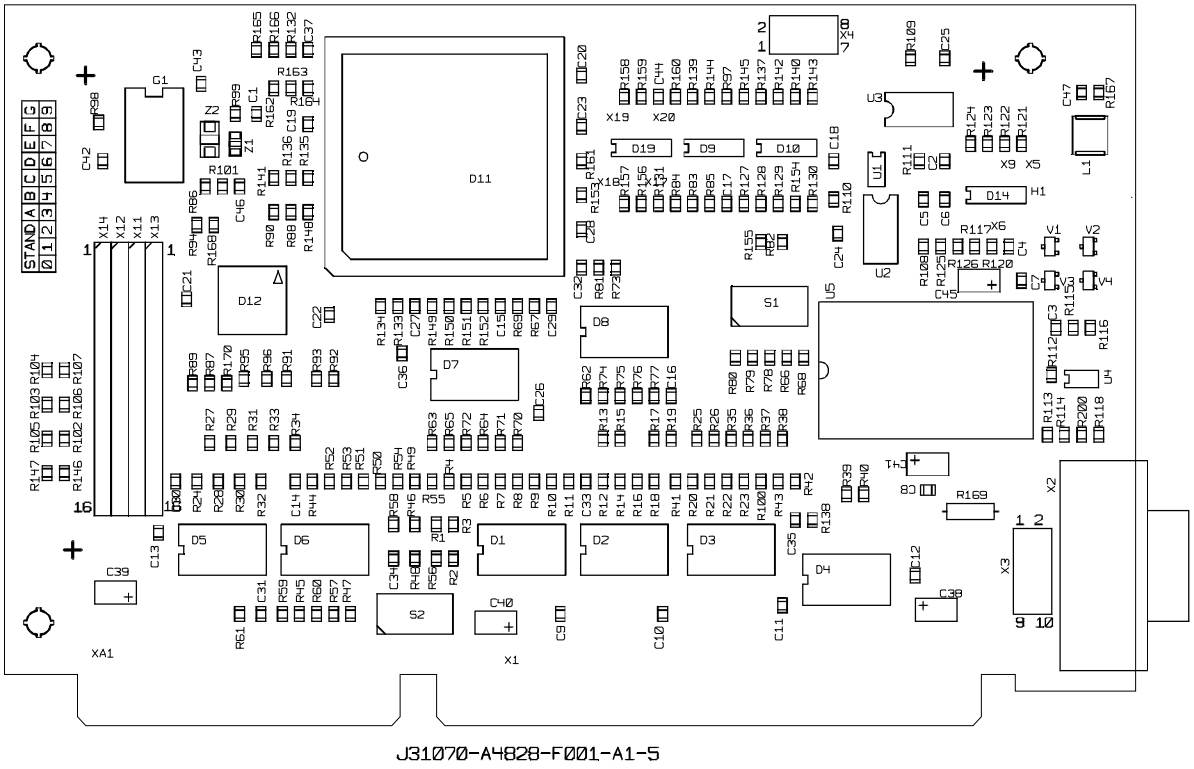


Figure 1: IM182

Copyright (C) Siemens AG 1997. All rights reserved.

3. Function

3.1 Block Diagram

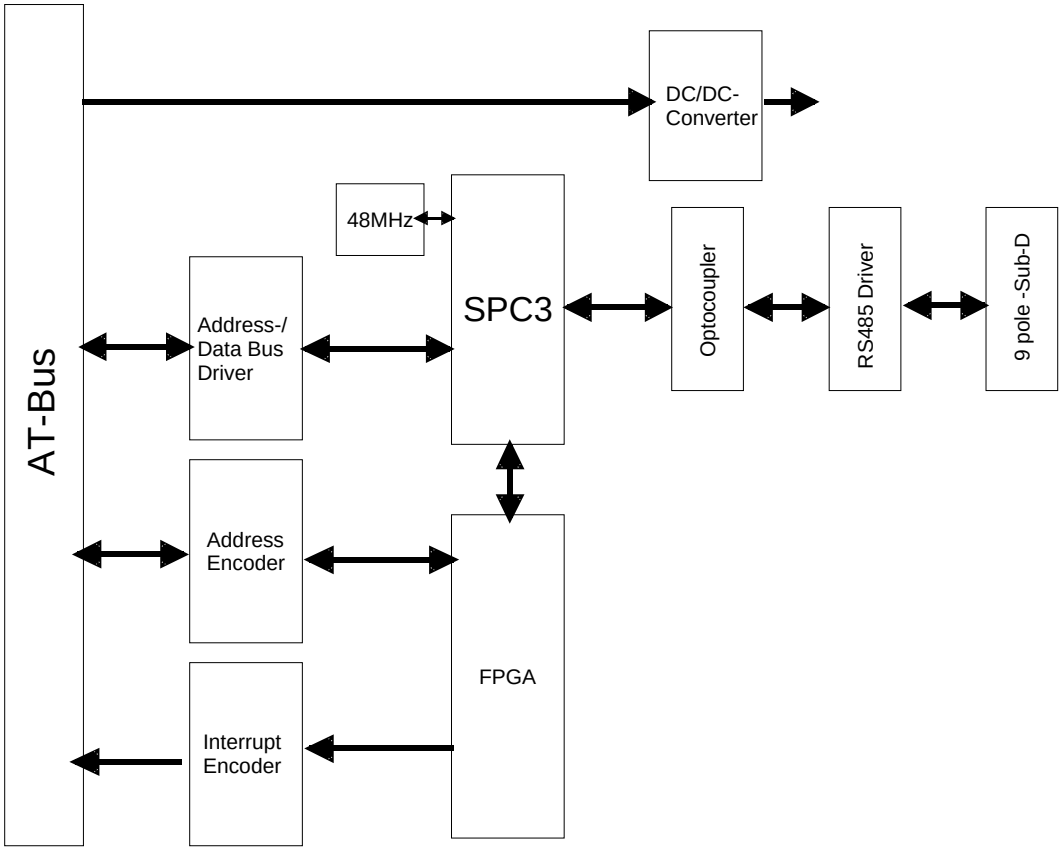


Figure 2: Block Diagram

3.2 Function Description

On the module, the following functionality is implemented; refer to Figure 2:

SPC3
- **o** PROFIBUS DP ASIC
- **o** processes PROFIBUS DP communication
- **o** 48 MHz clocked

FPGA
- **o** decoding of the control signals of AT-bus and SPC3

PROFIBUS Interface
- **o** isolated
- **o** 12 MBaud maximum transmission speed

AT-Bus Interface
- **o** buffering of data- and address line
- **o** address decoding

Copyright (C) Siemens AG 1997. All rights reserved.

4. Hardware Interface

4.1 Pin Assignment for the Profibus Connector

Data is transmitted in the operating mode RS485 (RS485 physics).
The PROFIBUS interface is designed as a 9-pole SUB-D connector (socket), with the following pin assignment:

Pin 1 - free
Pin 2 - free
Pin 3 - B - Line
Pin 4 - Request to Send (RTS)
Pin 5 - Ground 5V **(M5)**
Pin 6 - Potential 5V **(potential-free P5)**
Pin 7 - free
Pin 8 - A - Line
Pin 9 - free

The line shield is to be connected to the connector casing.

Attention:
The designations **A** and **B** for the lines on the connector correspond to the designations in the RS485 standard, and not to the pin designations of driver ICs.

When using the baudrates of 3 to 12 MBaud, new connectors have to be used. These connectors compensate line influences for all possible line combinations.

6ES7 972-0BB10-0XA0 with PG socket
6ES7 972-0BA10-0XA0 without PG socket

Transmission Timing

No.	Parameter	MIN	MAX	Unit
	Clock 48 MHz :			
1	RTS ↑ to TxD Setup Time	4T		
2	RTS ↓ to TxD Hold Token	4T		

T = Clock Cycle (48MHz)

Copyright (C) Siemens AG 1997. All rights reserved.

4.2 Setting the Module Address in the PC/AT-Address Space

The address decoder of the AT carrier module decodes a 16kByte address window from the PC/AT address space. Via this 16kByte address window, the DPRAM in the SPC3 may be accessed. HW-registers can be addressed via the port address range of the AT carrier module.

With the DIP switches S1.x , all settings of the module can be performed.

4.2.1 DPRAM Range

DPRAM-Range	S 1.1	S 1.2	S 1.3
C8000 - CBFFF	ON	OFF	ON
CC000 - CFFFF	ON	OFF	OFF
D0000 - D3FFF	OFF	ON	ON
D4000 - D7FFF	OFF	ON	OFF
D8000 - DBFFF	OFF	OFF	ON
DC000 - DFFFF	OFF	OFF	OFF

4.2.2 I/O Address Range

The following port address ranges can be set:

Port Address Range	S 1.5	S 1.6	S 1.7	S 1.8
220 - 22F	ON	OFF	ON	ON
230 - 23F	OFF	OFF	ON	ON
240 - 24F	ON	ON	OFF	ON
250 - 25F	OFF	ON	OFF	ON
260 - 26F	ON	OFF	OFF	ON
320 - 32F	ON	OFF	ON	OFF
330 - 33F	OFF	OFF	ON	OFF
340 - 34F	ON	ON	OFF	OFF
350 - 35F	OFF	ON	OFF	OFF

Gray shaded cells indicate the default setting.

