



Protocol API

PROFIBUS-MPI

Language: English

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC071001API04EN | Revision 4 | English | 2011-12 | Development | Public

Revision History

Rev	Date	Name	Revisions
1	24.10.07	AW/RG	Created Firmware/stack version 0.96.0
2	14.10.08	RG	Firmware/ stack version V2.1.8 Warmstart -> Set Configuration Added section on structure of protocol stack Many smaller corrections and enhancements Changed some error numbers to global error numbers
3	15.11.10	AB/RG	Firmware/ stack version V2.3.1 Reference to netX Dual-Port Memory Interface Manual Revision 9. Added description of 3 new packets in MPI task section: <ul style="list-style-type: none"> • PROFIBUS_MPI_CMD_DC_ALL_REQ/CNF – • PROFIBUS_MPI_CMD_MULTIPLE_READ_REQ/CNF - Multiple Read Request • PROFIBUS_MPI_CMD_MULTIPLE_WRITE_REQ/CNF - Multiple Write Request Added description of 4 new packets in DL task section: <ul style="list-style-type: none"> • PROFIBUS_DL_CMD_SEND_FDL_STATUS_REQ - Send FDL Status • PROFIBUS_DL_CMD_GET_LMS_REQ/CNF - Get List of active Stations • PROFIBUS_DL_CMD_REGISTER_LMS_REQ/CNF - Register an Application to receive LMS Change Indications • PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for Change in LMS Revised sections 4.1 and 4.2 A new stack structure diagram has been integrated (see <i>Figure 2</i> on page 19). Additions (missing ranges of values and default values) and corrections in description of bus parameters. Some error messages have been added, new formatting of error messages. Added list of figures

Rev	Date	Name	Revisions
4	08.12.11	RG	<p>Firmware/ stack version V2.4.1</p> <p>Reference to netX Dual-Port Memory Interface Manual Revision 12.</p> <p>Added description of packet PROFIBUS_DL_CMD_GET_LL_REQ/CNF – Get the Life List (List of Active or Passive Stations)</p> <p>Added description of packet PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T - Close a specific Socket Request/Confirmation</p> <p>Changed some incorrect data types</p> <p>Small text corrections at the following packets:</p> <ul style="list-style-type: none"> • PROFIBUS_DL_CMD_GET_LMS_REQ/CNF - Get List of active Stations • PROFIBUS_DL_CMD_REGISTER_LMS_REQ/CNF - Register an Application to receive LMS Change Indications • PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for Change in LMS

Table of Contents

1	Introduction	9
1.1	Abstract	9
1.2	System Requirements	9
1.3	Intended Audience	9
1.3.1	Technical Data	10
1.4	Terms, Abbreviations and Definitions	11
1.5	References	11
1.6	Legal Notes	12
1.6.1	Copyright	12
1.6.2	Important Notes	12
1.6.3	Exclusion of Liability	13
1.6.4	Export	13
2	Getting started / Configuration	13
2.1	Configuration of Bus Parameters	13
2.1.1	Detailed Description of Bus Parameters	13
3	Overview	18
3.1	Profibus MPI and the OSI/ISO Layer Model	18
3.2	Task Structure of the PROFIBUS MPI Stack	19
3.3	Functionality of the MPI and MPI-AP Task	21
3.3.1	Acyclic Data Transfer	21
3.3.2	Configuration via Database	21
3.3.3	Status information	21
3.4	Functionality of the DL Task (Layer 2)	22
3.4.1	Data Transfer Services	23
3.4.2	Management of Services Access Points	26
3.4.3	Management of Layer 2 System Variables and Tasks	28
4	The Application Interface	29
4.1	MPI AP Task	29
4.1.1	PROFIBUS_MPI_AP_CMD_SET_CONFIG_REQ/CNF – Set Configuration Parameter	30
4.2	MPI Task	33
4.2.1	PROFIBUS_MPI_CMD_INIT_REQ/CNF - Initialization of PROFIBUS MPI Protocol Stack	34
4.2.2	PROFIBUS_MPI_CMD_SET_BUS_PARAM_REQ/CNF - Load the Bus Parameter Set	36
4.2.3	PROFIBUS_MPI_CMD_RW_DB_REQ/CNF - Read/Write Data Block (Daten Baustein)	42
4.2.4	PROFIBUS_MPI_CMD_RW_ME_REQ/CNF - Read/Write Merker	46
4.2.5	PROFIBUS_MPI_CMD_RW_IO_REQ/CNF – Read/Write Input/Output	50
4.2.6	PROFIBUS_MPI_CMD_RW_CN_REQ/CNF - Read/Write Counter	54
4.2.7	PROFIBUS_MPI_CMD_RW_TI_REQ/CNF - Read/Write Timer	57
4.2.8	PROFIBUS_MPI_CMD_R_OP_REQ/CNF – Read Operation State	60
4.2.9	PROFIBUS_MPI_CMD_RW_TR_REQ/CNF – Send Transparent	63
4.2.10	PROFIBUS_MPI_CMD_DC_REQ/CNF - Disconnect Station	66
4.2.11	PROFIBUS_MPI_CMD_DC_ALL_REQ/CNF – Disconnect All Stations	68
4.2.12	PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T - Close a specific Socket Request/Confirmation	70
4.2.13	PROFIBUS_MPI_CMD_MULTIPLE_READ_REQ/CNF - Multiple Read Request	72
4.2.14	PROFIBUS_MPI_CMD_MULTIPLE_WRITE_REQ/CNF - Multiple Write Request	78
4.3	Packets of the PROFIBUS-DL-Task	84
4.3.1	PROFIBUS_DL_CMD_END_PROCESS_REQ/CNF– Finish the DL Task	86
4.3.2	PROFIBUS_DL_CMD_START_DLE_REQ/CNF – Starts the DL-Layer	88
4.3.3	PROFIBUS_DL_CMD_STOP_DLE_REQ/CNF – Stop DL Layer	90
4.3.4	PROFIBUS_DL_CMD_DATA_ACK_REQ/CNF - Send SDA Service	92
4.3.5	PROFIBUS_DL_CMD_DATA_ACK_IND - Receive SDA Service	99
4.3.6	PROFIBUS_DL_CMD_DATA_REQ/CNF - Send SDN Service	101
4.3.7	PROFIBUS_DL_CMD_DATA_IND - Receive SDN Service Indication	107
4.3.8	PROFIBUS_DL_CMD_DATA_REPLY_REQ/CNF - Send SRD Service	110
4.3.9	PROFIBUS_DL_CMD_DATA_REPLY_IND - Receive SRD Service Indication	117
4.3.10	PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_REQ/CNF - Reply Update Service	120
4.3.11	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ/CNF - Activate an SAP for Request	124
4.3.12	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ/CNF - Activate an SAP for Responder Functionality	130

4.3.13	PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ/CNF - Deactivate an SAP for Request	135
4.3.14	PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ/CNF - Load the Bus Parameter Set	138
4.3.15	PROFIBUS_DL_CMD_SET_VALUE_REQ/CNF - Set Value Service	144
4.3.16	PROFIBUS_DL_CMD_SEND_FDL_STATUS_REQ - Send FDL Status	147
4.3.17	PROFIBUS_DL_CMD_GET_LMS_REQ/CNF - Get List of active Stations	150
4.3.18	PROFIBUS_DL_CMD_REGISTER_LMS_REQ/CNF - Register an Application to receive LMS Change Indications.....	153
4.3.19	PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for Change in LMS	156
4.3.20	PROFIBUS_DL_CMD_GET_LL_REQ/CNF – Get the Life List (List of Active or Passive Stations)	158
5	Status/Error Codes Overview	161
5.1	PROFIBUS MPI-Error Codes	161
5.2	Error Codes of the MPI AP-Task.....	162
5.3	Error Codes of the DL-Task	163
5.3.1	Diagnostic Codes of the DL-Task.....	165
6	Contact.....	166

List of Figures

Figure 1: PROFIBUS in the OSI/ISO Layer Model..... 18

Figure 2: Internal Structure of PROFIBUS MPI Firmware..... 19

Figure 3: SDA Service with Acknowledgement..... 23

Figure 4: Service Access Points 26

List of Tables

Table 1: Terms, Abbreviations and Definitions	11
Table 2: References.....	11
Table 3: Bus and Master Parameters, their Meanings and their Ranges of allowed Values.....	15
Table 4: Supported Baud Rates.....	16
Table 5: Packets for Acyclic Data Transfer	21
Table 6: Communication State.....	21
Table 7: DL-Task process queue	22
Table 8: Overview over Packets of the PROFIBUS MPI-AP Task	29
Table 9: Variable ulSystemFlags	30
Table 10: structure PROFIBUS_MPI_AP_PACKET_SET_CONFIG_REQ_T	31
Table 11: structure PROFIBUS_MPI_AP_PACKET_SET_CONFIG_CNF_T	32
Table 12: Overview over the Packets of the PROFIBUS MPI-Task	33
Table 13: Request structure PROFIBUS_MPI_PACKET_INIT_REQ_T	34
Table 14: Confirmation structure PROFIBUS_MPI_PACKET_INIT_CNF_T	35
Table 15: Standard values of the MPI parameters.....	36
Table 16: Request structure PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ - Load the Bus Parameter Set	39
Table 17: Confirmation structure PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_CNF - Confirmation of Loading the Bus Parameter Set	41
Table 18: Request structure PROFIBUS_MPI_PACKET_RW_DB_REQ_T	43
Table 19: Confirmation structure PROFIBUS_MPI_PACKET_RW_DB_CNF_T	45
Table 20: Request structure PROFIBUS_MPI_PACKET_RW_ME_REQ_T	47
Table 21: Confirmation structure PROFIBUS_MPI_PACKET_RW_ME_CNF_T	49
Table 22: Request structure PROFIBUS_MPI_PACKET_RW_IO_REQ_T	51
Table 23: Confirmation structure PROFIBUS_MPI_PACKET_RW_IO_CNF_T	53
Table 24: Request structure PROFIBUS_MPI_PACKET_RW_CN_REQ_T	55
Table 25: Confirmation structure PROFIBUS_MPI_PACKET_RW_CN_CNF_T	56
Table 26: Request structure PROFIBUS_MPI_PACKET_RW_TI_REQ_T	58
Table 27: Confirmation structure PROFIBUS_MPI_PACKET_RW_TI_CNF_T	59
Table 28: Request structure PROFIBUS_MPI_PACKET_READ_OP_REQ_T	61
Table 29: Confirmation structure PROFIBUS_MPI_PACKET_READ_OP_CNF_T	62
Table 30: Request structure PROFIBUS_MPI_PACKET_TRANSPARENT_REQ_T	64
Table 31: Confirmation structure PROFIBUS_MPI_PACKET_TRANSPARENT_CNF_T	65
Table 32: Request structure PROFIBUS_MPI_PACKET_MPI_DC_REQ_T - Disconnect structure.....	66
Table 33 Confirmation structure PROFIBUS_MPI_PACKET_MPI_DC_CNF_T - Confirmation of Disconnect Request	67
Table 34: Request structure PROFIBUS_MPI_PACKET_DC_ALL_REQ_T - Close all Sockets Request Structure	68
Table 35 Confirmation structure PROFIBUS_MPI_PACKET_DC_ALL_CNF_T - Confirmation of Disconnect all open Handlers Request.....	69
Table 36: Request structure PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T - Close all Sockets Request...	70
Table 37 Confirmation structure PROFIBUS_MPI_PACKET_CLOSE_SOCKET_CNF_T - Confirmation of Close all Sockets Request	71
Table 38: PROFIBUS_MPI_PACKET_MPI_MULTIPLE_READ_CMD_REQ_T - Multiple Read Request	73
Table 39 PROFIBUS_MPI_PACKET_MPI_MULTIPLE_READ_CMD_CNF_T - Confirmation of Multiple Read Request	76
Table 40: Request structure PROFIBUS_MPI_PACKET_MPI_MULTIPLE_WRITE_CMD_REQ_T - Disconnect structure	79
Table 41 Confirmation structure PROFIBUS_MPI_PACKET_MPI_MULTIPLE_WRITE_CMD_CNF_T - Confirmation of Multiple Write Request	82
Table 42: structure PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_CNF_T	83
Table 43: Overview over the Packets of the PROFIBUS DL-Task of the PROFIBUS MPI Protocol Stack	85
Table 44: PROFIBUS_DL_CMD_END_PROCESS_REQ- Finish the DL Task.....	86
Table 45: PROFIBUS_DL_CMD_END_PROCESS_CNF- Finish the DL Task.....	87
Table 46: PROFIBUS_DL_CMD_START_DLE_REQ - Start the DL-Layer Request	88
Table 47: PROFIBUS_DL_CMD_START_DLE_CNF - Confirmation of Start the DL-Layer Request	89
Table 48: PROFIBUS_DL_CMD_STOP_DLE_REQ- Stop DL Layer Request.....	90
Table 49: PROFIBUS_DL_CMD_STOP_DLE_CNF - Confirmation of Stop DL Layer Request.....	91
Table 50: PROFIBUS_DL_CMD_DATA_ACK_REQ - Send SDA Service	93
Table 51: PROFIBUS_DL_CMD_DATA_ACK_CNF - Confirmation of Send SDA Service	96
Table 52: PROFIBUS_DL_CMD_DATA_ACK_CNF - Packet Status/Error	97

Table 53: Reported Errors, their Sources and Possible Actions	98
Table 54: PROFIBUS_DL_CMD_DATA_ACK_IND - Receive SDA Service.....	100
Table 55: PROFIBUS_DL_CMD_DATA_REQ - Send SDN Service Request	102
Table 56: PROFIBUS_DL_CMD_DATA_CNF – Confirmation of Send SDN Service Request	104
Table 57: PROFIBUS_DL_CMD_DATA_CNF - Packet Status/Error	105
Table 58: Reported Errors, their Sources and Possible Actions	106
Table 59: PROFIBUS_DL_CMD_DATA_IND - Receive SDN Service Indication.....	109
Table 60: PROFIBUS_DL_CMD_DATA_REPLY_REQ - Send SRD Service Request	111
Table 61: PROFIBUS_DL_CMD_DATA_REPLY_CNF – Confirmation of Send SRD Service Request	114
Table 62: PROFIBUS_DL_CMD_DATA_REPLY_CNF - Packet Status/Error	115
Table 63: Reported Errors, their Sources and Possible Actions	116
Table 64: PROFIBUS_DL_CMD_DATA_REPLY_IND - Receive SRD Service Indication.....	119
Table 65: Allowed Values and their Meanings for Variable bUpdateStatus	119
Table 66: PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_REQ - Reply Update Service.....	121
Table 67: PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_CNF – Confirmation of Reply Update Service.....	123
Table 68: PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ - Activate a SAP for Request.....	126
Table 69: Values of bServiceActivate Parameter and the according Service.....	127
Table 70: Values of bRoleInService Parameter and the according Operation	127
Table 71: Allowed combinations of bMaxDLSDULenReqLow, bMaxDLSDULenReqHigh, bMaxDLSDULenIndCnfLow and bMaxDLSDULenIndCnfHigh parameters for services SDA and SDN .	128
Table 72: Allowed combinations of bMaxDLSDULenReqLow, bMaxDLSDULenReqHigh, bMaxDLSDULenIndCnfLow and bMaxDLSDULenIndCnfHigh parameters for services SRD and MSRD	128
Table 73: PROFIBUS_DL_CMD_DLSAP_ACTIVATE_CNF - Confirmation of Activate a SAP for Request	129
Table 74: PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ - Activate a SAP for Responder Functionality	132
Table 75: Allowed combinations of bMaxDLSDULenReqLow, bMaxDLSDULenReqHigh, bMaxDLSDULenIndLow and bMaxDLSDULenIndHigh parameters for service SRD with role in service 'Responder' .	132
Table 76: Allowed Values and their Meanings for Variable bUpdateStatus	133
Table 77: PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_CNF – Confirmation of Activate a SAP for Responder Functionality.....	134
Table 78: PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ - Deactivate an SAP for Request	136
Table 79: PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_CNF – Confirmation of Deactivate an SAP for Request .	137
Table 80: PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ - Load the Bus Parameter Set.....	140
Table 81: PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_CNF – Confirmation of Loading the Bus Parameter Set	143
Table 82: PROFIBUS_DL_CMD_SET_VALUE_REQ - Set Value Service	145
Table 83: PROFIBUS_DL_CMD_SET_VALUE_CNF – Confirmation of Set Value Service	146
Table 84: Possible Values of the FDL Status and their Meanings	147
Table 85: PROFIBUS_DL_CMD_SEND_FDL_STATUS_REQ - Send FDL Status	148
Table 86: PROFIBUS_DL_CMD_SEND_FDL_STATUS_CNF – Confirmation of Send FDL Status Request	149
Table 87: PROFIBUS_DL_CMD_GET_LMS_REQ - Get List of active Stations	151
Table 88: PROFIBUS_DL_CMD_GET_LMS_CNF - - Confirmation of Get List of active Stations Request.....	152
Table 89: PROFIBUS_DL_CMD_REGISTER_LMS_REQ - Register an Application to receive LMS Change Indications	154
Table 90: PROFIBUS_DL_CMD_REGISTER_LMS_CNF – Confirmation of Register an Application to receive LMS Change Indications Request	155
Table 91: PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for acknowledged connectionless Data Transfer	157
Table 92: Possible Values of Single Bytes in Response.....	158
Table 93: PROFIBUS_DL_CMD_GET_LL_REQ - Get Life List Request	159
Table 94: PROFIBUS_DL_CMD_GET_LL_CNF - Confirmation to Get Life List Request.....	160
Table 95: MPI Error Codes	162
Table 96: Error Codes of the MPI AP-Task.....	162
Table 97: Error Messages of the DL-Task	164
Table 98: Diagnostic Messages of the DL-Task.....	165

1 Introduction

1.1 Abstract

This manual describes the application interface of the PROFIBUS-MPI stack, with the aim to support and lead you during the integration process of the given stack into your own application.

