SIEMENS

SIMATIC NET

DP Slave Programming Interface for CP 5511, CP 5512 and CP 5611 / CP 5611 A2

Manual

C79000-G8976-C126-05 Release 08/2005

SIEMENS

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. Die Angaben in der Druckschrift werden jedoch regelmäßig überprüft. Notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Suggestions for improvement are welcome.
Technische Änderungen vorbehalten.

C79000-G8976-C126-05

We have checked the contents of this manual for agreement with the hardware described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcome.
Technical data subject to change.

C79000-G8976-C126-05
Copyright © Siemens AG 1999 - 2005
All Rights Reserved

Nous avons vérifié la conformité du contenu du présent manuel avec le matériel et le logiciel qui y sont décrits. Or, des divergences n'étant pas exclues, nous ne pouvons pas nous porter garants pour la conformité intégrale. Si l'usage du manuel devait révéler des erreurs, nous en tiendrons compte et apporterons les corrections nécessaires dès la prochaine édition. Veuillez nous faire part de vos suggestions.

Nous nous réservons le droit de modifier les caractéristiques techniques.

C79000-G8976-C126-05
Copyright © Siemens AG 1999 - 2005
All Rights Reserved

Siemens Aktiengesellschaft

Elektronikwerk Karlsruhe
Printed in the Federal Republic of Germany

SIEMENS

SIMATIC NET

DP Slave Programming Interface for CP 5511, CP 5512 and CP 5611 / CP 5611 A2

Manual C79000-B8976-C126-05

Classification of Safety-Related Notices

This document contains notices which you should observe to ensure your own personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:

Danger

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

Warning

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

Caution

with warning triangle indicates that minor personal injury can result if proper precautions are not taken.

Caution

without warning triangle indicates that damage to property can result if proper precautions are not taken.

Notice

indicates that an undesirable result or status can occur if the relevant notice is ignored.

Note

highlights important information on the product, using the product, or part of the documentation that is of particular importance and that will be of benefit to the user.

Siemens AG
Automation and Drives
Industrial Communication
Postfach 48 48, D-90327 Nuernberg

C79000-G8976-C126-05

Siemens AktiengesellschaftPrinted in the Federal Republic of Germany

Trademarks

SIMATIC®, SIMATIC NET® and SINEC® are registered trademarks of Siemens AG.

Third parties using for their own purposes any other names in this document which refer to trademarks might infringe upon the rights of the trademark owners.

Safety Instructions Regarding your Product

Before you use the product described here, read the safety instructions below thoroughly.

Qualified Personnel

Only qualified personnel should be allowed to install and work on this equipment . Qualified persons are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage of Hardware Products

Please note the following regarding the correct usage of hardware products:

Caution

This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

Before you use the supplied sample programs or programs you have written yourself, make certain that no injury to persons nor damage to equipment can result in your plant or process.

EU Directive: Do not start up until you have established that the machine on which you intend to run this component complies with the directive 89/392/EEC.

Correct Usage of Software Products

Please note the following regarding the correct usage of software products:

Caution

This software may only be used for the applications described in the catalog or the technical description, and only in connection with devices or software products from other manufacturers which have been approved or recommended by Siemens.

Before you use the supplied sample programs or programs you have written yourself, make certain that no injury to persons nor damage to equipment can result in your plant or process.

Prior to Startup

Before putting the product into operation, note the following warning:

Caution

Prior to startup you must observe the instructions in the relevant documentation. For ordering data of the documentation, please refer to catalogs or contact your local Siemens representative.

Contents

Note

The table of contents below only includes two levels to preserve clarity.
You will find a detailed table of contents at the beginning of each chapter.

Notes for the User

Documentation This manual describes the DP slave programming interface.

1 Introduction

This description introduces the DP slave programming interface.

With this programming interface, a user program can use the services intended for a DP slave. The DP slave programming interface consists of functions belonging to the "C" programming language, grouped together to form a library (DLL).

The following section provides you with an overview of the properties and features of the DP slave programming interface.

Note

In this document, CP 5511 always means all the CPs from the SOFTNET family; in other words, CP 5511, CP 5512, CP 5611 and CP 5611 A2.
Exception: If information relates to only *one* specific CP type, this is indicated explicitly.

Contents of the Chapter

1.1 Classification

DP Master Class 1

A DP master class 1 communicates cyclically with the DP slaves. The communication includes central functions such as:

- Configuration and assignment of parameters to the slaves
- Cyclic data transfer (input/output) to the DP slaves
- Monitoring of the DP slaves
- Storage of diagnostic information

DP Master Class 2

With the CP 5511, access to the following diagnostic and parameter assignment functions are possible via a DP master class 2:

- Read slave diagnostic data
- Read master diagnostic data
- Read slave inputs
- Read save outputs
- Read slave configuration data
- Set slave address