Copyright (C) Siemens AG 1997. All rights reserved.

4.2.3 Port Assignment

Attention:
In the case of the IM182, only Register2 and Register3 is used with address offset 1 and 2!
(The other port addresses are relevant to the component variant IM181)

Address/Address Area	Designation	Function
Base Address + 0h	Register 1	reserved
Base Address + 1h	Register 2	Reset
Base Address + 2h	Register 3	Interrupt Source
Base Address + 3h	Register 4	reserved
Base Address + 4h	Register 5	reserved
Base Address + 5h	Register 6	reserved
Base Address + 6h	Register 7	reserved
Base Address + 7h	Register 8	reserved

4.2.3.1 Register 2

Description:

Access:	Hex Value	Explanation
Reading		
Writing	21h	perform hardware reset of the IM182

4.2.3.2 Register 3

A coming interrupt sets a corresponding bit. The setting of the interrupt channel specifies which hardware interrupt number is triggered on the PC. After that, the cell has to be read out; this also acknowledges the interrupt.

Attention: Reading out the register resets all assigned bits.

Bit	7	6	5	4	3	2	1	0
Signal	free	free	free	free	res	res	IM182	res

4.2.4 Interrupt Channel

Channel	S 2.1	S 2.2	S 2.3	S 2.4	S 2.5	S 2.6	S 2.7	S 2.8
3	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
5	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF
7	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF
10	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF
11	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF
12	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF
14	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
15	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON
none	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Gray-shaded cells indicate the default setting.

Copyright (C) Siemens AG 1997. All rights reserved.

5. Software Interface

The software package DPS2 for SPC3 makes a simple access on the C-level to the Profibus ASIC SPC3 possible. The programs are written in "C".

The sample program userspc3.c shows the use of the function calls and sequences documented in the SPC3 user description.

Module intspc3.c is not integrated as interrupt module in the PC/IM182 environment on delivery, but, to make getting started easier, as a polled call in the major loop. After setting the desired interrupt line and programming the start vector, this module can also be used as an interrupt module.

Various help programs are included in the module dps2spc3.c

The access structures and definitions are in the header file spc3dps2.h

To adapt the IM182 to your own PC environment, the following settings in the hardware/software have to be made:

Function	Hardware	Software
Base address of the SPC3 at the PC's AT-Bus	Setting with S1.1, S1.2,S1.3	Assignment of structure spc3 to address: for example (Watcom C Compiler: #define spc3 (*(SPC3 __far*) MK_FP(0xd800,0))
Reset-address of SPC3 at the PC's AT-Bus	Setting with S1.5, S1.6,S1.7.S1.8	Assignment of the register addresses: for example (Watcom C Compiler: #define SPC3_RESET 0x321
Interrupt channel of the SPC3 at the PC's AT-Bus	Setting with S2.1 to S2.8	Assignment of the interrupt vector to the program module intspc3.c

The address and interrupt allocations depend on the used compiler / linker.

Copyright (C) Siemens AG 1997. All rights reserved.

6. DPS2

6.1 Introduction

The PROFIBUS DP ASIC SPC3on the IM 182 almost completely relieves a connected microprocessor of processing the PROFIBUS DP state machine.

The interface to the user is the register or RAM interface, which is to be located in the hardware description.

The DPS2 program package for the SPC3 relieves the SPC3 user of hardware register manipulations and memory calculations. DPS2 provides a convenient „C“-interface, and particularly provides support when the buffer organization is set up.

The entire project package consists of:

Module		Function
userspc3.c	Main Program	The following functions are serviced here: start-up, input/output, and diagnostics
intspc3.c	Interrupt Module	This module handles the following functions: parameter assignment and configuration
dps2spc3.c	Help Functions	These functions calculate the buffer organization from the desired configuration.
dps2user.h	Macros and Definitions	These macros make it simple for the user to access the ASIC register structure.

As an interface to the user, DPS2 needs an interrupt for the SPC3 that the user must set up. It is also possible to block the interrupt entirely and process the corresponding functions with the polling process.

The functions which have to be carried out when the ASIC interrupt occurs are included in the intspc3.c program.

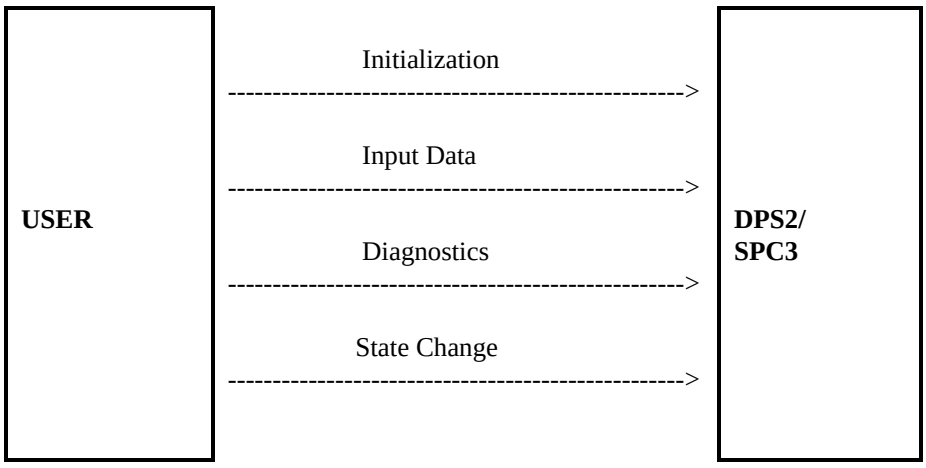
The interface between the user and the DPS2 firmware is divided into sequences and functions:

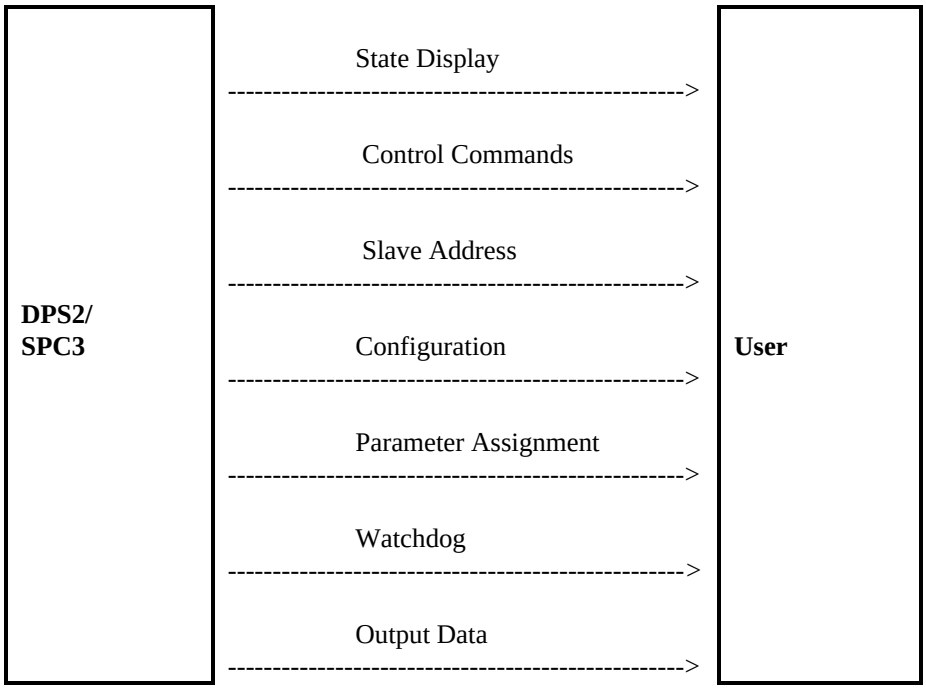
- Which the application makes available and which DPS2 calls up,

and functions

- Which DPS2 makes available and which the DPS2 application calls up.

Copyright (C) Siemens AG 1997. All rights reserved.





Copyright (C) Siemens AG 1997. All rights reserved.

6.2 Initialization

6.2.1 Hardware

During the first start-up step, the application program resets the ASIC SPC3 via the RESET pin, initializes the internal RAM and the resets connections of the connected processor.

With the declaration #define DPS2_SPC3 the DPS2 interface is activated.

To support the different memory allocation models the accesses to the SPC3 are distinguished with a seperate attribute.

For 80C32-Compiler the addressing of the user data is as follows
#define SPC3_DATA_XDATA /* user data is located to the external RAM*/
#define SPC3_DATA_IDATA /* user data is located to the internal RAM*/

With the definition #define SPC3_NO_BASE_TYPES the declaration of the basic types (UBYTE, BYTE, UWORD, WORD) can be suppressed.