Base of the development of the stack itself is the Hilscher's Task Layer Reference Programming Model. It is a description of how to program a Task in general, which is defined as a combination of appropriate functions belonging to the same type of protocol layer. It furthermore defines of how different Tasks have to communicate with each other in order to exchange their layer information in between. The reference model is commonly used by all programmers at Hilscher and shall be used by you as well when writing your Application Task on top of the stack.

1.2 System Requirements

The software package has the following system requirements to its environment:

- netX-Chip as CPU hardware platform
- Operating system for task scheduling required

1.3 Intended Audience

This manual is suitable for software developers with the following background:

- Knowledge of the programming language C
- Knowledge of the use of the real-time operating system rcX
- Knowledge of the Hilscher Task Layer Reference Model

Specifications

The data below applies to PROFIBUS-MPI firmware and stack version V2.4.1.

1.3.1 Technical Data

Maximum number of MPI connections	126
Maximum telegram length for write	212 bytes
Maximum telegram length for read	222 bytes
Functions	<p>MPI read/write DB, M, I (read only), O, C, T Data type bit to access to DB (data block), M (marker), Q (output) and I (Input, read only)</p> <p>MPI connect automatically, when the first read/write function is used</p> <p>MPI disconnect, MPI disconnect all</p> <p>MPI Get OP status</p> <p>MPI transparent (expert use only)</p>
Baud rate	<p>Fixed values from 9,6 kBits/s to 12 MBit/s</p> <p>Auto-detection mode is supported.</p>
Data transport layer	PROFIBUS FDL

Firmware/stack available for netX

netX 50	yes
netX 100, netX 500	yes

Configuration

Configuration by tool SyCon (exported configuration file named `config.dbm`).

Configuration by packet to transfer parameters.

Diagnostic

Firmware does not support common and extended diagnostic in the dual-port-memory for loadable firmware.

Information

The firmware has a packet interface. The IO area of dual-port memory is not used.

1.4 Terms, Abbreviations and Definitions

Term	Description
AP	Application on top of the Stack
DL	Data Link Layer
DPM	Dual-port Memory
MPI	Multi-Point Interface
DB	Datenbaustein (Data Block)
ME	Merker (Flag)
CN	Counter
TI	Timer

Table 1: Terms, Abbreviations and Definitions

All variables, parameters and data used in this manual have basically the LSB/MSB ("Intel") data representation. This corresponds to the convention of the Microsoft C Compiler.

1.5 References

This document is based on the following specifications:

1	DPM Interface Manual for netX based Products, Revision 12; Hilscher GmbH; 2011
2	Profibus DP Master Protocol API Manual; Hilscher GmbH; 2007-2011

Table 2: References

1.6 Legal Notes

1.6.1 Copyright

© 2006-2010 Hilscher Gesellschaft für Systemautomation mbH

All rights reserved.

The images, photographs and texts in the accompanying material (user manual, accompanying texts, documentation, etc.) are protected by German and international copyright law as well as international trade and protection provisions. You are not authorized to duplicate these in whole or in part using technical or mechanical methods (printing, photocopying or other methods), to manipulate or transfer using electronic systems without prior written consent. You are not permitted to make changes to copyright notices, markings, trademarks or ownership declarations. The included diagrams do not take the patent situation into account. The company names and product descriptions included in this document may be trademarks or brands of the respective owners and may be trademarked or patented. Any form of further use requires the explicit consent of the respective rights owner.

1.6.2 Important Notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

1.6.3 Exclusion of Liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

1.6.4 Export

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different countries, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product

2 Getting started / Configuration

This chapter explains some essential information you should know when starting to work with the PROFIBUS-MPI Protocol API.

2.1 Configuration of Bus Parameters

This section explains how to configure the bus parameter correctly. This can be done either by sending packets transferring the parameters or with SyCon.

2.1.1 Detailed Description of Bus Parameters

The PROFIBUS MPI network needs to be configured. The accurate choice of the bus parameters is the foundation of any correctly operating data exchange on the PROFIBUS MPI network. These parameters are important in the context of the following functions:

- PROFIBUS_MPI_AP_CMD_SET_CONFIG_REQ/CNF – Set Configuration Parameter
- PROFIBUS_MPI_CMD_SET_BUS_PARAM_REQ/CNF - Load the Bus Parameter Set
- PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ/CNF - Load the Bus Parameter Set

Bus Parameter Set

```

typedef __TLR_PACKED_PRE struct PROFIBUS_DL_BUS_PARAMETER_SET_Ttag {
    TLR_UINT16 usBus_Para_Len; /* Contains the length of the Bus_Para */
                                /* inclusive the field Bus_Para_Len itself */
    TLR_UINT8  bDL_Add; /* Contains the own address of the PROFIBUS Device */
    TLR_UINT8  bData_rate; /* Contains the Transmission speed */
    TLR_UINT16 usTSL; /* slot-time */
    TLR_UINT16 usMin_TSDR; /* min. station delay responder */
    TLR_UINT16 usMax_TSDR; /* max. station delay responder */
    TLR_UINT8  bTQUI; /* quite time */
    TLR_UINT8  bTSET; /* setup time */
    TLR_UINT32 ulTTR; /* target rotation time */
    TLR_UINT8  bG; /* Gap update factor */
    TLR_UINT8  bHSA; /* Highest station address */
    TLR_UINT8  bMax_Retry_Limit; /* retries if error occurs */

    /* The additional parameters are not relevant for PROFIBUS MPI */
    /* and will ignore if they are present. */
    TLR_UINT8  bBp_Flag;
    TLR_UINT16 usMin_Slave_Interval; /* Minimum Slave Interval Time */
    TLR_UINT16 usPoll_Timeout; /* Class2 Poll timeout */
    TLR_UINT16 usData_Control_Time; /* Data Control Time */
    TLR_UINT8  bMasterSetting; /* 0 == big Endian, 1 == little Endian */
                                /* (swapping should be done) */
    TLR_UINT8  bMax_User_Global_Control; /* Maximum allowed parallel active */
                                /* USER Global Control Commands */
    TLR_UINT8  abReserved[4]; /* 4 reserved Octets */
    TLR_UINT16 usMaster_User_Data_Len; /* Contains the length of the USER */
                                /* Master data */
    TLR_UINT8  abMaster_Class2_Name[32]; /* Name of the Master */
    TLR_UINT8  abMaster_User_Data[32]; /* USER specific Parameter data */
    TLR_UINT32 ulTCL; /* Isochronous cycle time */
    TLR_UINT8  bMax_TSH; /* Maximum Shift Time */
} __TLR_PACKED_POST PROFIBUS_DL_BUS_PARAMETER_SET_T;

```

The following table contains relevant information about the bus parameters for the PROFIBUS MPI firmware such as a short explanation of the meaning of the parameter and ranges of allowed values:

MPI Bus Parameters

Parameter	Meaning	Range of Values/ Default Value
Bus_Para_Len	Contains the length of the Bus_Para including the field Bus_Para_Len itself	>=19 / 19
DL_Add	Contains the own address of the PROFIBUS Device	0-126 / No default
Baud_rate	Contains the transmission speed	9.6 kBaud – 12 MBaud / 187.5 kBaud
T _{SL}	Slot-time	37-16383 / 415
Min T _{SDR}	Min. station delay responder	11-255 / 20
Max T _{SDR}	Max. station delay responder	12 - 65535 / 400
T _{QUI}	Quiet time	0-127 / 0
T _{SET}	Setup time	1-255 / 12
T _{TR}	Target rotation time	255 – (2 ²⁴ -1) / 9984
G	GAP update factor	1-255 / 5
HSA	Highest station address	1-126 / 31
bMax_Retry_Limit	retries if error occurs	1-15 / 2
The additional parameters are not relevant for PROFIBUS MPI and will ignore if they are present.		

Table 3: Bus and Master Parameters, their Meanings and their Ranges of allowed Values



Note: All Default values refer to a baudrate of 187.5 kBaud, this is the default setting for ports with are used for MPI communication.

In detail, the bus parameters are:

2.1.1.1 usBus_Para_Len

This parameter is calculated as the length of the bus parameter structure in bytes including the field itself. Default value is 19

2.1.1.2 bDL_Add

This parameter is used for defining the address of the PROFIBUS DP master itself. Must smaller or equal to HSA



Note: Configuration of addresses in a PROFIBUS network must be performed in such a way that all addresses are used uniquely. No addresses may appear twice or more often within the PROFIBUS network!



Note: Do not use the address 0 as the master address although this would both be possible and allowed because this address is often used by configuration and diagnosis devices. Therefore, it is a good idea to choose a non-zero address for the PROFIBUS Master.

2.1.1.3 bData_rate

This parameter contains the baud rate (i.e. the transmission speed of data signals) used by all stations of the PROFIBUS network in a coded manner. This value must be set to the same value at all slave

stations within the network. If this is not the case, all stations with deviating transmission speed setting will not operate correctly.

Supported Baud Rates

Symbolic Name	Baud rate	Value
PROFIBUS_DL_DATA_RATE_96	9,6 kBits/s	0
PROFIBUS_DL_DATA_RATE_19_2	19,2 kBits/s	1
PROFIBUS_DL_DATA_RATE_93_75	93,75 kBits/s	2
PROFIBUS_DL_DATA_RATE_187_5	187,5 kBits/s	3 (default)
PROFIBUS_DL_DATA_RATE_500	500 kBits/s	4
PROFIBUS_DL_DATA_RATE_1500	1500 kBits/s	6
PROFIBUS_DL_DATA_RATE_3000	3000 kBits/s	7
PROFIBUS_DL_DATA_RATE_6000	6000 kBits/s	8
PROFIBUS_DL_DATA_RATE_12000	12000kBits/s	9
PROFIBUS_DL_DATA_RATE_31_25	31.25kBits/s	10
PROFIBUS_DL_DATA_RATE_45_45	45.45 kBits/s	11
PROFIBUS_DL_DATA_RATE_AUTO	Auto-detection mode.	15

Table 4: Supported Baud Rates

2.1.1.4 usTSL (Slot Time)

This parameter defines the 'Wait for receipt' monitoring time, i.e. the time the master will wait for immediate response or confirmation after sending a request. If the slot time has passed and no response arrived, the telegram will be sent again until the maximum retry limit (i.e. maximum allowed number of repetitions of sending the telegram) has been reached.

This time is identical to the time interval within which the master must either send a request telegram or give the token to the next station.

Allowed values for the slot time range from 37 to 16383. The default value is 415.

2.1.1.5 usMin_TSDR

This is the shortest time that must elapse before a remote recipient (Responder) may send an acknowledgement of a received query telegram. The shortest time between the reception of the last Bit of a telegram to the sending of the first Bit of a following telegram.

Allowed values for the slot time range from 11 to 255. Default value is 20.

2.1.1.6 usMax_TSDR

This is the longest time that must elapse before a sender (Requestor) is allowed to send a further query telegram. The largest time between reception of the last Bit of a telegram to the sending of the first bit of a following telegram. The sender (Requestor, Master) must wait at least for this time after the sending of an unacknowledged telegram (e.g. Broadcast only) before a new telegram is sent.

Allowed values for the slot time range from 12 to 65535 and must be larger than usMin_TSDR. Default value is 400

2.1.1.7 bTQUI (Quiet Time)

The quiet time is defined as the time delay that occurs for modulators (Modulator-trip time) and Repeaters (Repeater-switch time) for the change over from sending to receiving.

Allowed values for the slot time range from 0 to 127 and must be less than usMin_TSDR. Default value is 0

2.1.1.8 bSET (Setup Time)

The setup time represents the minimum period “reaction time” between the receipt of an acknowledgement to the sending of a new query telegram (Reaction) by the Sender (Requestor).

Allowed values for the slot time range from 1 to 255. Default value is 12.

2.1.1.9 uLTTR (Target Rotation Time)

The target rotation time is defined as the pre-set nominal token cycling time within the sender authorization (Token) will cycle around the ring. How much time the Master still has available for sending data telegrams to the Slaves is dependent on the difference between the nominal and the actual token cycling time.

Allowed values for the target rotation time range from 255 to $2^{24}-1$ (=16.777.215). Default value is 9984.

2.1.1.10 bG (GAP Actualization Factor)

The GAP Actualization Factor is applied for determining after how many token cycles an added participant is accepted into the token ring. After expiry of the time $G \cdot TTR$, the station searches to see whether a further participant wishes to be accepted into the logical ring.

Allowed values for the GAP Actualization Factor range from 1 to 255. Default value is 5.

2.1.1.11 bHSA (Highest Station Address)

The ‘Highest Station Address’ parameter represents the highest bus address up to which a Master searches for another Master at the bus in order to pass on the Token. On no account, this station address must be smaller than the Master station address.

Allowed values for the highest station address range from 1 to 126. Default value is 31

2.1.1.12 bMax_Retry_Limit

As already mentioned above, this parameter is the maximum allowed number of repetitions in order to send the telegram before sending will be aborted. Allowed values for the maximum retry limit range from 1 to 15. Default value is 2.



Note: All other parameters of data structure PROFIBUS_DL_BUS_PARAMETER_SET_T are not relevant for MPI and should be set to 0.