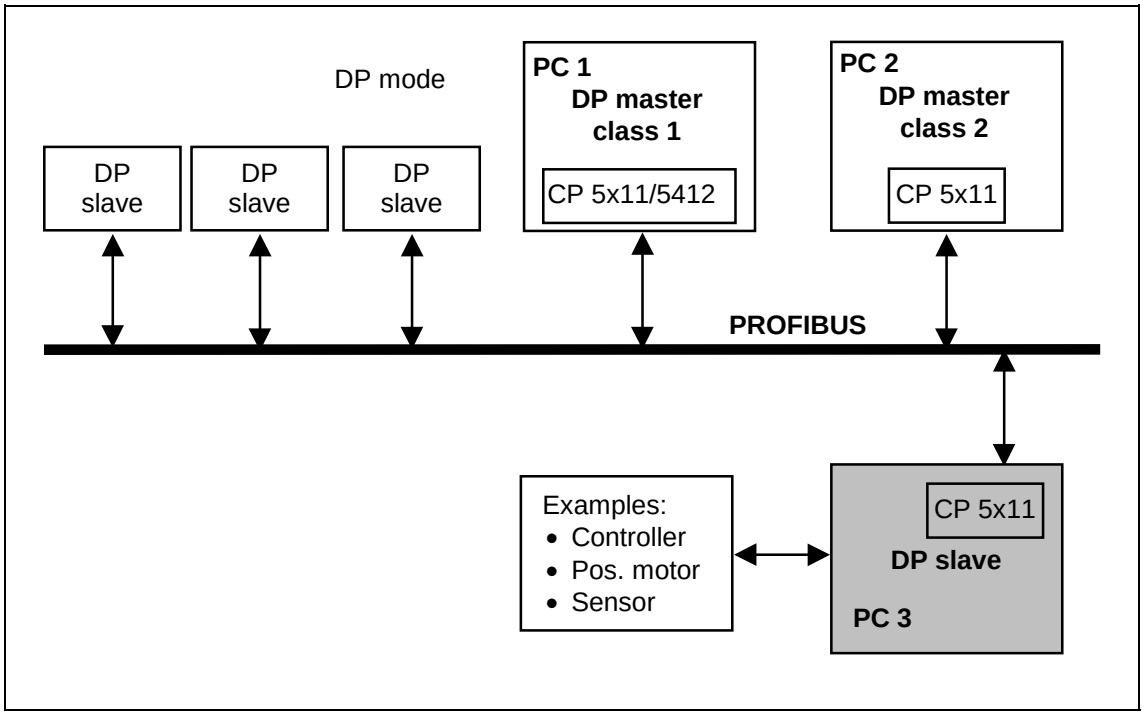
DP Slave

The slave is activated via the DP slave programming interface, configured and assigned parameters by the DP master class 1 and it then exchanges input and output data with the DP master in cyclic data transfer (see DP master class 1).

- Set input data for DP master (max. 122 bytes)
- Read output data of the DP master (max. 122 bytes)
- Start/stop DP slave

Overview of the DP Protocol

The following diagram illustrates how a DP master class 2 is included in a PROFIBUS network.



1.2 Properties of the Programming Interface

Names	The names of the functions and data components are based on the DP specification.
Single User Operation	The programming interface is designed for a call within one thread within a user program. Multi-user operation is not possible.
Multiboard Operation	Not approved.
Operating Systems and Compilers	The programming interface is designed for use with Windows 98, Windows NT 4.0, Windows 2000 Professional, Windows XP and Windows Server 2003. To find out which compiler or development environment is currently supported under which operating system, refer to the section "Supported Development Environments or Compilers" in the "Readme.htm" file of the relevant SIMATIC NET CD.
Synchronous and Parallel Processing	The function calls of the programming interface are synchronous. Synchronous functions only return to the caller after the job has been executed and provide the results of the job. The parallel execution of more than one DP slave service is not possible and is rejected with an error.
Supported Data Type	The byte data type is supported.
Consistency	Byte consistency is supported.
Maximum Data Lengths	A maximum of 122 bytes are possible in both the input and output direction.
Compatibility with the CP 5614	The call interface is compatible with the CP 5614, however, it provides only a subset of the CP 5614 slave functions.
Supported Boards	• CP 5511 • CP 5512 • CP 5611 • CP 5611 A2

2 Description of the Functions

This chapter describes the administrative and productive functions of the DP slave programming interface.

Contents of the Chapter

2.1 Introduction

Functions

The DP slave programming interface consists of a number of C functions that can be grouped as follows:

- Administrative Functions
- Productive functions (functions for data exchange)

Administrative Functions

The following table illustrates the calls for administrative functions.

Administrative Functions	Description
DPS_open	DP slave user program logs on at the interface.
DPS_close	User program logs off at the interface
DPS_stop	Deactivates the slave module (offline).
DPS_start	Activates the slave module (online).

Functions for Data Exchange

The following table illustrates the additional functions for data exchange.

Functions	Description
DPS_set_input	Set input data of the slave (input data for the DP master)
DPS_get_output	Read the output data of the slave. (output data from DP master)

Conventions used in Notation

The following table explains the meaning of the special representations in the following sample programs for the functions:

Representation	Meaning
// in	The value is provided by the user program as input to the DP slave programming interface.
// out	The value is returned to the user program by the DP slave programming interface.

2.2 Logging on a DP Slave Application – DPS_open

Purpose

With this function, the a DPS application logs on at the driver and sets the slave parameters. If successful, the function returns a user handle. The user handle must be included in all further function calls.
The bit "DPS_SM_SIMPLE“ is set in the parameter "slave_mode“; the expected parameter and configuration data are located in the parameter "init_data“.

Syntax

```
DPR_DWORD           DPS_open (
  DPR_STRING        *cp_name,       // in
  DPR_DWORD         *user_handle,   // out
  DPR_DWORD         slave_mode,     // in
  DPR_WORD          station_addr,   // in
  DPR_WORD          addr_change,    // in, not used!
  DPR_WORD          pno_ident_nr,   // in
  DPR_WORD          user_wd,        // in, not used!
  DPS_INIT_DATA_T   *init_data      // in
  DPS_MAX_DATA_T    *max_data_len   // in, not used!
  DPR_WORD          baud_rate       // in
  DP_ERROR_T        *error);        // out
```

Example

```
if (  DPS_open ("DPSONLINE",&user_handle,
DPS_SM_SIMPLE, (DPR_BYTE)StationAddr, 0, 0x8076,
0,&init_data,NULL, baudrate, &error)  == DP_OK)
{ // OK. Further actions }
else
{ // Error handling }
```

Parameters

Name	Description
cp_name	Access point to the hardware - Using the program "Set PG/PC Interface“, a "CP5511 PROFIBUS DP SLAVE“ interface parameter set must be assigned to this access point.
user_handle	Pointer to the user-handle variable - If successful, the user handle assigned to the application is entered here.
slave_mode	DPS_SM_SIMPLE
station_addr	Station address of the slave
addr_change	Always 0: address changes via the bus are not permitted
pno_ident_nr	Unique number for the slave assigned by the PROFIBUS Users Organization (for example 0x0008 for the sample application) when the customer applies for certification.
user_wd	No significance

Table continued on next page

Table continued from previous page

init_data	Pointer to a structure with information on the extended slave data.
	