6.3 Adjustments for the PC Slaveboard IM 182

The adaption of the software package to the board IM 182 is done automatically due to the detected compiler (Microsoft C/C++ or Watcom C/C++).
The following additional declarations have to be defined:
in file userspc3.c
#define PC_USE_INTERRUPT for interrupt mode (otherwise polling mode is defined)
#define SPC3_IOADDR_ 0x320 (Value for IM 182, refer to chapter 4.2.2)
#define PC_IRQ 11 (adjusted interrupt for IM 182, refer to chapter 4.2.5)

6.3.1 Locating the SPC 3

To have an easy access at the SPC3 it is possible to define a structure with the type SPC3. It has to be located at the address range defined by the hardware.

6.3.2 Hardware Mode

The macro DPS2_SET_HW_MODE (|) makes various SPC3 settings possible.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_SET_HW_MODE(x)	Hardware Settings	
Transfer		
	INT_POL_LOW	The interrupt output is low active.
	INT_POL_HIGH	The interrupt output is high active.
	EARLY_RDY	Ready is moved ahead by one pulse.
	SYNC_SUPPORTED	Sync_Mode is supported.
	FREEZE_SUPPORTED	Freeze_Mode is supported.
	DP_MODE	DP_Mode is enabled; the SPC3 sets up all DP_SAPs.
	EOI_TIMEBASE_1u	The interrupt inactive time is at least 1 usec.
	EOI_TIMEBASE_1m	The interrupt inactive time is at least 1 ms
	USER_TIMEBASE_1m	The User_Time_Clock interrupt occurs every 1 ms.
	USER_TIMEBASE_10m	The User_Time_Clock interrupt occurs every 10 ms.
Return	------	

The User_Time_Clock is a timer freely available for the application. This timer generates a 1 ms and a 10 ms timer tick. Through a relevant enable, this timer tick leads to an interrupt. (Refer to the following paragraph.)

6.3.3 Activating the Indication Function

The DPS2_SET_IND ( ) macro activates the indication functions and interrupt triggers. The transfer parameters can be represented as UWORD, as BYTE (ending _B) and as BIT (ending: _NR).

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_SET_IND(x|x..)	Activate Indication Field	
Transfer	MAC_RESET	After processing the current job, the SPC3 has entered the *Offline State* by setting the 'Go_Offline' bit.
here	GO_LEAVE_DATA_EX	The DP_SM has entered the 'DATA_EX' state or has exited it.
UWORD	BAUDRATE_DETECT	The SPC3 has exited the 'Baud_Search State' and has found a baud rate.
Representa-tion	WD_DP_MODE_TIMEOUT	The watchdog timer has expired in the 'DP_Control' WD state.
	USER_TIMER_CLOCK	The time base of the User_Timer_Clock has expired (1/10ms) timer tick.
	Reserved	for additional functions
	Reserved	for additional functions
	Reserved	for additional functions
	NEW_GC_COMMAND	The SPC3 has received a 'Global_Control Message' with a changed 'GC_Command-Byte' and has stored this byte in the 'R_GC_Command' RAM cell.
	NEW_SSA_DATA	The SPC3 has received a 'Set_Slave_Address Message' and has made the data available in the SSA buffer.
	NEW_CFG_DATA	The SPC3 has received a 'Check_Cfg Message' and has made the data available in the Cfg buffer.
	NEW_PRM_DATA	The SPC3 has received a 'Set_Param Message' and has made the data available in the Prm buffer.
	DIAG_BUFFER_CHANGED	On request by 'New_Diag_Cmd', the SPC3 has exchanged the diagnostics buffers and has made the old buffer available again to the user.
	DX_OUT	The SPC3 has received a 'Write_Read_Data Message' and has made the new output data available in the N buffer. For 'Power_On' or for 'Leave_Master', the SPC3 clears the N buffer contents and also generates this interrupt.
	Reserved	For additional functions
	Reserved	For additional functions
Return	------	

Example:

DPS2_SET_IND(GO_LEAVE_DATA_EX | WD_DP_MODE_TIMEOUT);
*/ The user is informed when the DATA_Exchange state is entered or exited, or when the watchdog timer has run out. */

An interrupt activation with byte variables could look like this:

DPS2_SET_IND(NEW_CFG_DATA_B | NEW_PRM_DATA_B | USER_TIMER_CLOCK_B);

6.3.4 User Watchdog

The user watchdog ensures that if the connected microprocessor fails, the SPC3 leaves the data cycle after a defined number (DPS2_SET_USER_WD_VALUE) of data messages. As long as the microprocessor doesn't „crash", it has to retrigger this watchdog (DPS2_RESET_USER_WD).

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_SET_USER_WD_VALUE (x)		Set User Watchdog Time
Transfer	UWORD	Number of data messages
Return	------	

DPS2_RESET_USER_WD()		Complete restart / retriggering of user watchdog
Transfer	------	
Return	------	

In the worst case scenario, the data telegrams can be sent in the time interval of the Min_Slave interval. By means of this time specification and the run length of its own program component, the application can specify the number of data messages.

Sample calculation: ($T_{application\ runtime}$ / min_slave interval) x 2 = number of data telegrams

Refer to DIN E 19245 Part 3 (maximum master polling time of telegrams to the slave).
2 = safety factor

6.3.5 Station Address

During startup, the application program reads in the station address (DIL switch, EEPROM, etc.), and transfers the station address to the ASIC. The user must also specify whether this station address can be changed via the PROFIBUS DP; that is, a memory medium (for example, serial EEPROM) is available.

DPS2_SET_STATION_ADRESS (x)		Set Station Address
Transfer	UBYTE	Address
Return	------	

DPS2_SET_ADD_CHG_DISABLE()		Station Address Change Disabled
Transfer	------	
Return	------	

DPS2_SET_ADD_CHG_ENABLE()		Station Address Change Permitted Attention: The user must set up buffers for this utility!
Transfer	------	
Return	------	

6.3.6 Ident Number

During startup, the application program reads in the ident number (EPROM, host system) and transfers it to the ASIC.

DPS2_SET_IDENT_NUMBER_HIGH(x)		Ident Number
Transfer	UBYTE	High byte of PNO ident number
Return	------	

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_SET_IDENT_NUMBER_LOW(x)		Ident Number
Transfer	UBYTE	Low byte of PNO ident number
Return	------	

6.3.7 Response Time

If special circumstances require it, the user can set the response time for the SPC3 during set-up. In operation with PROFIBUS DP, the parameter message of the PROFIBUS DP master specifies the response time.

DPS2_SET_MINTSDR(x)		MinTsdr
Transfer	UBYTE	Response time in bit timing (11-255)
Return	------	

6.3.8 Buffer Initialization

The user must enter the lengths of the exchange buffers for the different messages in the dps2_buf structure of the DPS2_BUFINIT type. These lengths determine the data buffers set up in the ASIC, and therefore are dependent in total sum on the ASIC memory. DPS2_INIT checks the maximum lengths of the buffers entered, and returns the test result. Please specify the overall calculation. Is the in/out buffer mutually specified?

```
typedef struct {
  UBYTE din_dout_buf_len;     /*overall length of the input/output buffer, 0-488*/
  UBYTE diag_buf_len;         /*length of the diagnostics buffer, 6-244*/
  UBYTE prm_buf_len;          /*length of the parameter buffer, 7-244*/
  UBYTE cfg_buf_len;          /*length of the config data buffer, 1-244*/
  UBYTE ssa_buf_len;          /*length of the Set-Slave-Add buffer, 0 and 4-244*/
} DPS2_BUFINIT;
```

Specifying the length 0 for the Set-Slave-Address buffer disables this utility.

For this type of buffer initialization, an additional macro is needed for adapting the lengths of the Din/Dout buffers, since these are the only ones that are allowed to be changed during operation (but not beyond the preset size).

DPS2_INIT (x)	Buffer Initialization	
Transfer	Pointer to values with the DPS2_BUFINIT structure	Desired/required buffer lengths
Return	DPS2_INITF_DIN_DOUT_LEN	Error with Din/Dout length
	DPS2_INITF_DIAG_LEN	Error with diagnostics length
	DPS2_INITF_PRM_LEN	Error with parameter assignment data length
	DPS2_INITF_SSA_LEN	Error with address data length
	DPS2_INITF_LESS_LEN	Overall, too much memory used
	DPS2_INITF_OK	Buffer length OK