3 Overview

3.1 Profibus MPI and the OSI/ISO Layer Model

As the picture below illustrates, Profibus MPI does not affect the layers 3, 4, 5 and 6 of the OSI/ISO reference model of data communication within a network. It only specifies

- Layer 1 (Physical layer)
- Layer 2 (Data link layer)
- Layer 7 (Application layer)

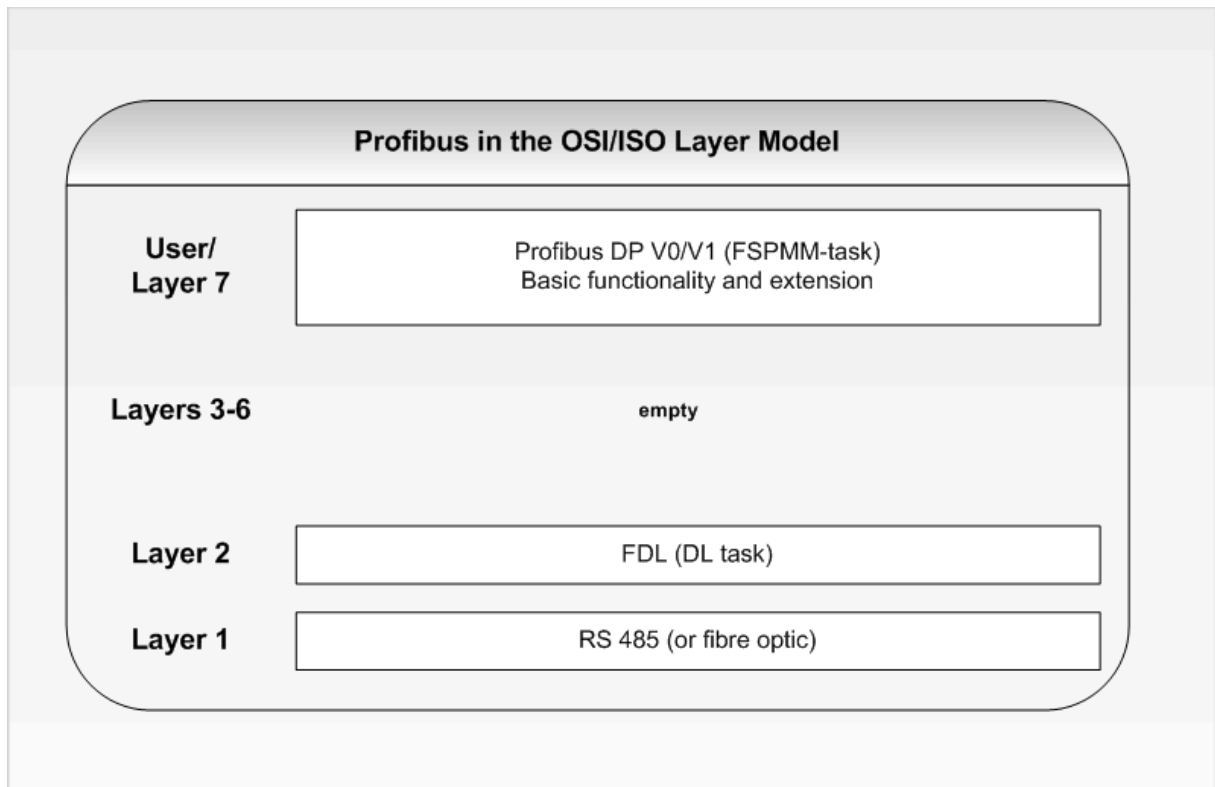


Figure 1: PROFIBUS in the OSI/ISO Layer Model

3.2 Task Structure of the PROFIBUS MPI Stack

The illustration below displays the internal structure of the tasks which together represent the PROFIBUS MPI Stack:

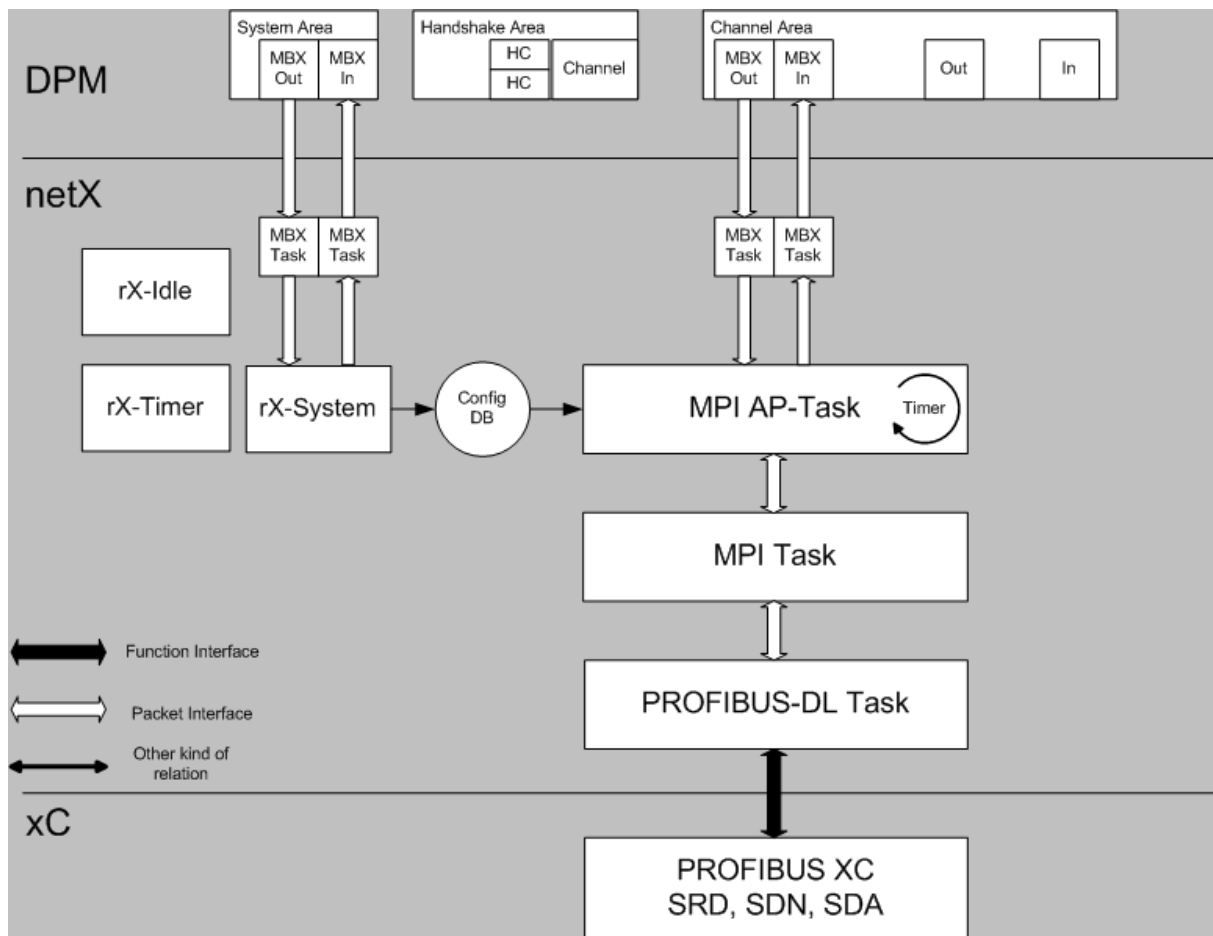


Figure 2: Internal Structure of PROFIBUS MPI Firmware

Explanation of the different kinds of arrows:

- The thick filled (red) arrows indicate data transfer via packet interface.
- The thin (blue) arrows directly below IN and OUT indicate access or update to a triple buffer. This is controlled via handshake cells. The thin (blue) arrows directly above box representing the AP task indicate access from the application task to the triple buffers. For each cycle new data are provided.
- The black curves within the boxes "AP Task" and "PROFIBUS-DL" indicate timers or cycles.
- The (red) arrow filled with white color crossing the thin horizontal line in the lowest part of the illustration indicates the access to the xC unit of the netX processor. This low level access is organized as FIFO. Communication blocks are transferred towards the PROFIBUS.

The dual-port memory is used for exchange of packets, I/O data, status and control information. Configuration and IO data will be transferred using the dual-port memory.

The user application only accesses the task located in the highest layer namely the AP task which constitutes the application interface of the PROFIBUS MPI stack.

The MPI task and DL task represent the core of the PROFIBUS MPI Stack.

The AP task represents the interface between the PROFIBUS MPI protocol stack and the dual-port memory. It is responsible for:

- Control of LEDs
- Diagnosis
- Packet routing

3.3 Functionality of the MPI and MPI-AP Task

The Profibus MPI provides the following functionality:

- Acyclic Data Transfer
- Configuration via Database
- Status information

3.3.1 Acyclic Data Transfer

The MPI Task can be used for transfer data from and to a MPI Server (for example Siemens S7). The main packets for acyclic data transfer are:

Section	Packet Name
4.2.3	PROFIBUS_MPI_CMD_RW_DB_REQ/CNF - Read/Write Data Block (Daten Baustein)
4.2.4	PROFIBUS_MPI_CMD_RW_ME_REQ/CNF - Read/Write Merker
4.2.5	PROFIBUS_MPI_CMD_RW_IO_REQ/CNF – Read/Write Input/Output
4.2.6	PROFIBUS_MPI_CMD_RW_CN_REQ/CNF - Read/Write Counter
4.2.7	PROFIBUS_MPI_CMD_RW_TI_REQ/CNF - Read/Write Timer
4.2.8	PROFIBUS_MPI_CMD_R_OP_REQ/CNF – Read Operation State
4.2.9	PROFIBUS_MPI_CMD_RW_TR_REQ/CNF – Send Transparent

Table 5: Packets for Acyclic Data Transfer

3.3.1.1 Description of Process during Acyclic Data Transfer

If the MPI task is configured properly, the MPI task can exchange data with other MPI server stations. The establishing of connection is done automatically if the first request is send. The Connection is persistent until a disconnect packet is send or an initialization is done. Normally there is no necessary to break the connection.

3.3.2 Configuration via Database

The MPI-AP Task will open and handle databases generated by SyCon. This Database can be used to configure the MPI Stack automatically at startup.

3.3.3 Status information

The MPI-AP Task will show the current Status via DPM Flags (see DPM Manual).

Communication State

The communication state field contains information about the current device status.

Value	Definition	Meaning
0x00000000	UNKNOWN	The Stack is in a unknown state
0x00000001	OFFLINE	No Configuration is available
0x00000002	STOP	Successfully configured - no bus communication
0x00000003	IDLE	Token exchange is active - Device is the only Master on the Bus
0x00000004	OPERATE	Token exchange is active

Table 6: Communication State

3.4 Functionality of the DL Task (Layer 2)

To get the handle of the process queue of the DL-Task the Macro `TLR_QUE_IDENTIFY()` has to be used in conjunction with the following ASCII-queue name

ASCII queue name	Description
"PB_DL_QUE"	Name of the DL-Task process queue

Table 7: DL-Task process queue

The returned handle has to be used as value `ulDest` in all initiator packets the AP-Task intends to send to the DL-Task. This handle is the same handle that has to be used in conjunction with the macros like `TLR_QUE_SENDBUFFER_FIFO/LIFO()` for sending a packet to the DL-Task.

A part of the functionality of the Profibus MPI Protocol Stack is situated within layer 2 of the OSI/ISO layer model of communication in networks. The DL task (DL = Data Link Layer) provides this functionality as an infrastructure to be used by the higher-level functions on layer 7 and higher.

This functionality concentrates mainly on the following 3 topics

- Data transfer services
- Management of Services Access Points (SAPs)
- Management of the system variables of layer 2

The following data transfer services are provided:

- SDA Service (Send data with acknowledge)
- SDN Service (Send data with no acknowledge)
- SRD Service (Send and request data with reply)
- MSRD-Service (Multi-cast send and request data with reply, currently not supported)

The following services are provided for the management of Service Access Points (SAPs)

- SAP Activate
- RSAP Activate
- SAP Deactivate

Furthermore, there are functions providing access to system variables of layer 2

- Set Value
- Set Value Bus Parameter Set

3.4.1 Data Transfer Services

All these services are requested by one participant (denominated as the “initiator”) and are performed by another participant (denominated as the “responder”). Both master and slaves may act as responders, but only masters are authorized to act as an initiator of a data transfer.

All these services connect a local user with a remote user.

In this context, the following definitions apply:

Local user:

A local user is a user of layer 2 functionality (FDL) at a master station.

Remote user:

A remote user is the FDL user at the remote station (slave or master).

Link service data unit (L_sdu)

The data area to be transferred is denominated as link service data unit or shortly as L_sdu.

3.4.1.1 SDA Service

The SDA-Service (SDA = Send data with acknowledge) provides acknowledged connectionless data transfer with immediate response. By the SDA service, a local user gets the possibility to send data (i.e. the L_sdu) to one single remote user (i.e. remote station).

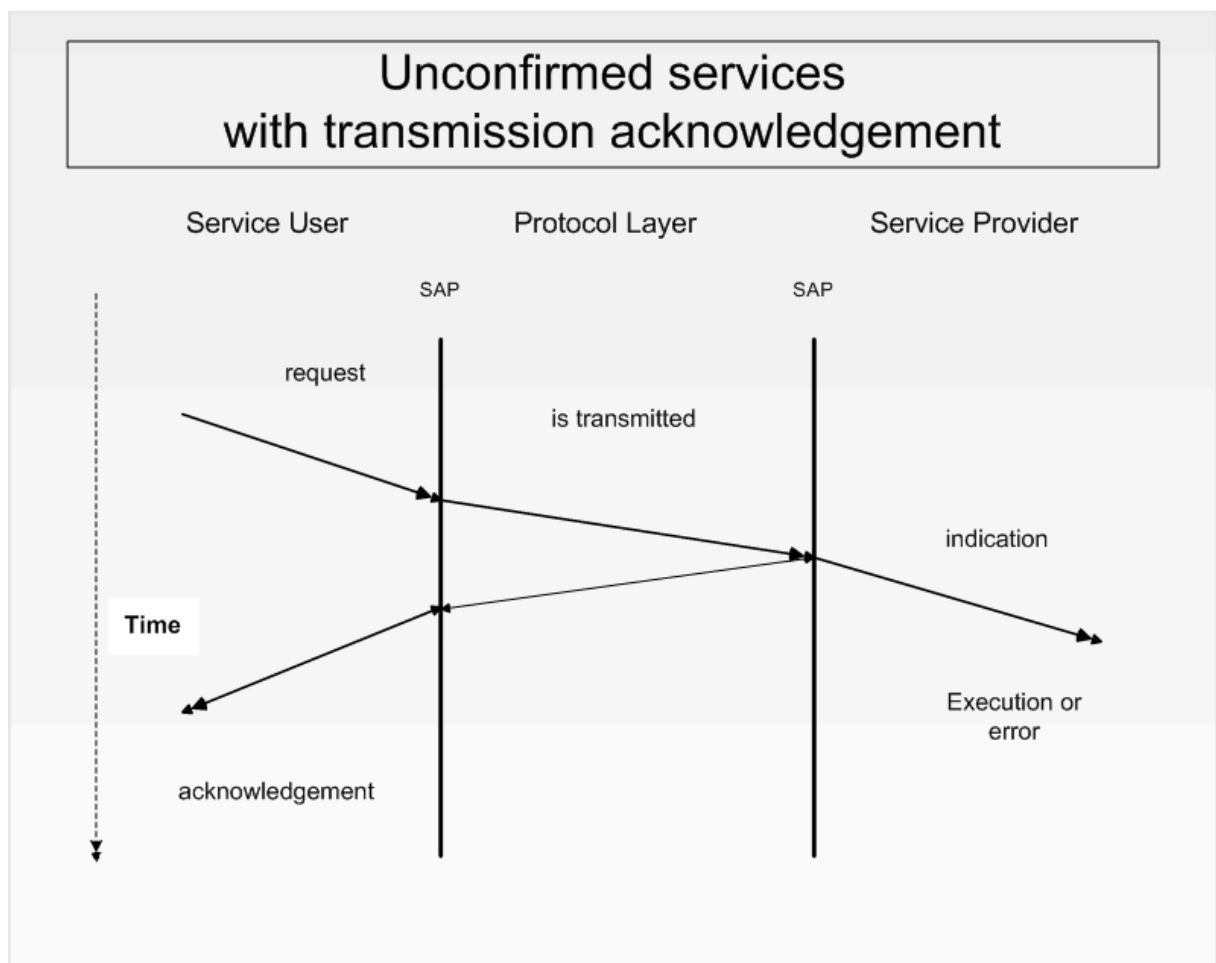


Figure 3: SDA Service with Acknowledgement

The end of the data transfer is acknowledged but not confirmed (which would mean waiting for finishing processing of the data instead of simply receiving the data). This means:

The local user is informed about successful reception or the non-reception of the data by an explicit acknowledgement. However, the user is not informed about successful execution of the request.

In case of an error during data transmission the data transfer will be repeated as often as necessary.



Note: The SDA service is usually not applied in Profibus DP systems (but important for Profibus FMS systems). Nevertheless, it is implemented due to compatibility reasons. This service is mainly used for master-to-master-communication.

The following packets provide the SDA functionality in the Profibus DP master firmware:

- PROFIBUS_DL_CMD_DATA_ACK_REQ/CNF - Send SDA Service Request/Confirmation
- PROFIBUS_DL_CMD_DATA_ACK_IND - Receive SDA Service Indication

3.4.1.2 SDN Service

The SDN -Service (SDN = Send data with no acknowledge) provides unacknowledged connectionless data transfer. By the SDN service, a local user gets the possibility to send data (i.e. the L_sdu) to

- one single remote user (i.e. remote station).
- many remote users (remote stations/ multi-cast communication)
- all remote users/remote stations within the Profibus DP network synchronously

The confirmation of an SDN service only locally confirms that the service has been received for further transmission at the bus, but it neither confirms successful transmission to the receiver nor successful processing there.

In case of an error during data transmission the data transfer will be repeated as often as necessary.

If the remote station receives the link service data unit in an error-free condition, it will be processed (i.e. handled over to the remote user). The local user, however, is not able to obtain information whether processing has taken place as originally intended.