	`Typedef union DPS_INIT_DATA_S`
	`{`
	`  struct DPS_SIMPLE_S`
	`  {`
	`   DPR_WORD  user_prm_data_len;  // in`
	`   DPR_BYTE  user_prm_data[DPS_MAX_PDU_LEN]; // in`
	`   DPR_WORD  cfg_data_len;  // in`
	`   DPR_BYTE  cfg_data[DPS_MAX_PDU_LEN];  // in`
	`  }simple;`
	
	`  struct DPS_DYNAMIC_S`
	`  {`
	`   DPR_WORD  def_cfg_data_len;  // in`
	`   DPR_BYTE  def_cfg_data[DPS_MAX_PDU_LEN];  // in`
	`  }dynamic;`
	
	`}DPS_INIT_DATA_T`
	
	user_prm_data_len:
	Length of the specified user parameter assignment data : always 0 ! *
	user_prm_data:
	Specified user parameter assignment data : always 0 ! *
	cfg_data_len:
	Length of the specified configuration data
	cfg_data:
	Predefined configuration data from the GSD file
	def_cfg_data_len:
	No significance, (struct DPS_DYNAMIC_S not used)
	def_cfg_data:
	No significance, (struct DPS_DYNAMIC_S not used)
	
	***** user_prm_data are data specific to the application and are appended to the standard section of parameter data (for example to set a mode, measuring range etc. on the slave). With a SIMPLE_SLAVE, the user parameter data of the master are simply compared with the specified parameter data.

Table continued on next page

Table continued from previous page

max_data_len	The structure is not evaluated by the CP 5511 DP slave and is therefore of no significance.
	`Typedef DPS_MAX_DATA_S`
	`{`
	`  DPR_BYTE  max_input_data_len;  // in`
	`  DPR_BYTE  max_output_data_len;  // in`
	`  DPR_BYTE  max_user_diag_len;  // in`
	`  DPR_BYTE  max_user_prm_data_len;  // in`
	`  DPR_BYTE  max_cfg_data_len;  // in`
	`  DPR_BYTE  max_user_ssa_data_len;  // in`
	`}DPS_MAX_DATA_T`

baud_rate	Transmission rates that can be set: The following constants are defined:
	`DPS_BD_9K6  //  9.6  Kbps`
	`DPS_BD_19K2  //  19.2  Kbps`
	`DPS_BD_45K45  //  45.45 Kbps`
	`DPS_BD_93K75  //  93.75 Kbps`
	`DPS_BD_187K5  //  187.5  Kbps`
	`DPS_BD_500K  //  500  Kbps`
	`DPS_BD_1M5  //  1.5  Mbps`
	`DPS_BD_3M  //  3  Mbps`
	`DPS_BD_6M  //  6  Mbps`
	`DPS_BD_12M  //  12  Mbps`

error	Address of a structure provided by the user program of the type DP_ERROR_T. If an error occurred, the structure contains details for troubleshooting.

Return Values

Name	Description
DP_OK	Successful completion of the function
other	Incorrect completion - the error IDs correspond to the function. For more detailed information, refer to Chapter 4.

Note

Make sure that the node address of the slave does not already exist on the bus.

This function is synchronous.

The PROFIBUS address of the slave and the baud rate of the bus transferred with the *DPS_open* function are ignored. The existing settings for these parameters are retained.

2.3 Logging off a DP Slave Application – DPS_close

Purpose

With this function, a DP user program logs off for communication at the slave module.

Syntax

```
DPR_DWORD  DPS_close(
    DPR_DWORD   user_handle,  // in
    DP_ERROR_T  *error );     // out
```

Parameters

Name	Description
user_handle	User handle assigned with the DPS_open call.
error	Address of a structure provided by the user program of the type DP_ERROR_T. If an error occurred, the structure contains details for troubleshooting.

Return Values

Name	Description
DP_OK	Successful completion of the function
other	Incorrect completion - the error IDs correspond to the function. For more detailed information, refer to Chapter 4.
Note	If the logoff was successful, the user handle is no longer valid and must not continue to be used. This function is synchronous.

2.4 Switching the Slave Online – DPS_start

Purpose

- This function can be used to switch the slave online. This is necessary after initialization. After switching the slave OFFLINE with DPS_stop, the slave module can be switched online again with DPS_start.
- Following the DPS_start call, the DP slave is ready to receive the parameter assignment and configuration frames of the DP masters and to change to the DATA_EXCHANGE state after successful parameter assignment and configuration. If DPS_get_output or DPS_set_input calls are sent immediately following DPS_start, but the DP slave is still in the startup phase, these calls will be acknowledged with DPS_ERROR_EV_NO_DATA_EX (see Section 4.1.1) until the startup phase is completed and the slave changes to the DATA_EXCHANGE state.

Syntax

```
DPR_DWORD    DPS_start  (
     DPR_DWORD   user_handle,          // in
     DP_ERROR_T     *error);           // out
```

Parameters

Name	Description
user_handle	User handle assigned with the DPS_open call.
error	Address of a structure provided by the user program of the type DP_ERROR_T. If an error occurred, the structure contains details for troubleshooting.

Return Values

Name	Description
DP_OK	Successful completion of the function
other	Incorrect completion - the error IDs correspond to the function. For more detailed information, refer to Chapter 4.
Note	This function is synchronous.

2.5 Switching the Slave Offline – DPS_stop

Purpose

This function is used to deactivate the slave module. The slave module then no longer reacts on the bus. To reactivate the slave, the DPS_start function must be used.

Note

The DPS_stop function starts to shut down the DP slave. It is possible that the function is acknowledged with DP_OK, although the slave has not yet completed the change to the OFFLINE state.

Syntax

```
DPR_DWORD    DPS_stop  (
     DPR_DWORD   user_handle,    // in
     DP_ERROR_T    *error );    // out
```

Parameters

Name	Description
user_handle	User handle assigned with the DPS_open call.
error	Address of a structure provided by the user program of the type DP_ERROR_T. If an error occurred, the structure contains details for troubleshooting.