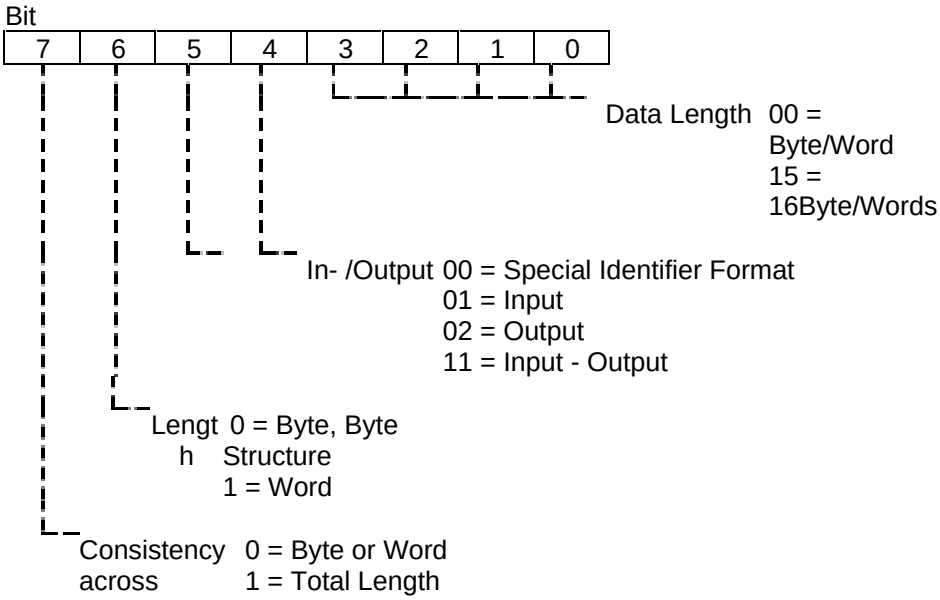
6.3.9 Entry of Setpoint Configuration

With the macro, the function first fetches a pointer to a data block for the configuration.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_GET_READ_CFG_BUF_PTR()	Fetch Pointer to Configuration Buffer	
Transfer	----	
Return	UBYTE *	Pointer to RAM area in the SPC3

In this data block, the user enters his configuration (identifier bytes). The individual identifier bytes are to be generated according to the following specification (refer also to DIN E 19245 Part 3):



For example, the identifiers correspond to 17 hex = 8 bytes input without consistency

27 hex = 8 bytes output without consistency

The special identifier formats are to be found in DIN E 19245 T3.

With the DPS2_SET_READ_CFG_LEN (CFG_LEN) macro, the user sets the length of the configuration data entered.

DPS2_SET_READ_CFG_LEN (x)		Set Length of Configuration Data
Transfer	UBYTE	Length of entries in the configuration buffer
Return	----	

Then the user uses the dps2_calculate_inp_outp_len() function made available in the dps2spc3.c file to determine the length of the input and output data from the identifier bytes. This function returns a pointer to a structure of the DPS2_IO_DATA_LEN type. A zero pointer indicates a faulty buffer configuration (for example, real_cfg_data_len = 0).

dps2_calculate_inp_outp_len(x,y)		Calculation of Inputs/Outputs
Transfer	UBYTE *	Pointer to configuration buffer
	UWORD	Length of configuration data
Return	DPS2_IO_DATA_LEN *	Pointer to structure with the calculated input- output lengths

Copyright (C) Siemens AG 1997. All rights reserved.

```
typedef struct {
UBYTE inp_data_len;
UBYTE outp_data_len;
} DPS2_IO_DATA_LEN;
```

With the DPS2_SET_IO_DATA_LEN(ptr) macro, the user initiates the DPS2 variables inp_data_len and outp_data_len.

DPS2_SET_IO_DATA_LEN(x)		Set Input-/Output Data Lengths
Transfer	DPS2_IO_DATA_LEN *	Pointer to structure with the calculated input-/output lengths
Return	UBYTE	TRUE: sufficient memory available FALSE: memory insufficient

6.3.10 Fetching the First Buffer Pointers

Before the first entry of its input data, the application has to fetch a buffer for the input data with the DPS2_GET_DIN_BUF_PTR() macro. With the DPS2_INPUT_UPDATE() macro, the user can transfer the input data to DPS2. The length of the inputs is not transferred with every input; the length must agree with the length transferred by DPS2_SET_IO_DATA_LEN().

Macro DPS2_GET_DIN_BUF_PTR()		Fetch First Input Data Buffer
Transfer	-----	
Return	UBYTE *	Pointer to input buffer

Before the first entry of external diagnostics, the user must get a pointer to the available diagnostics buffer with the DPS2_GET_DIAG_BUF_PTR() macro. The user can then enter his diagnostics messages or status messages (starting with Byte 6) in this buffer.

DPS2_GET_DIAG_BUF_PTR()		Fetch first diagnostics buffer.
Transfer	-----	
Return	UBYTE *	Pointer to diagnostics buffer; NIL if no diagnostics buffer available anymore

6.3.11 Baudrate Control

With the DPS2_SET_BAUD_CNTRL () macro, the root value of baudrate monitoring can be set. After the set time (Value x Value x 10ms), the SPC3 autonomously starts the baudrate search, if no valid message was received during this time. If the master system uses the watchdog, the value the master specified for baud rate monitoring is used for watchdog monitoring. If the slave is operated without a watchdog, ASIC SPC3 interprets the entry of the root value for the baud rate monitoring. This makes a time value in the range of 10 ms - 650 s possible (entry 2-255).

DPS2_SET_BAUD_CNTRL (x)		Baudrate Monitoring
Transfer	UBYTE	Root value of baudrate monitoring
Return	------	

6.3.12 Start of the SPC3

With DPS2_START, the SPC3 switches itself on-line.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_START ()		Start SPC3
Transfer	------	
Return	------	

6.4 DPS2 Interface Functions

6.4.1 DPS2 Indication Function (dps2_ind())

The user has to set up and make the dps2_ind() interrupt function ready. DPS2 will carry out this function as soon as a corresponding event has occurred which was enabled in the interrupt bit field with the DPS2_SET_IND() macro. (See above.)

dps2_ind		Interrupt Function
Transfer	-------	
Return	-------	

In a 16-bit field, the DPS2 indicates the reason for the indication to the user with bits, on which literals have been entered.

6.4.2 Read Out Reason for Indication

With the DPS2_GET_INDICATION macro, the user receives the event which has caused the indication, the interrupt trigger.

DPS2_GET_INDICATION()		Read Out Reason for Indication
Transfer	--------	
Return	UWORD	Refer to the field described under DPS2_SET_IND

In order to increase the performance, primarily the 803x and 805x (byte-oriented), you can also query each indication with its own macro (DPS2_GET_IND_...) instead. A runtime-optimized interface can be created with these macros.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_GET_IND_GO_LEAVE_DATA_EX()	The DP_SM has entered the 'DATA_EX' state or has exited it.
DPS2_GET_IND_MAC_RESET()	After processing the current request, the SPC3 has entered the *offline state* (by setting the 'Go_Offline' bit).
DPS2_GET_IND_BAUDRATE_DETECT()	The SPC3 has left the 'Baud_Search state' and has found a baud rate.
DPS2_GET_IND_WD_DP_MODE_TIMEOUT	In the 'DP_Control' WD state , the watchdog timer has expired.
DPS2_GET_IND_USER_TIMER_CLOCK	The time base of the User_Timer_Clock has expired (1/10ms).
DPS2_GET_IND_NEW_GC_COMMAND()	The SPC3 has received a 'Global_Control Message' with a changed 'GC_Command Byte' and has stored this byte in the 'R_GC_Command' RAM cell.
DPS2_GET_IND_NEW_SSA_DATA()	The SPC3 has received 'Set_Slave_Address Message' and has made the data available in the SSA buffer.
DPS2_GET_IND_NEW_CFG_DATA()	The SPC3 has received Check_Cfg Message' and has made the data available in the Cfg buffer.
DPS2_GET_IND_NEW_PRM_DATA()	The SPC3 has received 'Set_Param Message' and has made the data available in the Prm buffer.
DPS2_GET_IND_DIAG_BUFFER_CHANGED()	Requested by 'New_Diag_Cmd' , the SPC3 has exchanged the diagnostics buffer and has made the old buffer available again to the user.
DPS2_GET_IND_ DX_OUT()	The SPC3 has received a 'Write_Read_Data Message' and has made the new output data available in the N buffer. For 'Power_On' and for 'Leave_Master', the SPC3 clears the N buffer contents and also generates this interrupt.

Transfer	--------	
Return	UBYTE	0/FALSE: no interrupt 1/TRUE: This indication/interrupt has occurred.

Copyright (C) Siemens AG 1997. All rights reserved.

6.4.3 Acknowledging the Indication

The DPS2_IND_CONFIRM() macro acknowledges the indication received through dps2_ind().

DPS2_IND_CONFIRM(x)		Acknowledge the Indication
Transfer	UWORD	Refer to the field described under DPS2_SET_IND.
Return	--------	

Performance can also be increased by here defining a macro each for each indication (see „Read Out the Reason for indication“).

DPS2_CON_IND_GO_LEAVE_DATA_EX()		See above
DPS2_CON_IND_MAC_RESET()		
DPS2_CON_IND_BAUDRATE_DETECT()		
DPS2_CON_IND_WD_DP_MODE_TIMEOUT		
DPS2_CON_IND_USER_TIMER_CLOCK		
DPS2_CON_IND_NEW_GC_COMMAND()		
DPS2_CON_IND_NEW_SSA_DATA()		
DPS2_CON_IND_NEW_CFG_DATA()		
DPS2_CON_IND_NEW_PRM_DATA()		
DPS2_CON_IND_DIAG_BUFFER_CHANGED()		
DPS2_CON_IND_ DX_OUT()		
Transfer	--------	
Return	--------	

6.4.4 Ending the Indication

The DPS2_SET_EOI() macro ends the indication sequence / interrupt function.