The following packets provide the SDN functionality in the Profibus DP master firmware:

- PROFIBUS_DL_CMD_DATA_REQ/CNF - Send SDN Service Request/Confirmation
- PROFIBUS_DL_CMD_DATA_IND - Receive SDN Service Indication

3.4.1.3 SRD Service

The SRD-Service (SRD = Send and request data with reply) is designed to provide bidirectional connectionless data exchanged. By the SRD service, a local user gets the possibility to send and synchronously request data (i.e. the L_sdu) to one single remote user (i.e. remote station) at the same time. The requested data must have been stored for fetching there earlier. For the local user, it is also possible to request data without sending data in parallel (pure request command).

In case of successful execution, the local user will receive the requested data. Otherwise, an indication will be sent if the requested data are not available. Both cases confirm the correct reception of the transmitted data at the remote station. However, if the transmitted data have not been received properly, a negative confirmation will be sent.

In case of an error during data transmission, both the data transfer and the request will be repeated as often as necessary.

The following packets provide the SRD functionality in the Profibus DP master firmware:

- PROFIBUS_DL_CMD_DATA_REPLY_REQ/CNF - Send SRD Service Request/Confirmation
- PROFIBUS_DL_CMD_DATA_REPLY_IND - Receive SRD Service Indication

3.4.1.4 MSRD Service

An MSRD service request is simply an SRD request where the slave answers with a multi-cast telegram instead of an ordinary telegram for a one-to-one communication relation. This service is not supported by the current firmware.

3.4.2 Management of Services Access Points

The following services are provided concerning the administration of service access points (SAPs):

- Activation of SAP
- Activation of SAP for responder (RSAP)
- Deactivation of SAP

3.4.2.1 Services Access Points

A service access point is a means for managing communication between different communication layers correctly; it is associated with and identified by a non-negative integer numeric value. Service access points are used to characterize different kinds of services within the network. The available numeric values range from 0 to 63. Also the value 64 is permitted, but it represents the NIL SAP.



Note: NIL SAP in this context means that the telegram does not contain any SAP information. Such telegrams are used by Profibus DP for performing cyclic communication.

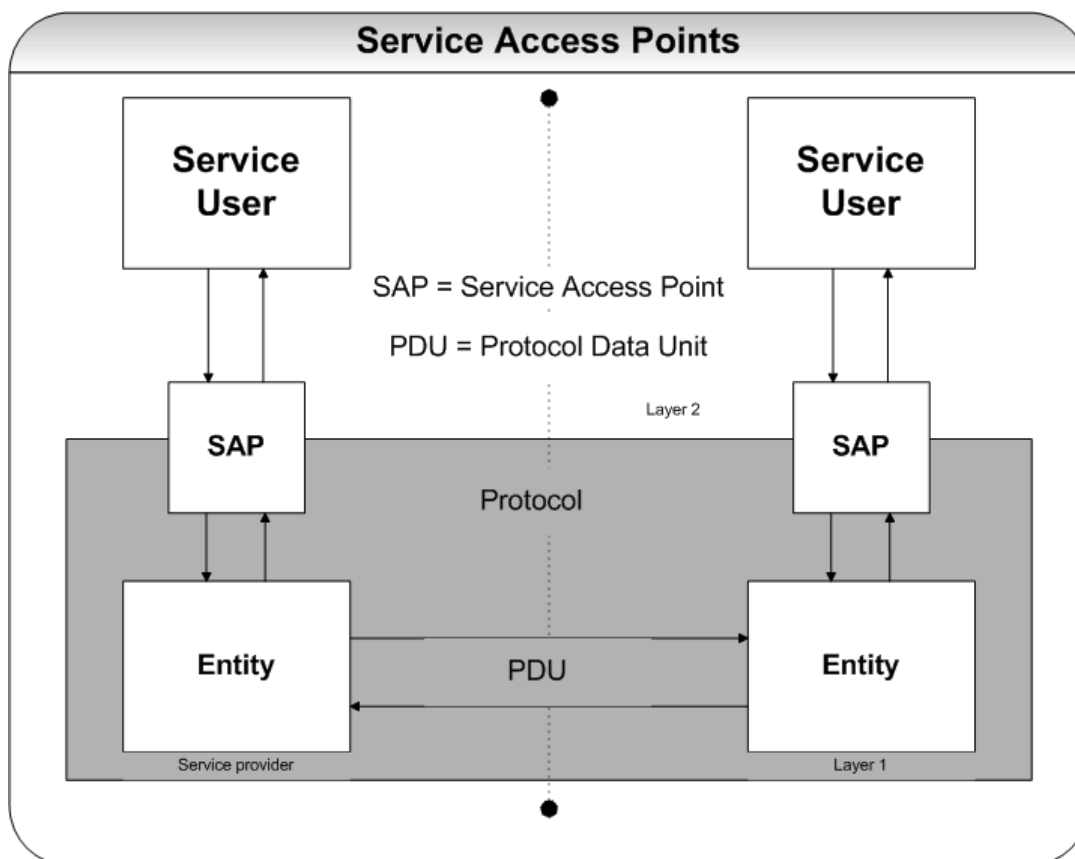


Figure 4: Service Access Points

However, some values might be predefined by other services such as the OSI layer 7 services of Profibus DP. Especially the values 50 (0x32), 51 (0x33), 54 (0x36) and 62 (0x3E) are usually applied for special purposes at the Profibus DP master.

You can also imagine SAP's as the entry and exit points of a layer in a multi-layered communication model. You may separate among the services for which to define SAP's between user services and management services, so you can introduce a separation between user SAP's (which might be activated by this packet) and previously activated management SAP's.

Along with the SAP's come automatic checking mechanism for rule conformity of the messages and the accompanying transmission-related informations stored in the messages. Only if all checks have been passed successfully, the data transfer for which an SAP has been defined will take place, otherwise a message will be issued and no data will be transferred.

Generally, the SAP's should be configured during the start-up phase of the Profibus network.

The philosophy of the SAP concept might look unnecessarily sophisticated, but its real aims justifying this amount of sophistication are:

- Precise Control of messages and transmission channels
- Avoidance of erroneous connections

3.4.2.2 SAP Activate

The following packet provides the SAP activate functionality in the Profibus DP master firmware:

- PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ/CNF - Activate an SAP for Request

This packet is applicable in the following situation:

- It is intended to activate a service access point for an SDA or SDN service
- It is intended to activate a service access point for an SRD or MSRD service with no responder functionality (i.e. the role in service is the one of an initiator).

3.4.2.3 RSAP Activate

The following packet provides the RSAP activate functionality in the Profibus DP master firmware:

- PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ/CNF - Activate an SAP for Responder Functionality(i.e. the role in service is responder or both initiator and responder).

This packet provides the possibility to establish a service access point for an SRD or MSRD service with responder functionality.

3.4.2.4 SAP Deactivate

This service allows the deactivation of a formerly defined service access point, which is not needed any longer. The following packet provides the SAP deactivate functionality in the Profibus DP master firmware:

- PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ/CNF - Deactivate an SAP for Request

3.4.3 Management of Layer 2 System Variables and Tasks

There are two services available for setting system parameters and variables within the PROFIBUS DL Task. One service sets some specific values of the DL Layer. See section 4.3.15 “PROFIBUS_DL_CMD_SET_VALUE_REQ/CNF - Set Value Service”. The other available service loads an entire bus parameter set as a whole. For more information on this service, have a look at section 4.3.13 “PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ/CNF - Deactivate an SAP for Request”.

There are also some functions dealing with administrative topics such as starting and stopping the DL functionality and stopping the whole DL task:

- PROFIBUS_DL_CMD_END_PROCESS_REQ/CNF – Finish the DL Task
- PROFIBUS_DL_CMD_START_DLE_REQ/CNF – Starts the DL-Layer
- PROFIBUS_DL_CMD_STOP_DLE_REQ/CNF – Stop DL Layer

4 The Application Interface

This chapter defines the application user interface of the PROFIBUS MPI Stack.

The application itself has to be developed as a task. The Application-Task is named AP-Task in the following sections and chapters.

The AP-Task's process queue is keeping track of all its incoming packets and has to be addressed from the user application. It provides the communication channel for the underlying PROFIBUS MPI Stack.

4.1 MPI AP Task

Overview over Packets of the PROFIBUS MPI-AP Task			
No. of section	Packet	Command code (REQ/CNF or IND/RES)	Page
4.1.1	PROFIBUS_MPI_AP_CMD_SET_CONFIG_REQ/CNF – Set Configuration Parameter	0x4200	30

Table 8: Overview over Packets of the PROFIBUS MPI-AP Task

4.1.1 PROFIBUS_MPI_AP_CMD_SET_CONFIG_REQ/CNF – Set Configuration Parameter

The packet below is used to provide configuration information to the PROFIBUS-MPI stack. It holds values for the system flags, the watchdog time and the network parameters.

Currently the system flags variable `ulSystemFlags` can have the following values:

Value	Meaning
PROFIBUS_MPI_AP_CONFIG_SYS_FLG_START_APPLICATION	0x00000001

Table 9: Variable `ulSystemFlags`

The parameter `ulWdgTime` contains the watchdog time. This parameter may be set to values in the range between 20 and 65535. A value of 0 is also allowed, it indicates that the watchdog timer is turned off.

The structure of the bus parameters of the network is described in section 2.1.1 “Detailed Description of Bus Parameters”.

The following applies:

- Configuration parameters will be stored internally.
- In case of any error, no data will be stored at all.
- A channel init is required to activate the parameterized data.
- This request will be denied, if the configuration lock flag is set

Packet Structure Reference

```
#define PROFIBUS_MPI_AP_CONFIG_SYS_FLG_START_MASK      0x00000001
#define PROFIBUS_MPI_AP_CONFIG_SYS_FLG_START_AUTO      0x00000000
#define PROFIBUS_MPI_AP_CONFIG_SYS_FLG_START_APPLICATION 0x00000001

#define PROFIBUS_MPI_AP_CONFIG_WATCHDOG_OFF            0
#define PROFIBUS_MPI_AP_CONFIG_WATCHDOG_MIN            20
#define PROFIBUS_MPI_AP_CONFIG_WATCHDOG_MAX            65535

typedef struct PROFIBUS_MPI_AP_SET_CONFIG_DATA_Ttag
{
    TLR_UINT32          ulSystemFlag;
    TLR_UINT32          ulWdgTime;
    PROFIBUS_DL_BUS_PARAMETER_SET_T tBusParameter;
} PROFIBUS_MPI_AP_SET_CONFIG_DATA_T;

typedef struct PROFIBUS_MPI_AP_PACKET_SET_CONFIG_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_MPI_AP_SET_CONFIG_DATA_T tData;
} PROFIBUS_MPI_AP_PACKET_SET_CONFIG_REQ_T;

typedef struct PROFIBUS_MPI_AP_PACKET_SET_CONFIG_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_MPI_AP_PACKET_SET_CONFIG_CNF_T;
```

structure PROFIBUS_MPI_AP_PACKET_SET_CONFIG_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	>= 23	Packet Data Length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4200	PROFIBUS_MPI_CMD_INIT_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_INIT_REQ_T			
	ulSystemFlag	UINT32	0	System flags, set to 0 (denoting auto start)
	ulWdgTime	UINT32	0, 20-65535	Host Watchdog Time in ms (ignored if 0)
	tBusParameter	PROFIBUS_DL_BUS_PARAMETER_SET_T		Bus parameter structure, see chapter 2.1.1 Detailed Description of Bus Parameters

Table 10: structure PROFIBUS_MPI_AP_PACKET_SET_CONFIG_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_AP_PACKET_SET_CONFIG_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_MPI_AP_PACKET_SET_CONFIG_CNF_T;
```

structure PROFIBUS_MPI_AP_PACKET_SET_CONFIG_CNF_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	0	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4201	PROFIBUS_MPI_CMD_INIT_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch

Table 11: structure PROFIBUS_MPI_AP_PACKET_SET_CONFIG_CNF_T

4.2 MPI Task

In detail, the following functionality is provided by the MPI -Task:

Overview over Packets of the PROFIBUS MPI-Task			
No. of section	Packet	Command code (REQ/CNF or IND/RES)	Page
4.2.1	PROFIBUS_MPI_CMD_INIT_REQ/CNF - Initialization of PROFIBUS MPI Protocol Stack	0x4300/ 0x4301	34
4.2.2	PROFIBUS_MPI_CMD_SET_BUS_PARAM_REQ/CNF - Load the Bus Parameter Set	0x4316/ 0x4317	36
4.2.3	PROFIBUS_MPI_CMD_RW_DB_REQ/CNF - Read/Write Data Block	0x4306/ 0x4307	42
4.2.4	PROFIBUS_MPI_CMD_RW_ME_REQ/CNF - Read/Write Merker	0x430A/ 0x430B	46
4.2.5	PROFIBUS_MPI_CMD_RW_IO_REQ/CNF – Read/Write Input/Output	0x430C/ 0x430D	50
4.2.6	PROFIBUS_MPI_CMD_RW_CN_REQ/CNF - Read/Write Counter	0x430E/ 0x430F	54
4.2.7	PROFIBUS_MPI_CMD_RW_TI_REQ/CNF - Read/Write Timer	0x4310/ 0x4311	57
4.2.8	PROFIBUS_MPI_CMD_R_OP_REQ/CNF – Read Operation State	0x4308/ 0x4309	60
4.2.9	PROFIBUS_MPI_CMD_RW_TR_REQ/CNF – Send Transparent	0x4304/ 0x4305	63
4.2.10	PROFIBUS_MPI_CMD_DC_REQ/CNF - Disconnect	0x4312/ 0x4313	66
4.2.11	PROFIBUS_MPI_CMD_DC_ALL_REQ/CNF –	0x4314/ 0x4315	68
4.2.12	PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T - Close a specific Socket Request/Confirmation	0x431A/ 0x431B	70
4.2.13	PROFIBUS_MPI_CMD_MULTIPLE_READ_REQ/CNF - Multiple Read Request	0x431E/ 0x431F	72
4.2.14	PROFIBUS_MPI_CMD_MULTIPLE_WRITE_REQ/CNF - Multiple Write Request	0x4320/ 0x4321	78

Table 12: Overview over the Packets of the PROFIBUS MPI-Task

4.2.1 PROFIBUS_MPI_CMD_INIT_REQ/CNF - Initialization of PROFIBUS MPI Protocol Stack

This packet resets the protocol stack. All outstanding packets will be returned with a failure, all open connections will be closed and all parameter will be discarded. After this command, no read or write operation is possible.

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_INIT_REQ_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
} PROFIBUS_MPI_PACKET_INIT_REQ_T;
```

structure PROFIBUS_MPI_PACKET_INIT_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	0	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4300	PROFIBUS_MPI_CMD_INIT_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch

Table 13: Request structure PROFIBUS_MPI_PACKET_INIT_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_INIT_CNF_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
} PROFIBUS_MPI_PACKET_INIT_CNF_T;
```

structure PROFIBUS_MPI_PACKET_INIT_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	0	Packet Data Length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 PROFIBUS MPI-Error Codes on page 161.
	ulCmd	UINT32	0x4301	PROFIBUS_MPI_CMD_INIT_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch

Table 14: Confirmation structure PROFIBUS_MPI_PACKET_INIT_CNF_T

4.2.2 PROFIBUS_MPI_CMD_SET_BUS_PARAM_REQ/CNF - Load the Bus Parameter Set

Ensure that all active (master) stations have the same bus parameter settings, especially the baudrate (`bData_rate`), the Highest Station Address (`bHSA`) and the Target Rotation Time (`ulTTR`).

The standard values of the MPI parameters are these:

<code>usBus_Para_Len</code>	Length of bus parameter data	35
<code>bDL_Add</code>	Address of the PROFIBUS MPI Device	0-126
<code>bData_rate</code>	Data rate (Transmission speed of data)	187.5 kBaud
<code>usTSL</code>	Slot-time	415 Bit
<code>usMin_TSDR</code>	Min. station delay responder time	60 tBit
<code>usMax_TSDR</code>	Max. station delay responder time	400 tBit
<code>bTQUI</code>	Quiet time	1 tBit
<code>bSET</code>	Setup time	1 tBit
<code>ulTTR</code>	Target rotation time	10000 tBit
<code>bG</code>	GAP update factor	20
<code>bHSA</code>	Highest station address	31

Table 15: Standard values of the MPI parameters

The `bData_rate` parameter is coded as listed in *Table 4: Supported Baud Rates* of this document.

If autobaud is configured (`bData_rate` = 15, `PROFIBUS_DL_DATA_RATE_AUTO`), the stack will listen to the PROFIBUS network and automatically retrieve the baudrate if it is able to receive broadcast telegrams from a connected PLC.

In order to avoid initialization problems, we recommend the following:

- Ensure that every participant on the network uses the same parameters.
- Ensure that the PLC connected to your Hilscher MPI device provides support for repeatedly sending broadcast messages and that this support has also been activated at your PLC.

A detailed description of all bus parameters is given in section 2.1.1 “*Detailed Description of Bus Parameters*” of this document.

A data structure containing the currently configured bus and master parameter set is returned along with the confirmation packet. If autobaud detection is activated (`bData_rate` = 15) the SPS must send cyclic a broadcast message with the correct bus parameter. It is possible to abort the command by sending an `PROFIBUS_MPI_CMD_INIT_REQ`.

Packet Structure Reference