Return Values

Name	Description
DP_OK	Successful completion of the function
other	Incorrect completion - the error IDs correspond to the function. For more detailed information, refer to Chapter 4.
Note	This function is synchronous.

2.6 Setting Input Data of the Slave (input data for master) – DPS_set_input

Purpose

This function sets input data of a slave (input data for the DP master).

If this service is sent immediately after a DPS_start, it is possible that the slave has not yet reached the DATA_EXCHANGE state, in other words, the parameter assignment and configuration by the DP master is not yet completed and data exchange is not yet possible. The call is acknowledged during this time with DPS_ERROR_EV_NO_DATA_EX (see also Section 4.1.1).

Data Consistency

Byte consistency is guaranteed for data input.

Syntax

```
DPR_DWORD DPS_set_input(
     DPR_DWORD      user_handle,      // in
     DPR_BYTE       *data_s,          // in
     DPR_WORD       len,              // in
     DP_ERROR_T     *error );         // out
```

Parameters

Name	Description
user_handle	User handle assigned with the DPS_open call.
data_s	Input data of the slave
len	Data length (maximum 122 bytes)
error	Address of a structure provided by the user program of the type DP_ERROR_T. If an error occurred, the structure contains details for troubleshooting.

Return Values

Name	Description
DP_OK	Successful completion of the function
other	Incorrect completion - the error IDs correspond to the function. For more detailed information, refer to Chapter 4.
Note	This function is synchronous.

2.7 Getting the Output Data of the Slave (output data of the master) – DPS_get_output

Purpose

This function gets the output data of the slave (output data of the DP master).

If this service is sent **immediately** after a DPS_start, it is possible that the slave has not yet reached the DATA_EXCHANGE state, in other words, the parameter assignment and configuration by the DP master is not yet completed and data exchange is not yet possible. The call is acknowledged during this time with DPS_ERROR_EV_NO_DATA_EX (see also Section 4.1.1).

Data Consistency

Byte consistency is guaranteed for data output.

Syntax

```
DPR_DWORD  DPS_get_output
     (DPR_DWORD    user_handle,          // in
      DPR_BYTE     *data_s,              // out
     DPR_WORD      len,                  // in
      DP_ERROR_T   *error );             // out
```

Parameters

Name	Description
user_handle	User handle assigned with the DPS_open call.
data_s	Output data of the slave
len	Data length (maximum 122 bytes)
error	Address of a structure provided by the user program of the type DP_ERROR_T. If an error occurred, the structure contains details for troubleshooting.

Return Values

Name	Description
DP_OK	Successful completion of the function
other	Incorrect completion - the error IDs correspond to the function. For more detailed information, refer to Chapter 4.
Note	This function is synchronous.

3 Structure of a DP Slave Application

This chapter describes the following:

- Why a GSD file is required
- How a DP slave application can be structured
- Call sequences of DP slave services

Contents of the Chapter

3.1 Requirements

GSD File

A GSD file must be provided by the developer of a DP slave to describe the properties of the product.

It is necessary both for configuration and certification. The file format specified for the GSD files (GSD is an acronym from the German meaning default device database) allows open configuration tools to be implemented for PROFIBUS DP.

The specified keywords and the fixed structure of a GSD file can be found in the PROFIBUS standard EN 50170, Part 2, or in the appropriate literature.

Caution

Please note the following:

The GSD file supplied with the demonstration program contains the description of the demonstration slave (with fixed configuration parameters). The ID number is a default number reserved for SIEMENS. Its use is permitted only for test purposes with the demonstration application.

For more detailed information on the structure of a GSD file for the DP slave in SOFTNET, refer to Section 4.2

GSD File Preparation

To implement "your" slave with this programming interface, you must describe the slave with its GSD file.

GSD File and Configuration Tool

The GSD file must be made known to your configuration tool for example SIEMENS COM PROFIBUS).

For more information, refer to the documentation of your configuration tool.

Then create a database containing the slave and configure your PROFIBUS DP network with this database.

Certification

If you want to certify your DP slave (with the corresponding GSD file), you must first apply for a free identification number from the PNO.

3.2 Example of a DP Slave Application

Procedure

The following section outlines the basic structure of a DP slave application.

Step	Description
1	Bind include files and libs
2	Create variables
3	Log on with the DP slave LIB (DPS_open)
4	Start the slave (switch ONLINE) (DPS_start)
5	Send one or more DP slave jobs
6	Stop slave (switch OFFLINE) (DPS_stop)
7	Log off from the DP slave LIB (DPS_close)

Step 1

Bind include files and libs

Include the header files.

You must also make sure that the dpslib.lib is linked to your project.

```
#include "dps_fkt.h"  // includes dps_user.h too
```

Step 2

Create variables

Creating data buffers and structures:

```
DPR_DWORD         user_handle;
DPR_WORD          station_address;
DPR_WORD          pno_ident_number = 1234;
//example.
DP_ERROR_T        error;
DPS_INIT_DATA_T   init_data;
DPR_WORD          baudrate = 0x03;
```

Step 3 **Log on at the DP slave LIB**

Before you can use the DP slave services, you must log on at the DP slave lib.

```
error.error_code = 0;
//example configuration
init_data.simple.cfg_data[0] = 0x3f;
init_data.simple.cfg_data[1] = 0x35;
init_data.simple.cfg_data[2] = 0x1f;
init_data.simple.cfg_data[3] = 0x23;
// The GSD file must include:
// Module = "module1" 0x3f, 0x35, 0x1f, 0x23
// see Section 4.2
init_data.simple.cfg_data_len = 4;
init_data.simple.user_prm_data_len = 0; //always 0!
init_data.simple.user_prm_data = 0;     //always 0!
// Example here:
// 0x3f = 16 inputs  /  16 outputs
// 0x35 =  6 inputs  /   6 outputs
// 0x1f = 16 inputs  /   0 outputs
// 0x23 =  0 inputs  /   4 outputs
// Total  38 inputs  /  26 outputs
// Bit coding of .cfg_data :
// Bit 0 to 3 = length+1
// Bit 4 = 1 : input
// Bit 5 = 1 : output
// Bit 6, 7 always 0
               .
               .
               .
if (  DPS_open ("DPSONLINE", &user_handle,
DPS_SM_SIMPLE, (DPR_BYTE)StationAddr, 0,
pno_ident_number, 0, &init_data, NULL, baudrate,
&error)  == DP_OK)
{ // OK. Further actions }
else
{ // Error handling }
```

Note

The values in init_data.simple.cfg_data[n] must match the values in the GSD file.