DPS2_SET_EOI()		Close Interrupt
Transfer	------	
Return	------	

6.4.5 Polling the Indication

The user can also poll indications instead of having them signaled with dps2_ind(). The DPS2_POLL_IND_xx macro is available for a single read-out, or DPS2_POLL_INDICATION() for global read-out. Polled indications can likewise be acknowledged with the DPS2_IND_CONFIRM macro.

DPS2_POLL_INDICATION()		Reason for Indication
Transfer	--------	
Return	UWORD	Refer to the field described under DPS2_SET_IND.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_POLL_IND_GO_LEAVE_DATA_EX()	The DP_SM has entered the 'DATA_EX' state or has exited it.
DPS2_POLL_IND_MAC_RESET()	After processing the current request, the SPC3 has entered the o*ffline state* (by setting the 'Go_Offline' bit
DPS2_POLL_IND_BAUDRATE_DETECT()	The SPC3 has left the 'Baud_Search State' and found a baud rate.
DPS2_POLL_IND_WD_DP_MODE_TIMEOUT()	In the WD state 'DP_Control', the watchdog timer has expired.
DPS2_POLL_IND_USER_TIMER_CLOCK()	The time base of the User_Timer_Clock has expired (1/10ms).
DPS2_POLL_IND_NEW_GC_COMMAND()	The SPC3 has received a 'Global_Control Message' with a changed 'GC_Command-Byte' and has filed this byte in the 'R_GC_Command' RAM cell .
DPS2_POLL_IND_NEW_SSA_DATA()	The SPC3 has received a 'Set_Slave_Address Message' and has made the data available in the SSA buffer.
DPS2_POLL_IND_NEW_CFG_DATA()	The SPC3 has received a 'Check_Cfg Message' and has made the data available in the Cfg buffer.
DPS2_POLL_IND_NEW_PRM_DATA()	The SPC3 has received a 'Set_Param Message' and has made the data available in the Prm buffer.
DPS2_POLL_IND_DIAG_BUFFER_CHANGED()	Requested by 'New_Diag_Cmd', the SPC3 has exchanged the diagnostics buffers and made the old buffer available again to the user.
DPS2_POLL_IND_ DX_OUT()	The SPC3 has received a 'Write_Read_Data Message' and has made the new output data available in the N buffer. For 'Power_On' and for 'Leave_Master', the SPC3 clears the N buffer and also generates this interrupt.

Transfer	--------	
Return	UBYTE	0/FALSE: No interrupt 1/TRUE: This indication/interrupt has occurred.

6.4.6 Checking Parametrization

The user has to program the function for checking the received parameter assignment data. DPS2 calls up the dps2_ind function in which NEW_PRM_DATA can determine whether the checking function has to be carried out. Macro call-ups from DPS2 can fetch the required pointer to the corresponding buffer and the length of this buffer.

The DPS2_GET_PRM_LEN() macro determines the length of the received data.

DPS2_GET_PRM_LEN ()		Fetch parameter buffer length.
Transfer	--------	
Return	UBYTE	Length of the parameter data buffer

DPS2_GET_PRM_BUF_PTR() supplies a pointer to the current parameter buffer.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_GET_PRM_ BUF_PTR()		Fetch pointer to parameter buffer.
Transfer	--------	
Return	UBYTE *	Address of the parameter buffer

Within this verification function, the user has the task of checking the received User_Prm_Data for validity. The user acknowledges the checked parameters as positive by calling the DPS2_SET_PRM_DATA_OK macro, and as negative by calling DPS2_SET_PRM_DATA_NOT_OK(). By acknowledging with these macros, the interrupt request is canceled; that is, this interrupt may **no** longer be acknowledged with DPS2_IND_CONFIRM(). The return value of the macros has to be evaluated as described below.

DPS2_SET_PRM_DATA_OK()		The received parameter assignment is OK.
DPS2_SET_PRM_DATA_NOT_OK()		This macro notifies DPS2 the parameter assignment isn't OK. The transferred parameters can't be used in the device.
Transfer	--------	
Return	DPS2_PRM_FINISHED	No further parameter assignment message is present => end of sequence.
	DPS2_PRM_CONFLICT	Another parameter assignment message is present! => repeat check of requested parameter assignment.
	DPS2_PRM_NOT_ALLOWED	Access in present bus mode is not permitted. For example, it is possible the watchdog has run out during verification. Verifying the parameter setting data (and possibly series-connected functions in the application) are to be cancelled.

Caution:

When configuration settings and parameter settings are received, first there **must** be verification of the **parameter setting data** and their confirmation. Then the configuration settings must be verified. The sequence is absolutely mandatory.

6.4.7 Checking Configuration Data

The user has to program the function for verifying received configuration data. DPS2 calls up the dps2_ind function in which NEW_CFG_DATA can determine whether the verification function has to be carried out. Macro calls from DPS2 supply the needed pointer as well as the buffer length.

The DPS2_GET_CFG_LEN() macro determines the length of the received data.

DPS2_GET_CFG_LEN ()		Fetch configuration buffer length.
Transfer	--------	
Return	UBYTE	Length of the received configuration byte

DPS2_GET_CFG_BUF_PTR() supplies a pointer to the current configuration buffer.

DPS2_GET_CFG_ BUF_PTR()		Fetch pointer to configuration buffer.
Transfer	--------	
Return	UBYTE *	Configuration buffer address

Within the verification function, the user has the task of comparing the received Cfg_Data with the Real_Cfg_Data; that is, its possible configuration. The user acknowledges the verified configuration data

Copyright (C) Siemens AG 1997. All rights reserved.

as positive by calling up the macro DPS2_SET_CFG_DATA_OK() or DPS2_SET_CFG_DATA_UPDATE(). The usre acknowledges the verified configuration data as negative by calling up DPS2_SET_CFG_DATA_NOT_OK() negative. By acknowledging with these macros, the interrupt request is removed; that is, this interrupt may **no** longer be acknowledged through DPS2_IND_CONFIRM(). The return value of the macros has to be evaluated as described below.

DPS2_SET_CFG_DATA_OK()	The transferred configuration is OK.
DPS2_SET_CFG_DATA_UPDATE()	If the user desires the verified configuration be exchanged with the one already in DPS2, this can be done with the *DPS2_SET_CFG_DATA_UPDATE()* macro.
DPS2_SET_CFG_DATA_NOT_OK()	This macro notifies the DPS2 that the configuration is not OK.

Transfer	--------	
Return	DPS2_CFG_FINISHED	No further configuration message is present => end of sequence.
	DPS2_CFG_CONFLICT	An additional configuration message is present! => Repeat verification of the requested configuration.
	DPS2_CFG_NOT_ALLOWED	Access is not permitted in the present bus mode. For example, it is possible the watchdog has run out during verification. The verification of the configuration data (and possibly subsequent functions in the application) are to be cancelled.

6.4.8 Transfer of Output Data

DX_OUT in dps2_ind() displays received output data. The macro DPS2_OUTPUT_UPDATE() changes the output buffers.

The DPS2_OUTPUT_UPDATE_STATE() buffer supplies the buffer pointer, and also the state of the Dout buffer.

The lengths of the outputs are not transferred with every update. The length agrees with the length transferred with DPS2_SET_IO_DATA_LEN(). If this were not the case, DPS2 would return to the WAIT-PRM state.

DPS2_OUTPUT_UPDATE_STATE ()		Fetch buffer pointer and state of the output buffer.
Transfer	UBYTE *	Pointer to variable into which the state of the output buffer is to be written
Return	UBYTE *	Pointer to output data buffer

The following states (bits) are encoded into the status (pointer to this variable was transferred):

NEW_DOUT_BUF	Received new output data
DOUT_BUF_CLEARED	Output data was deleted.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_OUTPUT_UPDATE ()		Fetch buffer pointer to output buffer.
Transfer	------	
Return	UBYTE *	Pointer to output buffer or NIL, if no buffer

6.4.9 Transfer of Input Data

As described, the application has to fetch a buffer for the input data with the DPS2_GET_DIN_BUF_PTR() macro before the first entry of its input data.

With the DPS2_INPUT_UPDATE() macro, the user can repeatedly transfer the current input data from the user to DPS2. The length of the inputs is not transferred with every update.. The length must agree with the length transferred by DPS2_SET_IO_DATA_LEN().

DPS2_INPUT_UPDATE ()		Fetch buffer pointer to input buffer.
Transfer	------	
Return	UBYTE *	Pointer to input data buffer

The input-/output data length can be reconfigured with the functions and macros described in the "Initialization" section (dps2_calculate_inp_outp_len(), DPS2_SET_IO_DATA_LEN(), ...).

6.4.10 Transferring Diagnostics Data

With this utility, the user can transfer diagnostics data to DPS2. Prior to the first entry of external diagnostics data, the user has to get a pointer to the free diagnostics buffer with the DPS2_GET_DIAG_BUF_PTR() macro. The user can then write his diagnostics messages or status messages (starting with Byte 6) into this buffer.