```

typedef struct PROFIBUS_DL_BUS_PARAMETER_SET_Ttag {
    TLR_UINT16 usBus_Para_Len; /* Contains the length of the Bus_Para inclusive the
field Bus_Para_Len itself */
    TLR_UINT8  bDL_Add; /* Contains the own address of the PROFIBUS Device */
    TLR_UINT8  bData_rate; /* Contains the Transmission speed */
    TLR_UINT16 usTSL; /* slot-time */
    TLR_UINT16 usMin_TSDR; /* min. station delay responder */
    TLR_UINT16 usMax_TSDR; /* max. station delay responder */
    TLR_UINT8  bTQUI; /* quite time */
    TLR_UINT8  bTSET; /* setup time */
    TLR_UINT32 ultTR; /* target rotation time */
    TLR_UINT8  bG; /* Gap update factor */
    TLR_UINT8  bHSA; /* Highest station address */
    TLR_UINT8  bMax_Retry_Limit; /* retries if error occurs */
    TLR_UINT8  bBp_Flag;
    TLR_UINT16 usMin_Slave_Interval; /* Minimum Slave Interval Time */
    TLR_UINT16 usPoll_Timeout; /* Class2 Poll timeout */
    TLR_UINT16 usData_Control_Time; /* Data Control Time */
    /*TLR_UINT8  bAlarm_Max; */ /* Maximum Alarms */
    TLR_UINT8  bMasterSetting; /* 0 == big Endian, 1 == little Endian (swapping
should be done)*/
    TLR_UINT8  bMax_User_Global_Control; /* Maximum allowed parallel active USER
Global Control Commands */
    TLR_UINT8  abReserved[4]; /* 4 reserved Octets */
    TLR_UINT16 usMaster_User_Data_Len; /* Contains the length of the USER Master data
*/
    TLR_UINT8  abMaster_Class2_Name[32]; /* Name of the Master */
    TLR_UINT8  abMaster_User_Data[32]; /* USER specific Parameter data */
    TLR_UINT32 ultTCL; /* Isochronous cycle time */
    TLR_UINT8  bMax_TSH; /* Maximum Shift Time */
}PROFIBUS_DL_BUS_PARAMETER_SET_T;

typedef struct PROFIBUS_MPI_SET_BUS_PARAMETER_DATA_Ttag
{
    PROFIBUS_DL_BUS_PARAMETER_SET_T tBusParameter;
}PROFIBUS_MPI_SET_BUS_PARAMETER_DATA_T;

typedef struct PROFIBUS_MPI_PACKET_SET_BUS_PARAMETER_REQ_Ttag
{
    TLR_PACKET_HEADER_T          tHead;
    PROFIBUS_MPI_SET_BUS_PARAMETER_DATA_T tData;
} PROFIBUS_MPI_PACKET_SET_BUS_PARAMETER_REQ_T;

#define SET_BUS_PARAMETER_REQ_MIN_LEN  19

```

Packet Description

structure PROFIBUS_MPI_PACKET_SET_BUS_PARAMETER_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulDL0Id	Destination Queue Reference
	ulSrcId	UINT32	ulAPMS0Id	Source Queue Reference
	ulLen	UINT32	>=19	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4316	PROFIBUS_MPI_CMD_SET_BUS_PARAM_REQ - Command
	ulExt	UINT32	0	Extension, unchanged
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_MPI_SET_BUS_PARAMETER_DATA_T			
	usBus_Para_Len	UINT16	>= 19	Contains the length of the Bus_Para including the field usBus_Para_Len itself. The minimum length is 19. See section 2.1.1.1 " <i>usBus_Para_Len</i> "
	bDL_Add	UINT8	0-126	Contains the own address of the PROFIBUS Device. See section 2.1.1.2 " <i>bDL_Add</i> "
	bData_rate	UINT8	0..11,15 Default: 3 (187,5 kBits/s)	Contains the Transmission speed coded as specified in <i>Table 4: Supported Baud Rates</i> .
	usTSL	UINT16	37-16383 Default: 415	Slot-time See section 2.1.1.4 " <i>usTSL (Slot Time)</i> "
	usMin_TSDR	UINT16	1-1023 Default:60	Min. station delay responder See section 2.1.1.5 " <i>usMin_TSDR</i> "
	usMax_TSDR	UINT16	1-1023 Default:400	Max. station delay responder See section 2.1.1.6 " <i>usMax_TSDR</i> "
	bTQUI	UINT8	0-127 Default:1	Quiet time See section 2.1.1.7 " <i>bTQUI (Quiet Time)</i> "
	bTSET	UINT8	1-255 Default:1	Setup time See section 2.1.1.8 " <i>bSET (Setup Time)</i> "
	ulTTR	UINT32	>= 255 Default:10000	Target rotation time See section 2.1.1.9 " <i>ulTTR (Target Rotation Time)</i> "
	bG	UINT8	1-255 Default:20	GAP update factor See section 2.1.1.10 " <i>bG (GAP Actualization Factor)</i> "
	bHSA	UINT8	1-126	Highest station address

			Default:31	See section 2.1.1.11“bHSA (Highest Station Address)”
	bMax_Retry_Limit	UINT8	1-15	Allowed number of retries if error occurs
	bBp_Flag	UINT8	0-255	Bp flag
	usMin_Slave_Interval	UINT16	0-65535	Minimum Slave Interval Time (this Parameter is not needed for using MPI)
	usPoll_Timeout	UINT16	0-65535	Class2 Poll timeout (this Parameter is not needed for using MPI)
	usData_Control_Time	UINT16	1-65535	Data Control Time (this Parameter is not needed for using MPI)
	bMasterSetting	UINT8	0,1	0 == BIG_ENDIAN, 1 == LITTLE_ENDIAN (swapping should be done) (this Parameter is not needed for using MPI)
	bMax_User_Global_Control	UINT8	1-255	Maximum allowed parallel active USER Global Control Commands (this Parameter is not needed for using MPI)
	abReserved	UINT8[4]		4 reserved Octets (this Parameter is not needed for using MPI)
	usMaster_User_Data_Len	UINT16	0-32	Contains the length of the USER Master data (this Parameter is not needed for using MPI)
	abMaster_Class2_Name[32]	UINT8[32]		Name of the Master (this Parameter is not needed for using MPI)
	abMaster_User_Data[32];	UINT8[32]	0-32	USER specific Parameter data (this Parameter is not needed for using MPI)
	ulTCL	UINT32		Isochronous cycle time (this Parameter is not needed for using MPI)
	bMax_TSH	UINT8		Maximum Shift Time (this Parameter is not needed for using MPI)

Table 16: Request structure PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ - Load the Bus Parameter Set

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_SET_BUS_PARAMETER_CNF_Ttag
{
    TLR_PACKET_HEADER_T          tHead;
    PROFIBUS_MPI_SET_BUS_PARAMETER_DATA_T tData;
} PROFIBUS_MPI_PACKET_SET_BUS_PARAMETER_CNF_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_SET_BUS_PARAMETER_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	>=19	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section "PROFIBUS MPI-Error Codes"
	ulCmd	UINT32	0x4317	PROFIBUS_MPI_CMD_SET_BUS_PARAM_CNF - Command
	ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_MPI_SET_BUS_PARAMETER_DATA_T			
	usBus_Para_Len	UINT16	>=19	Contains the length of the Bus_Para including the field usBus_Para_Len itself
	bDL_Add	UINT8	0-126	Contains the own address of the PROFIBUS Device
	bData_rate	UINT8	0-11	Contains the Transmission speed
	usTSL	UINT16	37-16383	Slot-time
	usMin_TSDR	UINT16	1-1023	Min. station delay responder
	usMax_TSDR	UINT16	1-1023	Max. station delay responder
	bTQUI	UINT8	0-127	Quiet time
	bTSET	UINT8	1-255	Setup time
	ulTTR	UINT32	>= 255	Target rotation time
	bG	UINT8	1-255	GAP update factor
	bHSA	UINT8	1-126	Highest station address
	bMax_Retry_Limit	UINT8	0	Retries if error occurs
	bBp_Flag	UINT8	0	Bp flag
	usMin_Slave_Interval	UINT16	0	Minimum Slave Interval Time

usPoll_Timeout	UINT16	0	Class2 Poll timeout
usData_Control_Time	UINT16	0	Data Control Time
bMasterSetting	UINT8	0,1	0 == BIG_ENDIAN, 1 == LITTLE_ENDIAN
bMax_User_Global_Control	UINT8	0	Maximum allowed parallel active USER Global Control Commands
abReserved	UINT8[4]	0	4 reserved Octets
usMaster_User_Data_Len	UINT16	0	Contains the length of the USER Master data
abMaster_Class2_Name[32]	UINT8[32]	0	Name of the Master
abMaster_User_Data[32];	UINT8[32]	0	USER specific Parameter data
ulTCL	UINT32	0	Isochronous cycle time
bMax_TSH	UINT8	0	Maximum Shift Time

Table 17: Confirmation structure *PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_CNF* – Confirmation of Loading the Bus Parameter Set

4.2.3 PROFIBUS_MPI_CMD_RW_DB_REQ/CNF - Read/Write Data Block (Daten Baustein)

This command allows reading or writing data registers within S7[®] components with integrated support for the MPI protocol. If there is no MPI connection established to the remote station and the command is called, the device automatically establishes the connection in the background before accessing the registers.

Packet Structure Reference

```
#define PROFIBUS_MPI_RW_DB_MAX_DATA_SIZE      222

typedef struct PROFIBUS_MPI_DATA_MPI_RW_DB_Ttag {
    TLR_UINT32  ulRemAddr;
    TLR_UINT32  ulDataType;
    TLR_UINT32  ulDBNumber;
    TLR_UINT32  ulOffset;
    TLR_UINT32  ulDataCnt;
    TLR_UINT32  ulBitOffset;
    TLR_UINT32  ulFunction;
    TLR_UINT8   abData[PROFIBUS_MPI_RW_DB_MAX_DATA_SIZE];
}PROFIBUS_MPI_DATA_MPI_RW_DB_T;

typedef struct PROFIBUS_MPI_PACKET_RW_DB_REQ_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_RW_DB_T tData;
}PROFIBUS_MPI_PACKET_RW_DB_REQ_T;
```

structure PROFIBUS_MPI_PACKET_RW_DB_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4306	PROFIBUS_MPI_CMD_RW_DB_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_DB_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulDataType	UINT32	0,1	Data type 0: Byte 1: Bool
	ulDBNumber	UINT32	0-65534	Data Block Number
	ulOffset	UINT32	0-255	Offset
	ulDataCnt	UINT32	1-222 1-212	Number of bytes to be read or written 222 bytes to read 212 bytes to write In case of data type bool only 1 is allowed
	ulBitOffset	UINT32	0-7	Bit offset, use only in case of data type bool
	ulFunction	UINT32	1,2	Function code 1: READ 2: WRITE
	abData[]	UINT8[]		Field containing user data in case of write. For writing data the size is limited to 212 byte. This field is only present in case of writing data.

Table 18: Request structure PROFIBUS_MPI_PACKET_RW_DB_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_RW_DB_CNF_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_RW_DB_T tData;
}PROFIBUS_MPI_PACKET_RW_DB_CNF_T;
```

structure PROFIBUS_MPI_PACKET_RW_DB_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4307	PROFIBUS_MPI_CMD_RW_DB_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_DB_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulDataType	UINT32	0,1	Data type 0: Byte 1: Bool
	ulDBNumber	UINT32	0-65534	Data Block Number
	ulOffset	UINT32	0-255	Offset
	ulDataCnt	UINT32	1-222 1-212	Number of bytes to be read or written 222 bytes to read 212 bytes to write
	ulBitOffset	UINT32	0-7	Bit offset
	ulFunction	UINT32	1,2	Function code 1: READ 2: WRITE
	abData[]	UINT8[]		Field containing user data in case of read. For reading data the size is limited to 222 byte. This field is only present in case of reading.

Table 19: Confirmation structure PROFIBUS_MPI_PACKET_RW_DB_CNF_T

4.2.4 PROFIBUS_MPI_CMD_RW_ME_REQ/CNF - Read/Write Merker

This command allows reading or writing data registers within S7[®] components with integrated support for the MPI protocol. If there is no MPI connection established to the remote station and the command is called, the device automatically establishes the connection in the background before accessing the registers.

Packet Structure Reference

```
#define PROFIBUS_MPI_RW_ME_MAX_DATA_SIZE      222

typedef struct PROFIBUS_MPI_DATA_MPI_RW_ME_Ttag {
    TLR_UINT32  ulRemAddr;
    TLR_UINT32  ulDataType;
    TLR_UINT32  ulMerkerAddr;
    TLR_UINT32  ulReserved2;
    TLR_UINT32  ulDataCnt;
    TLR_UINT32  ulBitOffset;
    TLR_UINT32  ulFunction;
    TLR_UINT8   abData[PROFIBUS_MPI_RW_ME_MAX_DATA_SIZE];
}PROFIBUS_MPI_DATA_MPI_RW_ME_T;

typedef struct PROFIBUS_MPI_PACKET_RW_ME_REQ_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_RW_ME_T tData;
}PROFIBUS_MPI_PACKET_RW_ME_REQ_T;
```

structure PROFIBUS_MPI_PACKET_RW_ME_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x430A	PROFIBUS_MPI_CMD_RW_ME_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_ME_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulDataType	UINT32	0,1	Data type 0: Byte 1: Bool
	ulMerkerAddr	UINT32	0-65534	Data address/ Memory Address
	ulReserved2	UINT32	0	Data index, unused
	ulDataCnt	UINT32	1-222 1-212	Number of bytes to be read or written 222 bytes to read 212 bytes to write In case of data type bool only 1 is allowed
	ulBitOffset	UINT32	0-7	Bit offset, use only in case of data type bool
	ulFunction	UINT32	1-2	Function code 1: READ 2: WRITE
	abData[]	UINT8[]		Field containing user data in case of write. For writing data the size is limited to 212 byte. This field is only present in case of writing data.