The lengths specified for the services DPS_get_output and DPS_set_input must match the values of the configuration parameters and GSD file exactly. In other words, in the example above DPS_get_output(..., len,...) with len = 26 and DPS_set_input(...,len,...) with len = 38 .

Step 4 **Start the slave (switching online)**

Send DPS_start.

```
DP_ERROR_T error;
error.error_code = 0;
if ( DPS_start(user_handle, &error) == DP_OK )
     {.......} // OK. Further actions
else {.......} // Error handling
```

Step 5 **Send a DP slave job, for example read output data**

```
DP_ERROR_T error;
DPR_BYTE   OutputData[OUTPUT_SIZE];
DPR_WORD   len;
      .
      .
error.error_code = 0;
if (DPS_get_output(user_handle, OutputData, len,
&error) ==   DP_OK)
     {.......} // OK. Further actions
else {.......} // Error handling
```

Step 6 **Stop slave (switch offline)**

```
DP_ERROR_T error;
error.error_code = 0;
if ( DPS_stop(user_handle, &error) == DP_OK )
     {.......} // OK. Further actions
else {.......} // Error handling
```

Step 7 **Log off from the DP slave LIB**

Before you can terminate the application, it must log-off:

```
DP_ERROR_T error;
error.error_code = 0;
if ( DPS_close(user_handle, &error) == DP_OK )
     {.......} // OK. Further actions
else {.......} // Error handling
```

Caution

Please note the following: Check the return values of the DP slave functions for any errors.

4 Additional DP Slave Information

This chapter describes the following:

- The meaning of the error bits and how to evaluate them
- General information on the range of functions of the DP slave
- General information on the operating systems
- General information on the administrative services

Contents of the Chapter

4.1 Evaluating Errors

Purpose

The error identifiers for the individual jobs have a uniform structure DP_ERROR. If an error occurs, the various elements return the exact description of the error. The declarations are in the include file "dps_user.h“ .

Syntax

```
typedef struct DP_ERROR
     {
     DPR_DWORD error_class;
     DPR_DWORD error_code;
     DPR_BYTE     error_decode;
     DPR_BYTE     error_code_1;
     DPR_BYTE     error_code_2;
     } DP_ERROR_T;
```

error_class Structure Element

The error_class structure element specifies the general error class. The entry in error_class is **identical** to the return value of the function call.

A DP slave application can therefore use either the error_class structure element or the return value of the function call.

The possible error classes are listed in Table 4-1.

error_code Structure Element

The error_code structure element is relevant in the error classes:

- DP_ERROR_EVENT_NET
- DP_ERROR_REQ_PAR
- DP DP_ERROR_RES

error_decode Structure Element

The error_decode structure element is relevant in the error classes:

- DP_ERROR_RES
- DP_ERROR_REQ_PAR

Structure Elements error_code_1, error_code_2

These elements are currently not used.

How to Evaluate Errors

The following table shows which entries are returned by a job in the error_class structure element.

In some situations, further information can be evaluated

- In the error_code structure elements.
- In the error_decode structure element.

Description of Table 4-1

The following Table 4-1 describes the entries in the error_class structure element..

Entry in the error_class Structure Element	Description	Further Information
DP_ERROR_EVENT_NET	No communication possible with the slave	Table 4-2
DP_ERROR_REQ_PAR	Incorrect transfer parameter of a call or call illegal	Table 4-3
DP_ERROR_RES	Not enough resources from system or layer 2	Table 4-4
DP_OK	Job completed without error	

Table 4-1 Explanation of the error_class Structure Element

4.1.1 Entries in the error_code and error_decode Structure Elements

Description of Table 4-2

The following Table 4-2 describes the entries in the error_code structure element. These belong to the error codes of the **DP_ERROR_EVENT_NET** class.

Entry in error_code	Description
DPS_ERROR_EV_NO_DATA_EX	DP slave is not in the DATA_EXCHANGE state
DPS_ERROR_EV_START_FAILED	Start command could not be executed
DPS_ERROR_SIM9SYNC_FAILED	There may be a problem in your installation. Repeat the installation and if the problem recurs, contact our hotline.
DPS_ERROR_EV_COMM_ERR	Bad SCP_Send. For example, DPS_open with bus short-circuit (physical bus defect).

Table 4-2 Error Coding of the DP_ERROR_EVENT_NET Class

Description of Table 4-3

The following Table 4-3 describes the entries in the error_code structure element. These belong to the error codes of the **DP_ERROR_REQ_PAR** class

Entry in error_code	Description
DPS_ERROR_PAR_INTERNAL *****	Internal error. Please call the Hotline.
DPS_ERROR_PAR_SLAVE_MODE	Slave mode not permitted
DPS_ERROR_PAR_ADDR_CHANGE	ADDR_CHANGE not permitted
DPS_ERROR_PAR_USER_HANDLE	Invalid user handle
DPS_ERROR_PAR_STATION_ADDR	Station address higher than 126 not permitted
DPS_ERROR_PAR_INPUT_DATA	No pointer to input data
DPS_ERROR_PAR_OUTPUT_DATA	No pointer to output data
DPS_ERROR_PAR_BAUDRATE	Wrong transmission rate
DPS_ERROR_REQ_NOT_ALLOWED	Request not permitted in this state
DPS_ERROR_REQ_DPS_ACTIVE	Request not permitted temporarily since active request not completed.
DPS_ERROR_PAR_OUTPUT_LEN	DPS_get_output (): "Output data length" parameter does not match the length specified in the configuration data.
DPS_ERROR_PAR_INPUT_LEN	DPS_set_input (): "Input data length" parameter does not match the length specified in the configuration data.
DPS_ERROR_PAR_IO_LEN	DPS_open (): Too many I/Os; maximum 122 permitted.
DPS_ERROR_PAR_CFG_DATA ******	Bad configuration data: No outputs/inputs configured
*** Additional entry in error_decode for DPS_ERROR_PAR_INTERNAL**	**Description**
DPS_ERROR_INTERNAL_SCP_PARAM	Error on the SCP interface
DPS_ERROR_INTERNAL_AMPRO2_LOAD	AMPRO2 driver could not be loaded
DPS_ERROR_INTERNAL_AMPRO2_OPEN	AMPRO2 driver could not be opened