DPS2_GET_DIAG_BUF_PTR()		Fetch pointer to diagnostics data buffer.
Transfer	------	
Return	UBYTE *	Pointer to diagnostics buffer NIL if no diagnostics data buffer in the 'U' state

The user specifies the length of the diagnostics data by calling up the DPS2_SET_DIAG_LEN() macro. The length is only to be set after a buffer was successfully received with DPS2_GET_DIAG_BUF_PTR().

The length **always** has to be transferred for the entire buffer, including the bytes specified by the standard (+6). This means that, if no user diagnostics is supposed to be transferred, the **length 6** is to be transferred.

DPS2_SET_DIAG_LEN()		Set length of diagnostics data.
Transfer	UBYTE	Length of diagnostics data
Return	UBYTE	Diagnostics length actually set 0xff, if no buffer is assigned to the user

The transferred pointer of DPS2 points to Byte 0 of the transferred diagnostics buffer. The user may enter his diagnostics in this buffer starting with **Byte 6**. DPS2 enters the fixed diagnostics bytes (bytes 0 to 5).

Copyright (C) Siemens AG 1997. All rights reserved.

Structure of the data block to be transferred for expanded diagnostics:

Byte	Diagnostics Data	Comment
0	Station Status_1	Byte 0 to 5 permanent diagnostics header
1	Station Status_2	
2	Station Status_3	
3	Diag.Master_Add	
4	Ident_Number_High	
5	Ident_Number_Low	
6 to 241 max.	Ext_Diag_Data	Start of user diagnostics in the DP Standard format

With the DPS2_SET_DIAG_STATE() macro, the user transfers the new diagnostics state to DPS2. The new diagnostics state has to be transferred before the diagnostics data is updated.

DPS2_SET_DIAG_STATE()		Setting the Diagnostics Bits	
Transfer	Bit	Designation	Meaning
	0	EXT_DIAG	This bit indicates whether expanded diagnostics data or status messages are present. The DPS2_SET_DIAG_LEN() macro transfers the length of the diagnostics data. The diagnostics data of the application starts with the (first) header byte as of Byte 6, and has to be present as described in the PROFIBUS DP Standard; that is, a block structure with a header byte per block where the block type (device-, identification- or channel-related diagnostics) and the length of the following are defined. If the Length 6 is also transferred, DPS2 will set this bit permanently to zero.
	1	STAT_DIAG	If this bit is 1, the diagnostics bit Diag.Stat_Diag will be set; otherwise, the bit will be reset.
	2	EXT_DIAG_OVF	If this bit is 1, the bit Diag.Ext_Diag_Overflow is set; otherwise, Diag.Ext_Diag_Overflow is reset.
Return	------		

With the DPS2_DIAG_UPDATE() macro, the user transfers the new, external diagnostics data to DPS2. As a return value, the user receives a pointer to the new diagnostics data buffer.

DPS2_DIAG_UPDATE()		Transfer diagnostics data and fetch new pointer.
Transfer	------	
Return	UBYTE *	Pointer to the diagnostics buffer; NIL if no diagnostics data buffer present

If no diagnostics data is to be transferred with the DPS2_DIAG_UPDATE() macro, or if the diagnostics data transferred previously is to be deleted, the diagnostics length has to be set to 6 with the DPS2_SET_DIAG_LEN() macro. The SPC3 responds to a diagnostics request from the PROFIBUS DP master with the 6 bytes of station diagnostics data.

Copyright (C) Siemens AG 1997. All rights reserved.

6.4.11 Checking Diagnostics Data Buffers

The other exchange buffer is not automatically available after the diagnostics data has been transferred. The user has two possibilities to find out when the diagnostics buffer was transmitted:

- DPS2 signals via the dps2_ind() indication function and indicates the event with DIAG_BUFFER_CHANGED. This indication function has to be enabled during initialization for this purpose.

With the DPS2_GET_DIAG_FLAG() macro, the user polls the state of the diagnostics buffer. The macro indicates whether the buffer has already been transmitted. If, however, „static diagnostics“ has been set, the „buffer not
transmitted“ state is always returned.

DPS2_GET_DIAG_FLAG()		Fetch state of diagnostics buffer.
Transfer	------	
Return	UBYTE	TRUE: Diagnostics buffer has not yet been transmitted (or static diagnostics). FALSE: Diagnostics buffer has already been transmitted.

6.4.12 Changing the Slave Address

NEW_SSA_DATA indicates a request to change in the slave address. With the DPS2_GET_SSA_BUF_PTR() macro, a pointer to the buffer with the new slave address can be determined, and with DPS2_GET_SSA_LEN() macro, the length of the received SSA buffer can be determined.

DPS2_GET_SSA_LEN()		Length of the Set_Slave_Address Buffer
Transfer	------	
Return	UBYTE	Length of the SSA buffer

DPS2_GET_SSA_BUF_PTR()		Fetch Pointer of Set_Slave_Address Buffer.
Transfer	------	
Return	UBYTE *	SSA buffer address

The user has to acknowledge the transfer of the data by calling the DPS2_SET_SSA_BUF_FREE() macro.

DPS2_SET_SSA_BUF_FREE()		Acknowledging the Set_Slave_Address utility
Transfer	------	
Return	------	

6.4.13 Signaling Control Commands

This message signals the arrival of a Global_Control message. The message is only made if group association and a change of the control command was recognized as compared to the previous command. The DPS2_GET_GC_COMMAND() macro supplies the Control_Command byte. This makes it possible for the user to additionally react to these commands. The DPS2 internally processes these commands regarding buffer management. That is, in the case of „Clear“, the output data is deleted.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_ GET_GC_COMMAND ()		Fetch Global Control Command	
Transfer	----		
Return	Bit	Designation	Meaning
	0	Reserved	
	1	Clear_Data	This command deletes the output data and makes the data available to the user. A switch to 'U' is made.
	2	Unfreeze	With „Unfreeze", the freeze of input data is canceled.
	3	Freeze	The input data is „frozen." The application does not fetch new input data until the master sends the next „freeze" command.
	4	Unsync	The „Unsync" command cancels the „Sync" command.
	5	Sync	The output data last received is made available to the application. The following transferred output data is not passed on to the application until the next 'Sync' command is given.
	6,7	Reserved	The „Reserved" designation indicates that these bits are reserved for future function expansions.

6.4.14 Leaving the Data Exchange State

The GO_LEAVE_DATA_EX message indicates that DPS2 has carried out a state change of the internal state machine.

With the DPS2_GET_DP_STATE() macro, the application is informed whether the DPS2 has entered the data exchange state or left it. The cause for this can be a faulty parameter assignment message in the data transfer phase, for example.

DPS2_GET_DP_STATE():		Fetching the status of the PROFIBUS DP state machine
Transfer	------	
Return	DPS2_DP_STATE_WAIT_PRM	Wait for parameter assignment
	DPS2_DP_STATE_WAIT_CFG	Wait for configuration
	DPS2_DP_STATE_DATA_EX	Data exchange
	DPS2_DP_STATE_ERROR	Error

6.4.15 DPS2_Reset (Go_Offline)

With this macro, the SPC3 enters the offline state. The offline state can only be exited with the DPS2_INIT function. This provides the possibility to transfer and start new configuration data.

DPS2_RESET()		Go to the offline state.
Transfer	-------	
Return	-------	

The DPS2_GET_OFF_PASS() macro can help to determine whether the transition to offline was made.

Copyright (C) Siemens AG 1997. All rights reserved.

DPS2_GET_OFF_PASS()		Check the offline state.
Transfer	--------	
Return	UBYTE/Bit	1 = Passive idle 0 = Offline

6.4.16 Response Monitoring Expired

WD_DP_MODE_TIMEOUT indicates the sequence of response monitoring. The SPC3_GET_WD_STATE() macro queries the status of the watchdog state machine.

SPC3_GET_WD_STATE()	State of the watchdog state machine	
Transfer	------	
Return	SPC3_WD_STATE_BAUD_SEARCH	Baudr ate search
	SPC3_WD_STATE_BAUD_CONTROL	Checking the baudrate
	SPC3_WD_STATE_DP_MODE	DP_Mode; that is, bus watchdog activated

6.4.17 Requesting Reparameterization

The DPS2_USER_LEAVE_MASTER() macro causes the DPS2/SPC3 to change into the „Wait_Prm“ state.

DPS2_USER_LEAVE_MASTER()		Enter the State Wait_Prm
Transfer	------	
Return	------	

6.4.18 Reading Out the Baudrate

The DPS2_GET_BAUD() macro supplies the recognized baud rate in coded form.