Table 20: Request structure PROFIBUS_MPI_PACKET_RW_ME_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_RW_ME_CNF_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_RW_ME_T tData;
} PROFIBUS_MPI_PACKET_RW_ME_CNF_T;
```


structure PROFIBUS_MPI_PACKET_RW_ME_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x430B	PROFIBUS_MPI_CMD_RW_ME_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_ME_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulDataType	UINT32	0,1	Data type 0: Byte 1: Bit
	ulMerkerAddr	UINT32	0-65534	Data address/ Memory Address
	ulReserved2	UINT32	0	Data index, unused
	ulDataCnt	UINT32	1-222 1-212	Number of bytes to be read or written 222 bytes to read 212 bytes to write
	ulBitOffset	UINT32	0-7	Bit offset
	ulFunction	UINT32	1-2	Function code 1: READ 2: WRITE
	abData[]	UINT8[]		Field containing user data in case of read. For reading data the size is limited to 222 byte. This field is only present in case of reading.

Table 21: Confirmation structure PROFIBUS_MPI_PACKET_RW_ME_CNF_T

4.2.5 PROFIBUS_MPI_CMD_RW_IO_REQ/CNF – Read/Write Input/Output

This command allows reading or writing data registers within S7[®] components with integrated support for the MPI protocol. If there is no MPI connection established to the remote station and the command is called, the device automatically establishes the connection in the background before accessing the registers.

Packet Structure Reference

```
#define PROFIBUS_MPI_RW_IO_MAX_DATA_SIZE      222

typedef struct PROFIBUS_MPI_DATA_MPI_RW_IO_Ttag
{
    TLR_UINT32    ulRemAddr;
    TLR_UINT32    ulIOArea;
    TLR_UINT32    ulIOAddr;
    TLR_UINT32    ulDataType;
    TLR_UINT32    ulDataCnt;
    TLR_UINT32    ulBitOffset;
    TLR_UINT32    ulFunction;
    TLR_UINT8     abData[PROFIBUS_MPI_RW_IO_MAX_DATA_SIZE];
}PROFIBUS_MPI_DATA_MPI_RW_IO_T;

typedef struct PROFIBUS_MPI_PACKET_RW_IO_REQ_Ttag
{
    TLR_PACKET_HEADER_T    tHead;
    PROFIBUS_MPI_DATA_MPI_RW_IO_T tData;
}PROFIBUS_MPI_PACKET_RW_IO_REQ_T;
```

structure PROFIBUS_MPI_PACKET_RW_IO_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x430C	PROFIBUS_MPI_CMD_RW_IO_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_IO_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulIOArea	UINT32	0-1	Data Area 0: input area 1: output area
	ulIOAddr	UINT32	0-65534	Data Address: IO Address
	ulDataType	UINT32	0,1	Data type 0: Byte 1: Bit
	ulDataCnt	UINT32	1-222 1-212	Number of bytes to be read or written 222 bytes to read 212 bytes to write In case of data type bool only 1 is allowed
	ulBitOffset	UINT32	0-7	Bit offset, use only in case of data type bool
	ulFunction	UINT32	1-2	Function code 1: READ 2: WRITE
	abData[]	UINT8[]		Field containing user data in case of write. For writing data the size is limited to 212 byte. This field is only present in case of writing data.

Table 22: Request structure PROFIBUS_MPI_PACKET_RW_IO_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_RW_IO_CNF_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_RW_IO_T tData;
}PROFIBUS_MPI_PACKET_RW_IO_CNF_T;
```

structure PROFIBUS_MPI_PACKET_RW_IO_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x430D	PROFIBUS_MPI_CMD_RW_IO_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_IO_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulIOArea	UINT32	0-1	Data Area 0: input area 1: output area
	ulIOAddr	UINT32	0-65534	Data Address: IO Address
	ulDataType	UINT32	0,1	Data type 0: Byte 1: Bit
	ulDataCnt	UINT32	1-222 1-212	Number of bytes to be read or written <i>222 Byte USR Data to read</i> <i>212 Byte USR Data to write</i>
	ulBitOffset	UINT32	0-7	Bit offset
	ulFunction	UINT32	1-2	Function code 1: READ 2: WRITE
	abData[]	UINT8[]		Field containing user data in case of read. For reading data the size is limited to 222 byte. This field is only present in case of reading.

Table 23: Confirmation structure PROFIBUS_MPI_PACKET_RW_IO_CNF_T

4.2.6 PROFIBUS_MPI_CMD_RW_CN_REQ/CNF - Read/Write Counter

This command allows reading or writing data registers within S7[®] components with integrated support for the MPI protocol. If there is no MPI connection established to the remote station and the command is called, the device automatically establishes the connection in the background before accessing the registers.

Packet Structure Reference

```
#define PROFIBUS_MPI_RW_CN_MAX_DATA_SIZE      111

typedef struct PROFIBUS_MPI_DATA_MPI_RW_CN_Ttag
{
    TLR_UINT32    ulRemAddr;
    TLR_UINT32    ulReserved1;
    TLR_UINT32    ulCounterAddr;
    TLR_UINT32    ulReserved2;
    TLR_UINT32    ulDataCnt;
    TLR_UINT32    ulReserved3;
    TLR_UINT32    ulFunction;
    TLR_UINT16    ausData[PROFIBUS_MPI_RW_CN_MAX_DATA_SIZE];
} PROFIBUS_MPI_DATA_MPI_RW_CN_T;

typedef struct PROFIBUS_MPI_PACKET_RW_CN_REQ_Ttag
{
    TLR_PACKET_HEADER_T    tHead;
    PROFIBUS_MPI_DATA_MPI_RW_CN_T    tData;
} PROFIBUS_MPI_PACKET_RW_CN_REQ_T;
```

structure PROFIBUS_MPI_PACKET_RW_CN_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + 2 * ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x430E	PROFIBUS_MPI_CMD_RW_CN_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_CN_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	Data area, unused
	ulCounterAddr	UINT32	0-65534	Counter address
	ulReserved2	UINT32	0	Data index, unused
	ulDataCnt	UINT32		Number of counters to be read or written
			1-111	Number of words of counter data which are read
			1-106	Number of words of counter data which were written
	ulReserved3	UINT32	0	Data index, unused
	ulFunction	UINT32	1-2	Function code 1: READ 2: WRITE
	abData[]	UINT16[]		Field containing user data in case of write. For writing data the size is limited to 106 Dwords. This field is only present in case of writing data.

Table 24: Request structure PROFIBUS_MPI_PACKET_RW_CN_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_RW_CN_CNF_Ttag
{
    TLR_PACKET_HEADER_T          tHead;
    PROFIBUS_MPI_DATA_MPI_RW_CN_T tData;
}PROFIBUS_MPI_PACKET_RW_CN_CNF_T;
```

structure PROFIBUS_MPI_PACKET_RW_CN_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + 2 * ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x430F	PROFIBUS_MPI_CMD_RW_CN_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_CN_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	Data area, unused
	ulCounterAddr	UINT32	0-65534	Counter address
	ulReserved2	UINT32	0	Data index, unused
	ulDataCnt	UINT32		Number of counters to be read or written
			1-111	Number of words of counter data which are read
			1-106	Number of words of counter data which were written
	ulReserved3	UINT32	0	Data index, unused
	ulFunction	UINT32	1-2	Function code
				1: READ 2: WRITE
	abData[]	UINT16[]		Field containing user data in case of reading. For reading data the size is limited to 111 Dwords. This field is only present in case of reading data.

Table 25: Confirmation structure PROFIBUS_MPI_PACKET_RW_CN_CNF_T

4.2.7 PROFIBUS_MPI_CMD_RW_TI_REQ/CNF - Read/Write Timer

This command allows reading or writing data registers within S7[®] components with integrated support for the MPI protocol. If the MPI connection to the remote station is not established and the command is called, the device automatically establishes the connection in the background before accessing the registers.

Packet Structure Reference

```
#define PROFIBUS_MPI_RW_TI_MAX_DATA_SIZE    111

typedef struct PROFIBUS_MPI_DATA_MPI_RW_TI_Ttag
{
    TLR_UINT32    ulRemAddr;
    TLR_UINT32    ulReserved1;
    TLR_UINT32    ulTimerAddr;
    TLR_UINT32    ulReserved2;
    TLR_UINT32    ulDataCnt;
    TLR_UINT32    ulReserved3;
    TLR_UINT32    ulFunction;
    TLR_UINT16    ausData[PROFIBUS_MPI_RW_TI_MAX_DATA_SIZE];
}PROFIBUS_MPI_DATA_MPI_RW_TI_T;

typedef struct PROFIBUS_MPI_PACKET_RW_TI_REQ_Ttag
{
    TLR_PACKET_HEADER_T    tHead;
    PROFIBUS_MPI_DATA_MPI_RW_TI_T    tData;
}PROFIBUS_MPI_PACKET_RW_TI_REQ_T;
```

structure PROFIBUS_MPI_PACKET_RW_TI_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + 2 * ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4310	PROFIBUS_MPI_CMD_RW_TI_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_TI_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	Data area, unused
	ulTimerAddr	UINT32	0-65534	Data address/ Timer address
	ulReserved2	UINT32	0	Data index, unused
	ulDataCnt	UINT32		Number of timers to be read or written
			1-111	Number of words of timer data which are read
			1-106	Number of words of timer data which were written
	ulReserved3	UINT32	0	Data index, unused
	ulFunction	UINT32	1-2	Function code 1: READ 2: WRITE
	abData[]	UINT16[]		Field containing user data in case of write. For writing data the size is limited to 106 Dwords. This field is only present in case of writing data.

Table 26: Request structure PROFIBUS_MPI_PACKET_RW_TI_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_RW_TI_CNF_Ttag
{
    TLR_PACKET_HEADER_T          tHead;
    PROFIBUS_MPI_DATA_MPI_RW_TI_T tData;
}PROFIBUS_MPI_PACKET_RW_TI_CNF_T;
```

structure PROFIBUS_MPI_PACKET_RW_TI_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28 + 2 * ulDataCnt	Packet Data Length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4311	PROFIBUS_MPI_CMD_RW_TI_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_RW_TI_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	Data area, unused
	ulTimerAddr	UINT32	0-65534	Data address/ Timer address
	ulReserved2	UINT32	0	Data index, unused
	ulDataCnt	UINT32		Number of timers to be read or written
			1-111	Number of words of timer data which are read
			1-106	Number of words of timer data which were written
	ulReserved3	UINT32	0	Data index, unused
	ulFunction	UINT32	1-2	Function code 1: READ 2: WRITE
	abData[]	UINT16[]		Field containing user data in case of reading. For reading data the size is limited to 111 Dwords. This field is only present in case of reading data.

Table 27: Confirmation structure PROFIBUS_MPI_PACKET_RW_TI_CNF_T

4.2.8 PROFIBUS_MPI_CMD_R_OP_REQ/CNF – Read Operation State

This command provide the current operational status of the remote station. If there is no MPI connection established to the remote station and the command is called, the device automatically establishes the connection in the background before accessing the registers.

Packet Structure Reference

```
#define OPSTATE_STOP          0  /* CPU state stopped      */
#define OPSTATE_START        1  /* CPU state start up    */
#define OPSTATE_RUN          2  /* CPU state run         */
#define OPSTATE_UNKNOWN      3  /* CPU state unknown     */

typedef struct PROFIBUS_MPI_DATA_MPI_OP_Ttag
{
    TLR_UINT32 ulRemAddr;
    TLR_UINT32 ulReserved1;
    TLR_UINT32 ulReserved2;
    TLR_UINT32 ulReserved3;
    TLR_UINT32 ulReserved4;
    TLR_UINT32 ulReserved5;
    TLR_UINT32 ulReserved7;
    TLR_UINT16 usOpState;
} PROFIBUS_MPI_DATA_MPI_OP_T;

typedef struct PROFIBUS_MPI_PACKET_READ_OP_REQ_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_OP_T tData;
}PROFIBUS_MPI_PACKET_READ_OP_REQ_T;
```

structure PROFIBUS_MPI_PACKET_READ_OP_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	28	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4308	PROFIBUS_MPI_CMD_R_OP_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_OP_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	Unused
	ulReserved2	UINT32	0	Unused
	ulReserved3	UINT32	0	Unused
	ulReserved4	UINT32	0	Unused
	ulReserved5	UINT32	0	Unused
	ulReserved6	UINT32	0	Unused
	usOpState	UINT8	0-3	<i>In case of request this field is ignored. The response can have one of the following values:</i> 0 – Stop 1 - Start 2 - Run 3 - Unknown

Table 28: Request structure PROFIBUS_MPI_PACKET_READ_OP_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_READ_OP_CNF_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_OP_T tData;
}PROFIBUS_MPI_PACKET_READ_OP_CNF_T;
```

structure PROFIBUS_MPI_PACKET_READ_OP_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	29	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4309	PROFIBUS_MPI_CMD_R_OP_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_OP_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	
	ulReserved2	UINT32	0	
	ulReserved3	UINT32	0	
	ulReserved4	UINT32	0	
	ulReserved5	UINT32	0	
	ulReserved6	UINT32	0	
	abData	UINT8	0-3	<i>In case of request this field is ignored. The response can have one of the following values:</i> 0 – Stop 1 - Start 2 - Run 3 - Unknown

Table 29: Confirmation structure PROFIBUS_MPI_PACKET_READ_OP_CNF_T

4.2.9 PROFIBUS_MPI_CMD_RW_TR_REQ/CNF – Send Transparent

This service provides the possibility to read out different types of information from the MPI remote station. If there is no MPI connection established to the remote station and the command is called, the device automatically establishes the connection in the background before accessing the registers.

ATTENTION! If you are no MPI-protocol expert, we do not recommend to use this function.

The Transparent Command is encoded in the first 4 Bytes of the User Data.

Packet Structure Reference

```
#define PROFIBUS_MPI_TR_MAX_DATA_SIZE      222

typedef struct PROFIBUS_MPI_DATA_MPI_TR_Ttag
{
    TLR_UINT32    ulRemAddr;
    TLR_UINT32    ulReserved1;
    TLR_UINT32    ulReserved2;
    TLR_UINT32    ulReserved3;
    TLR_UINT32    ulDataCnt;
    TLR_UINT32    ulReserved5;
    TLR_UINT32    ulReserved6;
    TLR_UINT8     abData[PROFIBUS_MPI_TR_MAX_DATA_SIZE];
}PROFIBUS_MPI_DATA_MPI_TR_T;

typedef struct PROFIBUS_MPI_PACKET_TRANSPARENT_REQ_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_TR_T  tData;
}PROFIBUS_MPI_PACKET_TRANSPARENT_REQ_T;
```

structure PROFIBUS_MPI_PACKET_TRANSPARENT_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	32	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4304	PROFIBUS_MPI_CMD_RW_TR_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_TR_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	Unused
	ulReserved2	UINT32	0	Unused
	ulReserved3	UINT32	0	Unused
	ulDataCnt	UINT32	4	Data count
	ulReserved4	UINT32	0	Unused
	ulReserved5	UINT32	0	Unused
	abData[]	UINT8[]		<i>The request has always 4 Byte user data. The response has a various data count.</i>

Table 30: Request structure PROFIBUS_MPI_PACKET_TRANSPARENT_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_PACKET_TRANSPARENT_CNF_Ttag
{
    TLR_PACKET_HEADER_T      tHead;
    PROFIBUS_MPI_DATA_MPI_TR_T  tData;
}PROFIBUS_MPI_PACKET_TRANSPARENT_CNF_T;
```

structure PROFIBUS_MPI_PACKET_TRANSPARENT_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	>=28	Packet Data Length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4304	PROFIBUS_MPI_CMD_RW_TR_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_TR_T			
	ulRemAddr	UINT32	0-126	Remote Station Address
	ulReserved1	UINT32	0	
	ulReserved2	UINT32	0	
	ulReserved3	UINT32	0	
	ulDataCnt	UINT32	4	Data count
	ulReserved4	UINT32	0	
	ulReserved5	UINT32	0	
	abData[]	UINT8[]		The request has always 4 Byte user data. The response has a various data count.

Table 31: Confirmation structure PROFIBUS_MPI_PACKET_TRANSPARENT_CNF_T

4.2.10 PROFIBUS_MPI_CMD_DC_REQ/CNF - Disconnect Station

While the device automatically establishes the connection to the remote station if not connected previously to it, the disconnect command must be executed by the HOST application. This service should be executed after having finished the access to the station to close the connection properly and the reserved communication channels (SAPs) are released again. If the command is sent before all requests have been processed, processing of these commands will be refused to the application and an error will be issued.

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_DATA_MPI_DC_Ttag
{
    TLR_UINT32    ulRemAddr;
} PROFIBUS_MPI_DATA_MPI_DC_T;

typedef struct PROFIBUS_MPI_PACKET_MPI_DC_REQ_Ttag
{
    TLR_PACKET_HEADER_T    tHead;
    PROFIBUS_MPI_DATA_MPI_DC_T    tData;
} PROFIBUS_MPI_PACKET_MPI_DC_REQ_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_MPI_DC_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	4	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4312	PROFIBUS_MPI_CMD_DC_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_DC_T			
	ulRemAddr	UINT32	0 - 126	Remote station address

Table 32: Request structure PROFIBUS_MPI_PACKET_MPI_DC_REQ_T – Disconnect structure

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_DATA_MPI_DC_Ttag
{
    TLR_UINT32    ulRemAddr;
} PROFIBUS_MPI_DATA_MPI_DC_T;

typedef struct PROFIBUS_MPI_PACKET_MPI_DC_CNF_Ttag
{
    TLR_PACKET_HEADER_T    tHead;
    PROFIBUS_MPI_DATA_MPI_DC_T    tData;
} PROFIBUS_MPI_PACKET_MPI_DC_CNF_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_MPI_DC_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	4	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4313	PROFIBUS_MPI_CMD_DC_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MPI_DC_T			
	ulRemAddr	UINT32	0 - 126	Remote station address

Table 33 Confirmation structure PROFIBUS_MPI_PACKET_MPI_DC_CNF_T - Confirmation of Disconnect Request

4.2.11 PROFIBUS_MPI_CMD_DC_ALL_REQ/CNF – Disconnect All Stations

This packet is used to disconnect all currently open connections by only one single request. Neither the request packet nor the confirmation packet have any parameters.