Table continued on next page

Table continued from previous page

** Additional entry in error_decode for DPS_ERROR_PAR_CFG_DATA	Description
Index	error_decode contains the index of the bad CFG byte (bit 4 or 5 in the CFG byte not set)
	For more detailed information, refer to the frequently asked questions (FAQ) for your product or contact your Siemens representative.

Table 4-3 Error Coding of the DP_ERROR_REQ_PAR Class

Description of Table 4-4

The following Table 4-4 describes the entries in the error_code structure element. These belong to the error codes of the **DP_ERROR_RES** class

Entry in error_code	Description
DPS_ERROR_RES_OPEN	No further SCP_OPEN possible
DPS_ERROR_RES_ALLOC *	Error in layer 2 resource or system resource allocation
DPS_ERROR_RES_DEALLOC **	Error in layer 2 resource or system resource deallocation
DPS_ERROR_RES_REG_BAUDRATE	Transmission rate could not be written to the registry.
DPS_ERROR_RES_REG_TS	Station address could not be written to the registry.
DPS_ERROR_RES_REG_LOG_DEVICE	Logical device or name not found in the registry.
DPS_ERROR_RES_REG_TSLOT	Writing the bus parameter Tslot to the registry failed
*** Additional entry in error_decode for DPS_ERROR_RES_ALLOC**	**Description**
DPS_ERROR_RES_ALLOC_SYS	Error in system resource allocation
DPS_ERROR_RES_ALLOC_AMPRO2	Error in layer 2 resource allocation
**** Additional entry in error_decode for DPS_ERROR_RES_DEALLOC**	**Description**
DPS_ERROR_RES_DEALLOC_SYS	Error in system resource deallocation
DPS_ERROR_RES_DEALLOC_AMPRO2	Error in layer 2 resource deallocation

Table 4-4 Error Coding of the DP_ERROR_RES Class

4.2 Access to the CP 5511, Range of Functions of the DP Slave, GSD File

Board

With the DPS_open function, an access point is selected in the "cp_name“ parameter. For example, "DPSONLINE“ or any other string (you can also define your own string) .

Settings in Set PG/PC Interface

A CP 5511 interface parameter set must be assigned to the access point in the "Set PG/PC Interface“ tool.

Example
"CP 5511 (PROFIBUS DP Slave)“.

This assigns a CP 5511 PCMCIA card to your DP slave.

Further Settings

No further settings are necessary with the "Set PG/PC Interface“ tool.

All the remaining parameters necessary for operating the slave are transferred using the "DPS_open (params)“ service.

Range of Functions

The CP 5511 DP slave provides you with a simple and clear programming interface, with which you can create an application with little effort.

Your slave application can be certified in the PROFIBUS interface center (SSC) in Fürth, Germany.

The interface of the CP 5511 DP slave does not currently provide the following (optional) functions:

1. DP-V1 functions
2. automatic data transmission rate recognition
3. SYNC/FREEZE

GSD File

A GSD file describes the functional properties of a DP slave. The specified keywords and the fixed structure of a GSD file can be found in the PROFIBUS standard EN 50170 (Volume 2), or in the appropriate literature.

If you create your own GSD file, base it on the sample file supplied.

Creating and modifying GSD files

When creating a GSD file for a SOFTNET DP slave, note the following points:

- In the configuration data of the submodule definition, only use "bytes" for the length of the data units.
- In the configuration data of the submodule definition, only use "byte consistency" for the consistency.
- Do not enter any empty submodules.
- The maximum data length per submodule is 16 bytes.
- The total length of all inputs or outputs is 122 bytes.
- If you use the GSD file "siem8076.gsd", remember that this is only an example. If you want to create a user program that uses the OPC interface, you must change the Ident number in the GSD file from Ident_Number=0x8076 to 0x9001.
- Fixed values must not be changed.
- Changes can lead to system malfunctions!
- Do not modify any values that are not marked as modifiable.
- Only PNO standard ID formats are supported.
- Additional parameters are not supported. Specifying additional parameters can lead to malfunctions.

GSD File Fixed and Variable Settings

A GSD file describes the functional properties of a DP slave. Some parameters **can** be changed if required and some **must** be adapted. Others, on the other hand, have fixed values that must not be changed. The GSD file of the demonstration program is shown below and indicates the status of the settings.

Note that multiple module definitions are not permitted.
The configuration data are specified in bytes separated by commas and entered after the value module = "module1".
Example: 1-byte input and 1-byte output appear as follows:

```
Module = "module1" 0x10, 0x20
```