DPS2_GET_BAUD()		Read baud rate.
Transfer	--------	
Return	BD_12M	12 MBaud
	BD_6M	6 MBaud
	BD_3M	3 MBaud
	BD_1_5M	1.5 MBaud
	BD_500k	500 KBaud
	BD_187_5k	187.5 KBaud
	BD_93_75k	93.75 KBaud
	BD_19_2k	19.2 KBaud
	BD_9_6k	9.6 KBaud

6.4.19 Determining Addressing Errors

The SPC3 indicates MAC_RESET and ACCESS_VIOLATION when an addressing error occurs during an access above 1.5 KB of the internal RAM. The macros SPC3_GET_OFF_PASS() and

Copyright (C) Siemens AG 1997. All rights reserved.

SPC3_GET_ACCESS_VIOLATION() are provided to distinguish between the transition between "offline" and "passive" when an addressing error occurs.

SPC3_GET_ACCESS_VIOLATION()		Addressing error has occurred
Transfer	------	
Return	UBYTES	≠ 0: Addressing error occurred = 0: No addressing error

Caution:
In C32 mode, an erroneous access of the processor does not trigger an interrupt.

6.4.20 Determining the Free Memory Space in the SPC3

During initialization, the SPC3_INI() macro sets up buffer space in the internal RAM of the SPC3. You can use this macro to provide yourself with a pointer to the beginning of the free memory space in the SPC3, and the number of bytes still available. This functions returns a ZERO pointer when the SPC3 has not been initialized.

SPC3_GET_FREE_MEM()		Determine free memory space
Transfer	UBYTE *	Pointer to the location containing the memory space available
Return	UBYTE *	Pointer to the free memory space in the SPC3 0 when SPC3 was not initialized correctly

Copyright (C) Siemens AG 1997. All rights reserved.

7. Sample Program

7.1 Overview

The sample program shows the utilization of the DPS2 software with the following examples:

- The received output data is filed in a defined memory area (io_byte_ptr).
- As input data, this memory area is read back or mirrored.
- The first byte of this input data influences the diagnostics bits in the manner already described.
- The sample slave has a switched on configuration of 0x13 / 0x23 (that is, 4 bytes I/Q) and can adapt itself to a configuration of 0x11/0x21 that is, 2 bytes I/Q). Based on your application, you must decide the extent to which a configuration change is a good idea
- If 0xAA and 0xAA is in the user-specific parameter data, the sample program will signal

a faulty parameter assignment. The user-specific parameter data is copied to the diagnostics data field.

You can insert your application to the interfaces described. You particularly have to determine and enter the station address via your mechanism (for example, rotary switch, keys, etc.). You can obtain your own device-/manufacturer-specific PNO ident number from the PNO (refer to address list). You can include your own interrupt programs, dependent on the application, in the interrupt routines provided in the source code.

The current state of the DPS2 software is stored on the delivery diskette. Please heed the current implementation instructions in the interface center's mailbox (++49 911 73 79 72).

Copyright (C) Siemens AG 1997. All rights reserved.

7.2 Main Program

The following sample program shows the principal sequence of DPS2 in an application.

```
/*********************************************************************/
/*  D e s c r i p t i o n :                                          */
/*                                                                   */
/*  Main programm                                                    */
/*   In this example the input and output bytes are transfered from/to  */
/*   the I/O area, which is addressed by the io_byte_ptr.            */
/*   The first input byte is used as a service byte for the diagnostics */
/*********************************************************************/

void    main ()

{
/* Reset sequence for the SPC3 and the microprocessor        */
/*   depending of the used hardware application              */
/* - force the Reset Pin                                     */
/* - Set the interrupt parameters of the microprocessor      */
/* - Delete the SPC3 internal RAM */

/* activate the indication functions */

DPS2_SET_IND(GO_LEAVE_DATA_EX | WD_DP_MODE_TIMEOUT | NEW_GC_COMMAND |\

NEW_SSA_DATA | NEW_CFG_DATA | NEW_PRM_DATA | USER_TIMER_CLOCK);
/*set the watchdog value in the SPC3, which supervice the microprocessor */
DPS2_SET_USER_WD_VALUE(20000);

/* In this example the input and output bytes are transferred to the

   IO area, which is addressed by the io_byte_ptr. In the case of the IM183
   there is RAM. */
io_byte_ptr = ((UBYTE*) 0x2E000L);

/* fetch station address */
this_station = OWN_ADDRESS;

/* get Identnumber       */
ident_numb_high = IDENT_HIGH;
ident_numb_low =  IDENT_LOW;


/* Do not allow changing of the slave address by the PROFIBUS DP */
real_no_add_chg = TRUE;

/* Reset User und DPS */
user_dps_reset();

/* Enable SPC3 interrupt in the interupt controller of the microprocessor */

for (;;)

    {   /*=== Begin of the endless loop ===*/
    zyk_wd_state = DPS2_GET_WD_STATE();    /*for info.:  the current WD State*/
    zyk_dps_state = DPS2_GET_DP_STATE();   /*for info.: the current PROFIBUS DP State*/

   /* Trigger user watchdog */
   DPS2_RESET_USER_WD();

/*============ Handling the output data =================*/

   if (DPS2_POLL_IND_DX_OUT()) /* is new output date available? */

        {
   /* Confirm taking over of the output data */
   DPS2_CON_IND_DX_OUT();

   /* Get pointer to the current output data */
   user_output_buffer_ptr = DPS2_OUTPUT_UPDATE();

   /* Example: Copy output data to the IO */
   for (i=0; i<user_io_data_len_ptr->outp_data_len; i++)
          {
   (*((io_byte_ptr) + i)) = (*(((UBYTE DPS2_PTR_ATTR*) user_output_buffer_ptr) + i));

          }
        }

/*============ Handling the input data =================*/

   /* Write input data from the periphery to the ASIC */
```


Copyright (C) Siemens AG 1997. All rights reserved.

```
   for (i=0; i<user_io_data_len_ptr->inp_data_len; i++)
        {
   *(((UBYTE DPS2_PTR_ATTR*) user_input_buffer_ptr) + i) = *((io_byte_ptr) + i);

        }

   /* Give current pointer / data to the SPC3/DPS2 and get a new pointer,

   where the next input data can be written */

   user_input_buffer_ptr = DPS2_INPUT_UPDATE();

/*== Handling external diagnostics and other user-defined actions =====*/
/* ATTENTION:         this is only an example                           */
/* Take the first byte of the input data as a service byte              */
/*  for the change diag function                                        */

   dps_chg_diag_srvc_byte_new = *((UBYTE*)(io_byte_ptr));
   if (user_diag_flag) /* is a diagnostics buffer available? */
        {
      /* Is there a change in the service byte (1.input byte) */
      if (dps_chg_diag_srvc_byte_new == dps_chg_diag_srvc_byte_old)
   { /* no action */}

   else

            {
      /*== Handling the external diagnostics  =====*/
      /* only the least significant 3 byte are used */
      if ((dps_chg_diag_srvc_byte_new & 0x07) !=
   (dps_chg_diag_srvc_byte_old & 0x07))

               {
         /* Mask the 3 bits */
         diag_service_code = dps_chg_diag_srvc_byte_new & 0x07;

         /* Write the length of the diagnostics data to the SPC3 */
         if (dps_chg_diag_srvc_byte_new & 0x01)
   diag_len = 16;   //max. value of the IM308B

   else

   diag_len = 6;

   diag_len = DPS2_SET_DIAG_LEN(diag_len);

         /* Write external diagnostics data to the SPC3 */
         build_diag_data_blk ((struct diag_data_blk *)user_diag_buffer_ptr);

                /* Set the service code             */
                /* 0x01 External diagnostics          */
                /* 0x02 Static   diagnostics          */
         /* 0x01 External diagnostics Overflow */
         DPS2_SET_DIAG_STATE(diag_service_code);

         /* Trigger diagnostics update in the SPC3*/
         DPS2_DIAG_UPDATE();

         /* Store „no diagnostics buffer available“  */
         user_diag_flag = FALSE;
               }
   dps_chg_diag_srvc_byte_old = dps_chg_diag_srvc_byte_new;

            }
        }
/*================= Check the buffers and the state ==================*/

/* Is a new diagnostics buffer available */

   if (DPS2_POLL_IND_DIAG_BUFFER_CHANGED())

        {
   DPS2_CON_IND_DIAG_BUFFER_CHANGED(); /* Confirm the indication */
   user_diag_buffer_ptr = DPS2_GET_DIAG_BUF_PTR(); /* Fetch the pointer */
   user_diag_flag = TRUE; /* Set the Notice „Diag. buffer availble      */
        }

   }   /*=== endless loop        ===*/

}
```