Packet Structure Reference

```
typedef TLR_EMPTY_PACKET_T PROFIBUS_MPI_PACKET_DC_ALL_REQ_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_DC_ALL_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	0	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4314	PROFIBUS_MPI_CMD_DC_ALL_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch

Table 34: Request structure *PROFIBUS_MPI_PACKET_DC_ALL_REQ_T* – Close all Sockets Request Structure

Packet Structure Reference

```
typedef TLR_EMPTY_PACKET_T PROFIBUS_MPI_PACKET_DC_ALL_CNF_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_DC_ALL_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	0	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4315	PROFIBUS_MPI_CMD_DC_ALL_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch

Table 35 Confirmation structure *PROFIBUS_MPI_PACKET_DC_ALL_CNF_T* – Confirmation of Disconnect all open Handlers Request

4.2.12 PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T - Close a specific Socket Request/Confirmation

This request allows you to close a specific socket of your choice. The sockets are numbered in the range from 0 to 63.

The request packet has one parameter `ulSocket` containing the number of the specific socket to be closed. The confirmation packet does not have any parameters.

Packet Structure Reference

```
/* Close a specific socket */
typedef struct PROFIBUS_MPI_SOCKET_CLOSE_REQ_Ttag
{
    TLR_UINT32 ulSocket;
} PROFIBUS_MPI_SOCKET_CLOSE_REQ_T;

typedef struct PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_MPI_SOCKET_CLOSE_REQ_T tData;
} PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination End Point Identifier, specifying the final receiver of the packet within the Destination Process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32		Source End Point Identifier, specifying the origin of the packet inside the Source Process
	ulLen	UINT32	4	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification as unique number generated by the Source Process of the Packet
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x431A	PROFIBUS_MPI_CMD_SOCKET_CLOSE_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_SOCKET_CLOSE_REQ_T			
	ulSocket	UINT32	0 ... 63	Socket to be closed

Table 36: Request structure PROFIBUS_MPI_PACKET_CLOSE_SOCKET_REQ_T – Close all Sockets Request

Packet Structure Reference

```
typedef TLR_EMPTY_PACKET_T PROFIBUS_MPI_PACKET_CLOSE_SOCKET_CNF_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_CLOSE_SOCKET_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination End Point Identifier, specifying the final receiver of the packet within the Destination Process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32		Source End Point Identifier, specifying the origin of the packet inside the Source Process
	ulLen	UINT32	0	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification as unique number generated by the Source Process of the Packet
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x431B	PROFIBUS_MPI_CMD_SOCKET_CLOSE_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch

Table 37 Confirmation structure *PROFIBUS_MPI_PACKET_CLOSE_SOCKET_CNF_T* - Confirmation of Close all Sockets Request

4.2.13 PROFIBUS_MPI_CMD_MULTIPLE_READ_REQ/CNF - Multiple Read Request

This packet allows reading data registers within S7[®] components with integrated support for the MPI protocol. It can be used for the following S7[®] data types:

#define	PROFIBUS_MPI_AREA_COUNTER	0x01
#define	PROFIBUS_MPI_AREA_TIMER	0x02
#define	PROFIBUS_MPI_AREA_INPUT	0x03
#define	PROFIBUS_MPI_AREA_OUTPUT	0x04
#define	PROFIBUS_MPI_AREA_MARKER	0x05
#define	PROFIBUS_MPI_AREA_DATA	0x06

The request packet has a set of read order structures which is described below. One multiple read request can contain up to 6 order structures. The maximum data which can be transferred depends on the number of requests. Every extra request costs 5 byte data, so if you use all 6 requests you only have 197 byte maximum data size.

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_ORDER_MULTIPLE_READ_REQ_Ttag
{
    TLR_UINT8  bArea;
    TLR_UINT8  bDataType;
    TLR_UINT16 usNumOfElements;
    TLR_UINT16 usDBNum;
    TLR_UINT16 usOffset;
    TLR_UINT8  bBitOffset;
    TLR_UINT8  abReserved[7];
}PROFIBUS_MPI_ORDER_MULTIPLE_READ_REQ_T;

typedef struct PROFIBUS_MPI_DATA_MULTIPLE_READ_REQ_Ttag
{
    TLR_UINT8 bRemAddr;
    TLR_UINT8 bNumOfRequests;
    TLR_UINT8 abReserved[2];
    PROFIBUS_MPI_ORDER_MULTIPLE_READ_REQ_T  atRequests[MAX_MULTIPLE_READ_CMD];
}PROFIBUS_MPI_DATA_MULTIPLE_READ_REQ_T;

typedef struct PROFIBUS_MPI_PACKET_MULTIPLE_READ_REQ_Ttag
{
    TLR_PACKET_HEADER_T          tHead;
    PROFIBUS_MPI_DATA_MULTIPLE_READ_REQ_T tData;
}PROFIBUS_MPI_PACKET_MULTIPLE_READ_REQ_T;
```


Packet Description

structure PROFIBUS_MPI_PACKET_MPI_MULTIPLE_READ_CMD_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32	4	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x431E	PROFIBUS_MPI_CMD_MULTIPLE_READ_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MULTIPLE_READ_REQ_T			
	bRemAddr	UINT8	0 - 126	Remote station address
	bNumOfRequests	UINT8	1 - 6	Number of requests
	abReserved[2]	UINT8[2]		Reserved area
	atRequests[]	PROFIBUS_MPI_ORDER_MULTIPLE_READ_REQ_T[]		Array of request structures. The field is described below

Table 38: PROFIBUS_MPI_PACKET_MPI_MULTIPLE_READ_CMD_REQ_T – Multiple Read Request

Structure Description

structure PROFIBUS_MPI_ORDER_MULTIPLE_READ_REQ_T				
Type:				
Area	Variable	Type	Value / Range	Description
	bArea	UINT8		The area which should be read
	bDataType	UINT8		Data type
	usNumOfElements	UINT16		Number of elements which should be read. For type bool only 1 is allowed.
	usDBNum	UINT16		Data block number within the S7. This field is only used for the area data, in all other cases it must be set to 0.
	usOffset	UINT16		The offset of the element which should be read. In case of bool the offset means the byte offset
	bBitOffset	UINT8	0 - 7	Bit of data, this field is only used for the data type bool
	abReserved[7]	UINT8[]		Reserved for future use

Table 39: Structure PROFIBUS_MPI_ORDER_MULTIPLE_READ_REQ_T

The response contains all responses of the request. It is possible that the status code is TLR_S_OK in case that one or more request has failed. Every response has its own access result field. The codes are:

```
#define PROFIBUS_MPI_ACCESSRESULT_NO_ERROR      0xFF
#define PROFIBUS_MPI_ACCESSRESULT_HARDWARE_ERROR 0x01
#define PROFIBUS_MPI_ACCESSRESULT_INVALID_CODE  0x03
#define PROFIBUS_MPI_ACCESSRESULT_INVALID_ADDRESS 0x05
#define PROFIBUS_MPI_ACCESSRESULT_UNKNOWN_DATA_TYPE 0x06
#define PROFIBUS_MPI_ACCESSRESULT_PDU_ERROR    0x07
#define PROFIBUS_MPI_ACCESSRESULT_OBJECT_NOT_EXIST 0x0A
```

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_ORDER_MULTIPLE_READ_CNF_Ttag
{
    TLR_UINT8  bArea;
    TLR_UINT8  bDataType;
    TLR_UINT16 usNumOfElements;
    TLR_UINT16 usDBNum;
    TLR_UINT16 usOffset;
    TLR_UINT8  bBitOffset;
    TLR_UINT8  abReserved[2];
    TLR_UINT8  bAccessResult;
    TLR_UINT16 usDataOffset;
    TLR_UINT16 usDataLen;
}PROFIBUS_MPI_ORDER_MULTIPLE_READ_CNF_T;

typedef struct PROFIBUS_MPI_DATA_MULTIPLE_READ_CNF_Ttag
{
    TLR_UINT8 bRemAddr;
    TLR_UINT8 bNumOfResponses;
    TLR_UINT8 abReserved[2];
    union {
        PROFIBUS_MPI_ORDER_MULTIPLE_READ_CNF_T atResponses[6];
        TLR_UINT8 abData[318];
    };
}PROFIBUS_MPI_DATA_MULTIPLE_READ_CNF_T;

typedef struct PROFIBUS_MPI_PACKET_MULTIPLE_READ_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_MPI_DATA_MULTIPLE_READ_CNF_T tData;
}PROFIBUS_MPI_PACKET_MULTIPLE_READ_CNF_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_MULTIPLE_READ_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32		Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x431F	PROFIBUS_MPI_CMD_MULTIPLE_READ_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MULTIPLE_READ_CNF_T			
	bRemAddr	UINT8	0 - 126	Remote station address
	bNumOfRequests	UINT8	1 - 6	Number of requests
	abReserved[2]	UINT8[2]		
	Union{ atResponses[], abData[] }	PROFIBUS_MPI_ORDER_MULTIPLE_READ_CNF_T[] UINT8[]	NOTE: This two field are within a union! All offsets in the response structure refer to the beginning of abData	Array of responses structures. The field is described below The data of the response

Table 40 PROFIBUS_MPI_PACKET_MPI_MULTIPLE_READ_CMD_CNF_T - Confirmation of Multiple Read Request

Structure Description

structure PROFIBUS_MPI_ORDER_MULTIPLE_READ_CNF_T				
Type:				
Area	Variable	Type	Value / Range	Description
	bArea	UINT8		The area which should be read
	bDataType	UINT8		Data type
	usNumOfElements	UINT16		Number of elements which should be read. For type bool only 1 is allowed.
	usDBNum	UINT16		Data block number within the S7. This field is only used for the area data, in all other cases it must be set to 0.
	usOffset	UINT16		The offset of the element which should be read. In case of bool the offset means the byte offset
	bBitOffset	UINT8	0 - 7	Bit of data, this field is only used for the data type bool
	abReserved[2]	UINT8[2]		Reserved for future use
	bAccessResult	UINT8		Access result of this request
	usDataOffset	UINT16		Data offset in bytes. This offset refers to the abData[] field in PROFIBUS_MPI_DATA_MULTIPLE_READ_CNF_T structure.
	usDataLen	UINT16		Length of the data

Table 41: Structure PROFIBUS_MPI_ORDER_MULTIPLE_READ_CNF_T

4.2.14 PROFIBUS_MPI_CMD_MULTIPLE_WRITE_REQ/CNF - Multiple Write Request

This packet allows writing data registers within S7[®] components with integrated support for the MPI protocol. It can be used for the following S7[®] data types:

#define	PROFIBUS_MPI_AREA_COUNTER	0x01
#define	PROFIBUS_MPI_AREA_TIMER	0x02
#define	PROFIBUS_MPI_AREA_INPUT	0x03
#define	PROFIBUS_MPI_AREA_OUTPUT	0x04
#define	PROFIBUS_MPI_AREA_MARKER	0x05
#define	PROFIBUS_MPI_AREA_DATA	0x06

The request packet has a set of write order structures which is described below. One multiple write request can contain up to 6 order structures. The maximum data which can be transferred depends on the number of requests. Every extra request costs 17 byte data, so if you use all 6 requests you only have 110 byte maximum data size.

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_REQ_Ttag
{
    TLR_UINT8  bArea;
    TLR_UINT8  bDataType;
    TLR_UINT16 usNumOfElements;
    TLR_UINT16 usDBNum;
    TLR_UINT16 usOffset;
    TLR_UINT8  bBitOffset;
    TLR_UINT8  abReserved[3];
    TLR_UINT16 usDataOffset;
    TLR_UINT16 usDataLen;
} PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_REQ_T;

typedef struct PROFIBUS_MPI_DATA_MULTIPLE_WRITE_REQ_Ttag
{
    TLR_UINT8 bRemAddr;
    TLR_UINT8 bNumOfRequests;
    TLR_UINT8 abReserved[2];
    union __TLR_PACKED_PRE{
        PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_REQ_T atRequests[6];
        TLR_UINT8 abData[308];
    };
} PROFIBUS_MPI_DATA_MULTIPLE_WRITE_REQ_T;

typedef struct PROFIBUS_MPI_PACKET_MULTIPLE_WRITE_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_MPI_DATA_MULTIPLE_WRITE_REQ_T tData;
} PROFIBUS_MPI_PACKET_MULTIPLE_WRITE_REQ_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_MULTIPLE_WRITE_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32		Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4320	PROFIBUS_MPI_CMD_MULTIPLE_WRITE_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MULTIPLE_WRITE_REQ_T			
	bRemAddr	UINT8	0 - 126	Remote station address
	bNumOfRequests	UINT8	1 - 6	Number of requests
	abReserved[2]	UINT8[2]		
	Union{ atResponses[], abData[] }	PROFIBUS_MPI_ORDER_MULTIPLE_READ_CONF_T[] UINT8[]	NOTE: This two fields are within a union! All offsets in the request structure refer to the beginning of abData	Array of responses structures. The field is described below The data of the response

Table 42: Request structure PROFIBUS_MPI_PACKET_MPI_MULTIPLE_WRITE_CMD_REQ_T – Disconnect structure

Structure Description

structure PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_REQ_T				
Type:				
Area	Variable	Type	Value / Range	Description
	bArea	UINT32		The area which should be write
	bDataType	UINT32		Data type
	usNumOfElements	UINT32		Number of elements which should be write. For type bool only 1 is allowed.
	usDBNum	UINT16		Data block number within the S7. This field is only used for the area data, in all other cases it must be set to 0.
	usOffset	UINT16		The offset of the element which should be write. In case of bool the offset means the byte offset
	bBitOffset	UINT8	0 - 7	Bit of data, this field is only used for the data type bool
	abReserved[3]	UINT8[3]		Reserved for future use
	usDataOffset	UINT16		Data offset in bytes. This offset refers to the abData[] field in PROFIBUS_MPI_DATA_MULTIPLE_WRITE_REQ_T structure.
	usDataLen	UINT16		Length of the data

Table 43: Structure PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_REQ_T

Packet Structure Reference

```
typedef struct PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_CNF_Ttag
{
    TLR_UINT8    bArea;
    TLR_UINT8    bDataType;
    TLR_UINT16   usNumOfElements;
    TLR_UINT16   usDBNum;
    TLR_UINT16   usOffset;
    TLR_UINT8    bBitOffset;
    TLR_UINT8    abReserved[2];
    TLR_UINT8    bAccessResult;
    TLR_UINT8    abReserved2[4];
}PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_CNF_T;

typedef struct PROFIBUS_MPI_DATA_MULTIPLE_WRITE_CNF_Ttag
{
    TLR_UINT8    bRemAddr;
    TLR_UINT8    bNumOfResponses;
    TLR_UINT8    abReserved[2];
    PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_CNF_T atResponses[6];
}PROFIBUS_MPI_DATA_MULTIPLE_WRITE_CNF_T;