The configuration data of the SOFTNET DP slave must not contain any empty slots (byte ID 00).

```
#Profibus_DP
; Example: CP5511 slave with input and output data length of 8 bytes.
; Unit-Definition-List:
GSD_Revision=1                      ; can be modified
Vendor_Name="SIEMENS AG"            ; can be modified
Model_Name="SN DEMO Slave122b"      ; must be modified
Revision="V1.0"                     ; can be modified
Ident_Number=0x8076                 ; must be modified
Protocol_Ident=0                    ; fixed
Station_Type=0                      ; fixed
Hardware_Release="V2.0"             ; can be modified
Software_Release="V1.0"             ; can be modified
9.6_supp=1                          ; supported transmission rates are fixed
19.2_supp=1                         ; fixed
45.45_supp=1                        ; fixed
93.75_supp=1                        ; fixed
187.5_supp=1                        ; fixed
500_supp=1                          ; fixed
1.5M_supp=1                         ; fixed
3M_supp=1                           ; fixed
6M_supp=1                           ; fixed
12M_supp=1                          ; fixed

; fixed values dependent on transmission rate
MaxTsdr_9.6=60                      ; fixed
MaxTsdr_19.2=60                     ; fixed
MaxTsdr_45.45=400                   ; fixed
MaxTsdr_93.75=60                    ; fixed
MaxTsdr_187.5=60                    ; fixed
MaxTsdr_500=100                     ; fixed
MaxTsdr_1.5M=150                    ; fixed
MaxTsdr_3M=250                      ; fixed
MaxTsdr_6M=450                      ; fixed
MaxTsdr_12M=800                     ; fixed
Implementation_Type="ASPC2"; fixed
;Bitmap_Device=" "                  ; can be modified

; Slave-Specification:
OrderNumber="6AV3 xx7-1xxxx-xxxx" ;must be modified
Auto_Baud_supp=0                    ; fixed
Sync_Mode_supp=0                    ; fixed
Freeze_Mode_supp=0                  ; fixed
Min_Slave_Interval=20               ; fixed
Max_Diag_Data_Len=10                ; fixed
Modul_Offset=0                      ; fixed
Slave_Family=3@TdF@B + B ; optional information. Example here: The slave
; appears in COM PROFIBUS in a catalog called B + B.;
Max_Module=1                        ; fixed
Max_Input_Len=8           ; must be adapted. Possible values: 0 to 122
Max_Output_Len=8          ; must be adapted. Possible values: 0 to 122
;
User_Prm_Data_Len=0                 ; fixed
;
; Module-Definitions:
Module = "module1" 0x37
; Configuration : 8 bytes inputs/ 8 bytes outputs
; these bytes must be entered as cfg_data, see page 26
EndModule
```

5 Checklist for Programmers

Please note the following points when programming applications with the DP slave programming interface.

Program Structure

Make sure that you keep to the rules below when structuring your program:

- Start your application with a DPS_open and always complete it with a DPS_close.
- Remember that DP slave functions are synchronous. After the job is sent and completed, the return values are already available. Fetching the data later is not necessary.
- Only one job can be sent at any one time.
- Evaluate all reported errors.

Data Areas

Use the checklist below for your data areas:

- Select data buffer lengths to match the longest possible data (122 bytes).
- Use static data areas for your job fields and data areas.
- Link job fields with valid data buffers in the startup phase of your program.

Asynchronous Mode

Remember that jobs can only be sent to the DP slave lib from one thread of one application. The next job can only be sent after the previous job has been executed.

A Few Last Points

You should also keep in mind the following points:

- In your program, assume that “reserved” job and return parameters will be used in future for expansions. Assign a default value (0) to “reserved” elements.
- In bit-oriented return parameters, the reserved bits are also intended for future expansions. When checking these bit-oriented return parameters, mask reserved bits (for example by ANDing them with the value 0).
- Read the information in the FAQ for your product.
- Make sure that the lengths you specify for "DPS_set_input“ or "DPS_get_output“ match the values of your slave configuration for "DPS_open“ and the values of your GSD file.
- Note the information about the startup behavior of the DP slave with the services DPS_start, DPS_set_input and DPS_get_output. The startup behavior of the DP slave may be different. This depends, for example, on whether the DP master or the DP slave was started first.
- The current state of the slave is transferred with the return value. No status query is necessary. If the slave is not in the DATA_EXCHANGE state, no data exchange with the DP master is possible.
  Return Value DPS_ERROR_EV_NO_DATA_EX (see Section 4.1.1).

6 Formats of the Slave Data

This chapter describes the formats of the slave data.

Contents of the Chapter

6.1 Formats of the Input and Output Data

Order of the Slave Data

The order of the data corresponds to the configured input/output ports of the DP slaves.
For example, the input ports of an ET 200B-16DI station are stored as follows: Port 0 in the first byte, Port 1 in the second byte, etc.

Format of Data Words

The following order is necessary for storing values in the word format (2-byte numbers):

First, the high byte (lower order address) of the word is entered followed by the low byte (higher order address).

Note that the SOFTNET DP slave supports only byte consistency.

6.2 Formats of the Slave Diagnostic Data

Introduction

In some cases, in the following data structures single bits in the bytes are significant. The numbering of the bits is always as follows: The least significant bit (LSB) has the number 0 and the most significant bit (MSB) has the number 7.

Note

The user program cannot access the diagnostic data.