Copyright (C) Siemens AG 1997. All rights reserved.

```
/*********************************************************************/
/*  D e s c r i p t i o n :                                          */
/* Reset the USER and DPS2                                           */
/*********************************************************************/

void user_dps_reset (void)

{
enum DPS2_INIT_RET dps2_init_result;            /* result of the initial. */

DPS2_SET_IDENT_NUMBER_HIGH(ident_numb_high);    /* Set the Identnumber    */
DPS2_SET_IDENT_NUMBER_LOW(ident_numb_low);
DPS2_SET_STATION_ADDRESS(this_station);         /* Set the station address*/
DPS2_SET_HW_MODE(SYNC_SUPPORTED | FREEZE_SUPPORTED | INT_POL_LOW | USER_TIMEBASE_10m);

/* Set div. modes of the SPC3 */

if (!real_no_add_chg)

    {
    DPS2_SET_ADD_CHG_ENABLE();                  /* Open or suspend the    */
    }                                           /* address change service */
else

    {
DPS2_SET_ADD_CHG_DISABLE();
    }

/* initialize the length of the buffers for DPS2_INIT() */
dps2_buf.din_dout_buf_len = 244;
dps2_buf.diag_buf_len = sizeof(struct diag_data_blk);
dps2_buf.prm_buf_len = 20;
dps2_buf.cfg_buf_len = 10;
dps2_buf.ssa_buf_len = 5;

/* initialize the buffers in the SPC3                   */
dps2_init_result = DPS2_INIT(&dps2_buf);
if(dps2_init_result != DPS2_INIT_OK)
   {       /* Failure */}

/* Get a buffer for the first configuration */

real_config_data_ptr = (UBYTE DPS2_PTR_ATTR*) DPS2_GET_READ_CFG_BUF_PTR();

/* Set the length of the configuration data */
DPS2_SET_READ_CFG_LEN(CFG_LEN);

/* Write the configuration bytes in the buffer */
*(real_config_data_ptr) = CONFIG_DATA_INP;      /* Example 0x13 */
*(real_config_data_ptr + 1) = CONFIG_DATA_OUTP; /* Example 0x23 */

/* Store the actuall configuration in RAM for the check in the

   check_configuration sequence (see the modul intspc3.c)       */

cfg_akt[0] = CONFIG_DATA_INP;
cfg_akt[1] = CONFIG_DATA_OUTP;
cfg_len_akt = 2;

/* Calculate the length of the input and output using the configuration bytes*/

user_io_data_len_ptr = dps2_calculate_inp_outp_len (real_config_data_ptr,(UWORD)CFG_LEN);

if (user_io_data_len_ptr != (DPS2_IO_DATA_LEN *)0)

    {
   /* Write the IO data length in the init block */
   DPS2_SET_IO_DATA_LEN(user_io_data_len_ptr);
    }
else

    {
   { /* error */}

    }

/* Fetch the first input buffer */
user_input_buffer_ptr = DPS2_GET_DIN_BUF_PTR();

/* Fetch the first diagnostics buffer, initialize service bytes */
dps_chg_diag_srvc_byte_new = dps_chg_diag_srvc_byte_old = 0;
user_diag_buffer_ptr = DPS2_GET_DIAG_BUF_PTR();
user_diag_flag = TRUE;

/* for info: get the baudrate               */
user_baud_value = DPS2_GET_BAUD();
```


Copyright (C) Siemens AG 1997. All rights reserved.

```
/* Set the Watchdog for the baudrate control */
DPS2_SET_BAUD_CNTRL(0x1E);

/* and finally, at last, los geht's start the SPC3 */
DPS2_START();
}
```


Copyright (C) Siemens AG 1997. All rights reserved.

7.3 Interrupt Program

The following interrupt program shows the sequence in principle of the DPS2 interrupt program in an application.

```
/*****************************************************************************/
/*  D e s c r i p t i o n :                                                  */
/*                                                                           */
/*  dps2_ind                                                                 */
/*                                                                           */
/*  This function is called by the hardware interrupt                        */
/*****************************************************************************/

#if defined __C51__

   void dps2_ind(void)    interrupt 0

#else

   interrupt (0x1b) void dps2_ind(void)    /* CC11 = EX3IN   SAB 165 application */

#endif

{
if(DPS2_GET_IND_GO_LEAVE_DATA_EX())

    {   /*=== Start or the end of the Data-Exchange-State ===*/
    go_leave_data_ex_function();
    DPS2_CON_IND_GO_LEAVE_DATA_EX();    /* confirm this indication */
    }

if(DPS2_GET_IND_NEW_GC_COMMAND())

    {   /*===  New Global Control Command ===*/
    global_ctrl_command_function();
    DPS2_CON_IND_NEW_GC_COMMAND();  /* confirm this indication */
    }

if(DPS2_GET_IND_NEW_PRM_DATA())

    {   /*=== New parameter  data ===*/
    UBYTE   DPS2_PTR_ATTR * prm_ptr;
    UBYTE   param_data_len, prm_result;
    UBYTE   ii;

   prm_result = DPS2_PRM_FINISHED;
   do
      { /* Check parameter until no conflict behavior */
      prm_ptr = DPS2_GET_PRM_BUF_PTR();
      param_data_len = DPS2_GET_PRM_LEN();

      /* data_length_netto of parametration_telegram > 7 */
      if (param_data_len > 7)
            {
                  {
         /* path for wrong user parameter data */
         prm_result = DPS2_SET_PRM_DATA_NOT_OK();
                  }
   else

                  {
         /* correct user parameter data        */
         prm_result = DPS2_SET_PRM_DATA_OK();
                  }
            }
   else

   prm_result = DPS2_SET_PRM_DATA_OK();

   } while(prm_result == DPS2_PRM_CONFLICT);

    }

if(DPS2_GET_IND_NEW_CFG_DATA())

    {   /*=== New Configuration data ===*/
    UBYTE DPS2_PTR_ATTR * cfg_ptr;
    UBYTE i, config_data_len, cfg_result, result;

    cfg_result = DPS2_CFG_FINISHED;
    result = DPS_CFG_OK;
```


Copyright (C) Siemens AG 1997. All rights reserved.

```
    do
        {   /* check configuration data until no conflict behavior m*/
        cfg_ptr = DPS2_GET_CFG_BUF_PTR();             /* pointer to the config_data_block */
   config_data_len = DPS2_GET_CFG_LEN();

   /* check the received configuration data  */

         /*     result = DPS_CFG_OK;
                result = DPS_CFG_UPDATE;
                result = DPS_CFG_FAULT ;           */

   if (result == DPS_CFG_UPDATE)

                    {
   user_io_data_len_ptr = dps2_calculate_inp_outp_len

   (cfg_ptr,(UWORD)config_data_len);

   if (user_io_data_len_ptr != (DPS2_IO_DATA_LEN *)0)

                        {
   DPS2_SET_IO_DATA_LEN(user_io_data_len_ptr);

                        }
   else
   result = DPS_CFG_FAULT;
                    }
                }
   switch (result)

                {
   case DPS_CFG_OK: cfg_result = DPS2_SET_CFG_DATA_OK();

   break;

   case DPS_CFG_FAULT: cfg_result = DPS2_SET_CFG_DATA_NOT_OK();

   break;

   case DPS_CFG_UPDATE: cfg_result = DPS2_SET_CFG_DATA_UPDATE();

   break;

                }
            }

   } while(cfg_result == DPS2_CFG_CONFLICT);

    }

if(DPS2_GET_IND_NEW_SSA_DATA())

    {   /*=== New Slave address received ===*/
    address_data_function(DPS2_GET_SSA_BUF_PTR(), DPS2_GET_SSA_LEN());
    DPS2_CON_IND_NEW_SSA_DATA();    /* confirm this indication */
    }

if(DPS2_GET_IND_WD_DP_MODE_TIMEOUT())

    {   /*=== Watchdog is run out ===*/
    wd_dp_mode_timeout_function();
    DPS2_CON_IND_WD_DP_MODE_TIMEOUT();  /* confirm this indication */
    }
if(DPS2_GET_IND_USER_TIMER_CLOCK())

    {   /*==== Timer tick received ====*/
    DPS2_CON_IND_USER_TIMER_CLOCK();
    }
DPS2_SET_EOI(); /* */
}   /* End dps2_ind() */
```


Copyright (C) Siemens AG 1997. All rights reserved.

8. General

8.1 Addresses

PROFIBUS User Organization

PNO
Agency
Mr. Volz
Haid- und Neu- Strasse 7
76131 Karlsruhe/Germany
Tel.: (0721) 9658-590

Contact persons in the Interface Center in Germany

Siemens AG
AUT 7 B1 T2
Mr. Schmidt X.
Mr. Putschky

Mailing Address:
Postfach 2355
90713 Fuerth/Germany

Street Address
Wuerzburgerstr.121
90766 Fuerth/Germany

Tel.: (0911) 750 -	2079
	2078
Fax: (0911) 750 -	2100
Mailbox: (0911) -	737972

Contact Persons in the Interface Center in the USA

PROFIBUS Interface Center
3000 Bill Garland Road
Johnson City, TN 37605-1255/USA

Fax : (423) 461 - 2016
BBS:(423) 461 - 2751

Your Partner: Tim Black, Rainer Friess
Tel.: (423) 461 - 2332
2687

Copyright (C) Siemens AG 1997. All rights reserved.

Siemens AG
Division Automation Engineering
Combination Engineering
PO Box 23 55, D-90713 Fuerth/Germany

© Siemens AG
Subject to change without prior notice

SIEMENS Aktiengesellschaft

Printed in the Fed. Rep. of Germany
Order No. 6ES7 182-0AA00-8BA0