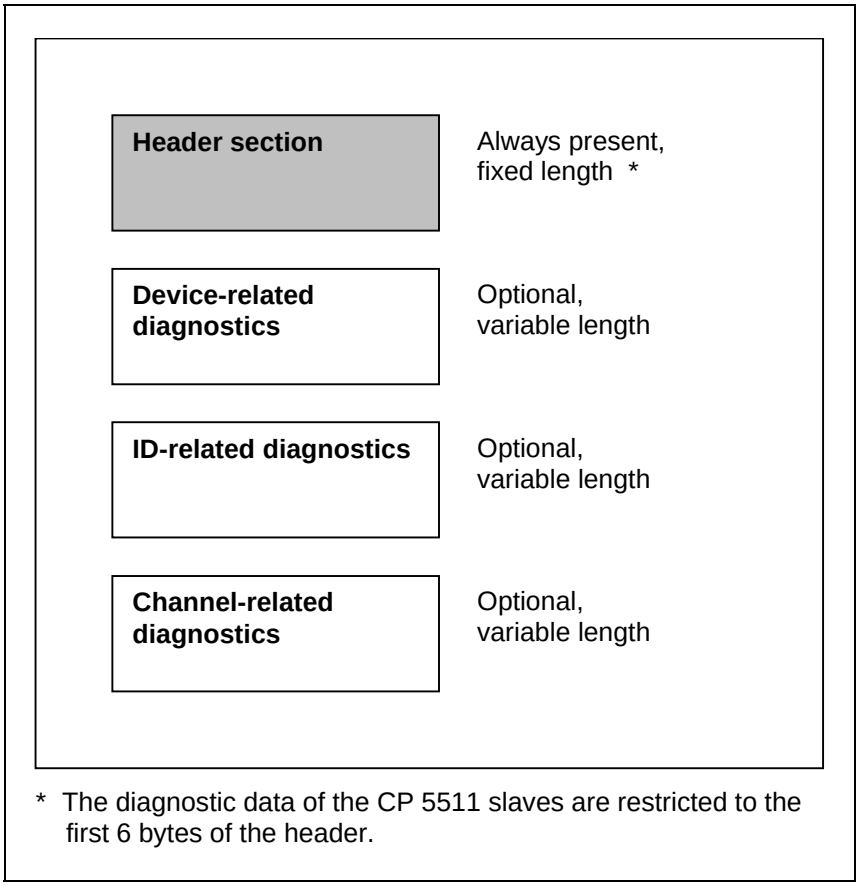
6.2.1 Overview of the Entire Structure

Possible Length

The typical length of diagnostic data is between 6 and 32 bytes. The maximum possible length is 244 bytes.

Header and up to Three Following Fields

Diagnostic data have a fixed header and up to three further fields:



6.2.2 Format of the Diagnostic Data Header

Overall Structure of the Header

byte	Meaning
Byte 1	Stationstatus_1
Byte 2	Stationstatus_2
Byte 3	Stationstatus_3
Byte 4	Diag.Master_Add
Byte 5 and 6	Ident_Number

Byte 1 (Stationstatus_1)

Each bit in byte 1, the "station_status_1“ byte has a special meaning.

Bit	Meaning
7	**Diag. Master_Lock,** The DP slave has already been assigned parameters by another master, in other words the local master does not currently have access to this slave.
6	**Diag. Prm_Fault,** This bit is set by the slave if the last parameter assignment frame was incorrect (for example wrong length, wrong ident number, invalid parameters).
5	**Diag. Invalid_Slave_Response,** This bit is set as soon as an implausible response is received from an addressed DP slave.
4	**Diag. Not_Supported,** This bit is set as soon as a function is requested that is not supported by the slave (for example operation in the SYNC mode although the slave does not support this).
3	**Diag. Ext_Diag,** This bit is set by the DP slave. If the bit is set, there must be a diagnostic entry in the slave-specific diagnostic area (Ext_Diag_Data). If the bit is not set, there may be a status message in the slave-specific diagnostic area (Ext_Diag_Data). The meaning of the status message must be negotiated with each specific user program.
2	**Diag. Cfg_Fault,** This bit is set when the configuration data last sent by the master do not match those determined by the DP slave, in other words when there is a configuration error.
1	**Diag. Station_Not_Ready,** This bit is set when the DP slave is not yet ready for the productive phase.
0	**Diag. Station_Non_Existent,** The DP master sets this bit, if the DP slave is not obtainable on the bus. If this bit is set, the diagnostic bits contain the status of the last diagnostic message or the initial value. The DP slave always sets this bit to zero.

Byte 2 (Stationstatus_2)

Each bit in byte 2, the "station_status_2“ byte has a special meaning.

Bit	Meaning
7	**Diag. Deactivated,** This bit is set as soon as the DP slave is indicated as inactive in the local parameter record and was taken out of cyclic processing.
6	reserved
5	**Diag. Sync_Mode,** This bit is set by the DP slave as soon as it receives the Sync control command.
4	**Diag. Freeze_Mode,** This bit is set by the DP slave as soon as it receives the Freeze control command.
3	**Diag. WD_On (Watchdog on),** This bit is set by the DP slave. If this bit is set to 1, the watchdog monitoring is activated on the DP slave.
2	The DP slave always sets this bit to 1.
1	**Diag.Stat_Diag (Static Diagnostics)** - If the DP slave sets this bit, the DP master must fetch diagnostic information until the bit is cleared. The DP slave, for example, sets this bit when it is not capable of providing valid user data.
0	**Diag. Prm_Req,** If the DP slave sets this bit, it must be reassigned parameters and be reconfigured. This bit remains set until parameters have been assigned (if bit 1 and bit 0 are set, bit 0 has the higher priority).

Byte 3 (Stationstatus_3)

bit	Meaning
7	**Diag. Ext_Diag_Overflow,** If this bit is set, there is more diagnostic data than can be indicated. The DP slave sets this bit, for example, when there is more channel diagnostic data than the DP slave can enter in its send buffer. The DP master also sets this bit when the DP slave sends more diagnostic data than the DP master can accommodate in its diagnostic buffer.
6 to 0	reserved

Byte 4 (Diag. Master_Add)

The address of the DP master that assigned parameters to the DP slave is entered here. If the DP slave has not been assigned parameters by a DP master, the DP slave sets the address 255 in this byte.

Bytes 5 and 6 (Ident_Number)

The vendor ID for a DP slave type is assigned in bytes 5 and 6, the "Ident_Number“ byte. This identifier can be used on the one hand for test purposes and on the other for precise identification of the slave.

Glossary

COM PROFIBUS	Configuration tool for configuring PROFIBUS DP systems
CP	Communications Processor
CP 5511	The SOFTNET CPs: • CP 5511 (PCMCIA card), CP5512 (PC-CARD card) • CP 5611 (PCI card), CP 5611 A2 (PCI-X card)
CP 5614	Powerful CP with its own CPU and hardware-controlled DP preprocessing.
DP	Distributed peripheral I/Os - fast, serial transmission protocol for connecting a central programmable controller with distributed I/O devices.
GSD file	Device database file (.GSD) contain DP slave descriptions complying with EN 50170 or DIN E 19245 Part 3 .
MCL1	Master class 1
MCL2	Master class 2
SIMATIC	Siemens automation system
SIMATIC NET	SIMATIC network architecture for automation and engineering - there are three versions of this architecture: • SIMATIC NET Industrial Ethernet • SIMATIC NET PROFIBUS • SIMATIC AS-i - bus at the actuator-sensor level
SSC	Interface Center in Fürth, Germany; certification center

Index

❑