typedef struct PROFIBUS_MPI_PACKET_MULTIPLE_WRITE_CNF_Ttag
{
    TLR_PACKET_HEADER_T          tHead;
    PROFIBUS_MPI_DATA_MULTIPLE_WRITE_CNF_T tData;
}PROFIBUS_MPI_PACKET_MULTIPLE_WRITE_CNF_T;
```

Packet Description

structure PROFIBUS_MPI_PACKET_MULTIPLE_WRITE_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32		Destination Queue Reference
	ulSrcId	UINT32		Source Queue Reference
	ulLen	UINT32		Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification As Unique Number
	ulSta	UINT32		See section 5.1 <i>PROFIBUS MPI-Error Codes</i> on page 161.
	ulCmd	UINT32	0x4321	PROFIBUS_MPI_CMD_MULTIPLE_WRITE_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_MPI_DATA_MULTIPLE_WRITE_CNF_T			
	bRemAddr	UINT8	0 - 126	Remote station address
	bNumOfRequests	UINT8	1 - 6	Number of requests
	abReserved[2]	UINT8[2]		
	atResponses[]	PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_CNF_T []		Array of responses structures. The field is described below

Table 44 Confirmation structure PROFIBUS_MPI_PACKET_MPI_MULTIPLE_WRITE_CMD_CNF_T - Confirmation of Multiple Write Request

Structure Description

structure PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_CNF_T				
Type:				
Area	Variable	Type	Value / Range	Description
	bArea	UINT32		The area which should be write
	bDataType	UINT32		Data type
	usNumOfElements	UINT32		Number of elements which should be write. For type bool only 1 is allowed.
	usDBNum	UINT16		Data block number within the S7. This field is only used for the area data, in all other cases it must be set to 0.
	usOffset	UINT16		The offset of the element which should be write. In case of bool the offset means the byte offset
	bBitOffset	UINT8	0 - 7	Bit of data, this field is only used for the data type bool
	abReserved[2]	UINT8[2]		Reserved for future use
	bAccessResult	UINT8		Access result of this request
	abReserved2[4]	UINT8[4]		Reserved for future use

Table 45: Structure PROFIBUS_MPI_ORDER_MULTIPLE_WRITE_CNF_T

4.3 Packets of the PROFIBUS-DL-Task

At the Profibus DL task, the FDL functionality is implemented.

In detail, the following functionality is provided by the PROFIBUS-DL-Task:

Overview over Packets of the PROFIBUS DL-Task			
No. of section	Packet	Command code (REQ/CNF or IND/RES)	Page
4.3.1	PROFIBUS_DL_CMD_END_PROCESS_REQ/CNF– Finish the DL Task	0x100/ 0x101	86
4.3.2	PROFIBUS_DL_CMD_START_DLE_REQ/CNF – Starts the DL-Layer	0x102/ 0x103	88
4.3.3	PROFIBUS_DL_CMD_STOP_DLE_REQ/CNF – Stop DL Layer	0x104/ 0x105	90
4.3.4	PROFIBUS_DL_CMD_DATA_ACK_REQ/CNF - Send SDA Service	0x106/ 0x107	92
4.3.5	PROFIBUS_DL_CMD_DATA_ACK_IND - Receive SDA Service	0x108	99
4.3.6	PROFIBUS_DL_CMD_DATA_REQ/CNF - Send SDN Service	0x10A/ 0x10B	101
4.3.7	PROFIBUS_DL_CMD_DATA_IND - Receive SDN Service Indication	0x10C	107
4.3.8	PROFIBUS_DL_CMD_DATA_REPLY_REQ/CNF - Send SRD Service	0x10E/ 0x10F	110
4.3.9	PROFIBUS_DL_CMD_DATA_REPLY_IND - Receive SRD Service Indication	0x110	117
4.3.10	PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_REQ/CNF - Reply Update Service	0x112/ 0x113	120
4.3.11	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ/CNF - Activate an SAP for Request	0x120/ 0x121	124
4.3.12	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ/CNF - Activate an SAP for Responder Functionality	0x122/ 0x123	130
4.3.13	PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ/CNF - Deactivate an SAP for Request	0x128/ 0x129	135
4.3.14	PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ/CNF - Load the Bus Parameter Set	0x126/ 0x127	138
4.3.15	PROFIBUS_DL_CMD_SET_VALUE_REQ/CNF - Set Value Service	0x12A/ 0x12B	144

4.3.16	PROFIBUS_DL_CMD_SEND_FDL_STATUS_REQ - Send FDL Status	0x0132/ 0x0133	147
4.3.17	PROFIBUS_DL_CMD_GET_LMS_REQ/CNF - Get List of active Stations	0x0134/ 0x0135	150
4.3.18	PROFIBUS_DL_CMD_REGISTER_LMS_REQ/CNF - Register an Application to receive LMS Change Indications	0x0136/ 0x0137	153
4.3.19	PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for Change in LMS	0x0138/ 0x0139	156

Table 46: Overview over the Packets of the PROFIBUS DL-Task of the PROFIBUS MPI Protocol Stack

4.3.1 PROFIBUS_DL_CMD_END_PROCESS_REQ/CNF– Finish the DL Task

This packet shall be sent to close the DL-Task. All resources will be deallocated and the task finishes.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_PACKET_CMD_END_PROCESS_REQ _Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_CMD_END_PROCESS_REQ_T;
```

Packet Description

Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	0	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section 5.2 Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
				Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task				
ulCmd	UINT32	0x100	PROFIBUS_DL_CMD_END_PROCESS_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					

Table 47: PROFIBUS_DL_CMD_END_PROCESS_REQ– Finish the DL Task

Packet Structure Reference

```
typedef struct PROFIBUS_DL_PACKET_CMD_END_PROCESS_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_CMD_END_PROCESS_CNF_T;
```

Packet Description

Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	0	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section 5.2 Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
				Hexadecimal Value	Definition Description			
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task								
ulCmd	UINT32	0x101	PROFIBUS_DL_CMD_END_PROCESS_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					

Table 48: PROFIBUS_DL_CMD_END_PROCESS_CNF– Finish the DL Task

4.3.2 PROFIBUS_DL_CMD_START_DLE_REQ/CNF – Starts the DL-Layer

This packet should be sent to start the DL Layer with the current bus parameter settings.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_PACKET_CMD_START_DLE_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_CMD_START_DLE_REQ_T;
```

Packet Description

Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	0	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task“ <table border="1"><thead><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr></thead><tbody><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></tbody></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
				Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.				
ulCmd	UINT32	0x102	PROFIBUS_DL_CMD_START_DLE_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					

Table 49: PROFIBUS_DL_CMD_START_DLE_REQ – Start the DL-Layer Request

Packet Structure Reference

```
typedef struct PROFIBUS_DL_PACKET_CMD_START_DLE_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_CMD_START_DLE_CNF_T;
```

Packet Description

Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	0	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
				Hexadecimal Value	Definition Description			
0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.							
Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.								
ulCmd	UINT32	0x103	PROFIBUS_DL_CMD_START_DLE_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					

Table 50: PROFIBUS_DL_CMD_START_DLE_CNF – Confirmation of Start the DL-Layer Request

4.3.3 PROFIBUS_DL_CMD_STOP_DLE_REQ/CNF – Stop DL Layer

This packet shall be sent to stop the DL Layer.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_PACKET_CMD_STOP_DLE_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_CMD_STOP_DLE_REQ_T;
```

Packet Description

Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	0	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x104	PROFIBUS_DL_CMD_STOP_DLE_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					

Table 51: PROFIBUS_DL_CMD_STOP_DLE_REQ– Stop DL Layer Request

Packet Structure Reference

```
typedef struct PROFIBUS_DL_PACKET_CMD_STOP_DLE_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_CMD_STOP_DLE_CNF_T;
```

Packet Description

Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	0	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
				Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.				
ulCmd	UINT32	0x105	PROFIBUS_DL_CMD_STOP_DLE_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					

Table 52: PROFIBUS_DL_CMD_STOP_DLE_CNF – Confirmation of Stop DL Layer Request

4.3.4 PROFIBUS_DL_CMD_DATA_ACK_REQ/CNF - Send SDA Service

This packet provides the SDA (Send Data with Acknowledge) service for acknowledged connectionless data transfer with immediate response. It allows transparently sending an SDA frame with variable data length to the PROFIBUS network. At the remote station, the user data (which is contained in the Service Data Unit represented by the variable `abSDU[]`) is delivered to the user. The length of the Service Data Unit is limited to at most 246 bytes

The local confirmation message of this command informs about the receipt or non-receipt of the user data in the remote station. Within the confirmation packet, a value of `ulSta` equal to the following indicates successful receipt of the SDA message:

- TLR_S_OK (0x00000000)
- TLR_E_PROFIBUS_DL_ACK_NR (0xC0060083L)
- TLR_E_PROFIBUS_DL_ACK_DH (0xC0060086L)
- TLR_E_PROFIBUS_DL_ACK_DL (0xC0060087L)

All other values indicate the failure of the SDA service.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DATA_ACK_REQ_Ttag {
    TLR_UINT8 bSrvCls;      /* Service Class */
    TLR_UINT8 bDstAddr;     /* Destination address */
    TLR_UINT8 bDstSAPIdx;   /* Destination Service Access Point */
    TLR_UINT8 bSrcSAPIdx;   /* Source Service Access Point */
    TLR_UINT8 abPad[4];     /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_DATA_ACK_REQ_T;

#define PROFIBUS_DL_DATA_ACK_REQ_SIZE (sizeof(PROFIBUS_DL_DATA_ACK_REQ_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Request-Packet for acknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_DATA_ACK_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_ACK_REQ_T tData;      /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DATA_ACK_REQ_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_ACK_REQ_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	8 + n	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x106	PROFIBUS_DL_CMD_DATA_ACK_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_ACK_REQ_T							
	bSrvCls	UINT8	0,1	Service Class (indicating the priority)				
	bDstAddr	UINT8	0-126	Destination address within the Profibus network				
				Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bDstSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Destination Service Access Point				
	bSrcSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Source Service Access Point				
	abPad	UINT8[4]	0	Padding bytes				
abSDU	UINT8[]		Service Data Unit					

Table 53: PROFIBUS_DL_CMD_DATA_ACK_REQ - Send SDA Service

Additional explanations:

- 1) The parameter `bSrvCls` defines the FDL priority for the data transfer. Two priorities are possible in this context:
 - `PROFIBUS_DL_SERVICE_CLASS_LOW = 0`
 - `PROFIBUS_DL_SERVICE_CLASS_HIGH = 1`
- 2) The parameter `bDstAddr` defines the FDL address of the remote station. For this parameter, the value 127 is not permissible as remote address because it signifies the global address used for broadcasts.
- 3) The parameter `bDsap` defines the service access point of the remote user. The value 64 represents the NIL SAP.
- 4) The parameter `bSsap` defines the service access point of the local user. The value 64 again represents the NIL SAP.

Packet Structure Reference

```

typedef struct PROFIBUS_DL_DATA_ACK_CNF_Ttag {
    TLR_UINT8 bSrvCls;      /* Service Class */
    TLR_UINT8 bDstAddr;     /* Destination address */
    TLR_UINT8 bDstSAPIdx;   /* Destination Service Access Point */
    TLR_UINT8 bSrcSAPIdx;   /* Source Service Access Point */
} PROFIBUS_DL_DATA_ACK_CNF_T;

/* Confirmation-Packet for acknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_DATA_ACK_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_ACK_CNF_T tData;    /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DATA_ACK_CNF_T;

```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_ACK_CNF_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	4	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x107	PROFIBUS_DL_CMD_DATA_ACK_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_ACK_CNF_T							
	bSrvCls	UINT8	0,1	Service Class				
	bDstAddr	UINT8	0-126	Destination address Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bDstSAPIdx	UINT8	0-62, 64 = NIL SAP	Destination Service Access Point				

	bSrcSAPIdx	UINT8	0-62, 64 = NIL SAP	Source Service Access Point
--	------------	-------	-----------------------	-----------------------------

Table 54: PROFIBUS_DL_CMD_DATA_ACK_CNF – Confirmation of Send SDA Service

Packet Status/Error

Definition / (Value)	Description
TLR_S_OK (0x00000000)	Status ok
TLR_E_PROFIBUS_DL_ACK_UE (0xC0060080L)	The remote station the service has been sent to indicates a User Error as service acknowledgement.
TLR_E_PROFIBUS_DL_ACK_RR (0xC0060081L)	Some resource is not available at the remote station as indicated by the remote station, for instance the slave has no left buffer space for the requested service. <u>Action:</u> Increase buffer space or reduce amount of required buffer space.
TLR_E_PROFIBUS_DL_ACK_RS (0xC0060082L)	The remote station the service has been sent to indicates a Service Access Point Error as service acknowledgement.
TLR_E_PROFIBUS_DL_ACK_NR (0xC0060083L)	The remote station the service has been sent to confirms its positive reception but has no data to confirm.
TLR_E_PROFIBUS_DL_ACK_RDH (0xC0060084L)	The remote station the service has been sent to, confirms its reception negatively but has returned low priority data in the response.
TLR_E_PROFIBUS_DL_ACK_RDL (0xC0060085L)	The remote station the service has been sent to, confirms its reception negatively but has returned high priority data in the response.
TLR_E_PROFIBUS_DL_ACK_DH (0xC0060086L)	The remote station the service has been sent to, confirms its reception positively and has returned low priority data in the response.
TLR_E_PROFIBUS_DL_ACK_DL (0xC0060087L)	The remote station the service has been sent to, confirms its reception positively and has returned high priority data in the response.
TLR_E_PROFIBUS_DL_ACK_NA (0xC0060088L)	The remote station the service has been sent to shows no or no plausible reaction at all.
TLR_E_PROFIBUS_DL_ACK_UNKNOWN (0xC0060089L)	The remote station the service has been sent has returned an unknown acknowledgement code.

Table 55: PROFIBUS_DL_CMD_DATA_ACK_CNF - Packet Status/Error

For the following causes of failures some hints can be given:

Error Definition / (Value)	Error source	Description of problem and necessary actions
TLR_E_PROFIBUS_DL_ACK_LR (0xC006008BL)	Slave	The local resources needed to execute the requested service are not available or not sufficient for instance the slave has no left buffer space for the requested service. <u>Action:</u> Increase buffer space or reduce amount of required buffer space.
TLR_E_PROFIBUS_DL_ACK_LS (0xC006008AL)	Slave	The requested function of master is not activated within the slave, i.e. the slave does not support the service which had been requested. <u>Action:</u> If possible, use another slave providing the requested functionality.
TLR_E_PROFIBUS_DL_ACK_NR (0xC0060083L)	Slave	No answer-data available, as the slave did not sent back any data. <u>Action:</u> Check slave for correct operation.
TLR_E_PROFIBUS_DL_ACK_NA (0xC0060088L)	Slave	No response of the station at all <u>Action:</u> Check for correct network wiring, also check the bus address of slave or baud rate support.
TLR_E_PROFIBUS_DL_ACK_DS (0xC006008CL)	Network in general	The master is not included into the logical token ring. <u>Action:</u> Check master DP-Address or highest-station-address of other masters Examine bus wiring to avoid bus short circuits.

Table 56: Reported Errors, their Sources and Possible Actions

4.3.5 PROFIBUS_DL_CMD_DATA_ACK_IND - Receive SDA Service

This indication is transferred from the remote FDL controller to the local host, when an SDA (Send Data with Acknowledge) frame has been received from an initiator on the Profibus network. The reception of the indication message needs no acknowledgment message to the local FDL.

The data of the message is contained in the Service Data Unit, represented by the field `abSDU[]`. The length of the Service Data Unit is limited to at most 246 bytes

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DATA_ACK_IND_Ttag {
    TLR_UINT8 bSrvCls;      /* Service Class */
    TLR_UINT8 bDstAddr;     /* Destination address */
    TLR_UINT8 bDstSAPIdx;   /* Destination Service Access Point */
    TLR_UINT8 bSrcAddr;     /* Source address */
    TLR_UINT8 bSrcSAPIdx;   /* Source Service Access Point */
    TLR_UINT8 abPad[3];     /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_DATA_ACK_IND_T;

#define PROFIBUS_DL_DATA_ACK_IND_SIZE (sizeof(PROFIBUS_DL_DATA_ACK_IND_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Indication Packet for acknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_DATA_ACK_IND_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_ACK_IND_T tData;
} PROFIBUS_DL_PACKET_DATA_ACK_IND_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_ACK_IND_T				
Type: Indication				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	8 + n	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet

	ulSta	UINT32		See section „Error Codes of the MPI AP-Task“ <table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
	Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.							
	ulCmd	UINT32	0x108	PROFIBUS_DL_CMD_DATA_ACK_IND - Command				
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_ACK_IND_T							
	bSrvCls	UINT8	0,1	Service Class				
	bDstAddr	UINT8	0-126	Destination address Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bDstSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Destination Service Access Point				
	bSrcAddr	UINT8	0-126	Source address Note: Mask out the highest bit when working with bSrcAddr in order to avoid disturbances as this bit stores a separate information.				
	bSrcSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Source Service Access Point				
	abPad	UINT8[3]	0	Padding bytes				
	abSDU	UINT8[]		Service Data Unit				

Table 57: PROFIBUS_DL_CMD_DATA_ACK_IND - Receive SDA Service

Additional explanations:

1. The parameter bSrcSAPIdx and bDstSAPIdx specify the source and destination Service Access Points of the received SDA frame.
2. The parameter bDstAddr defines the FDL address of the remote station, to which this service has been sent.
3. The parameter bSrcAddr defines the FDL address of the remote station, from which this service has been sent.
4. The parameter bSrvCls specifies the FDL priority of the received SDA frame. Two priorities are possible in this context:
 - PROFIBUS_DL_SERVICE_CLASS_LOW = 0
 - PROFIBUS_DL_SERVICE_CLASS_HIGH = 1

4.3.6 PROFIBUS_DL_CMD_DATA_REQ/CNF - Send SDN Service

This packet provides the SDN (Send Data with no Acknowledge) service for unacknowledged connectionless data transfer to one or more stations in the PROFIBUS network. It allows transparently sending a SDN (Send Data with no Acknowledge) frame to the PROFIBUS network with variable data length to a single remote station, to many remote stations (multicast) or to all remote stations (broadcast) at the same time. At the remote station(s), the user data (which is contained in the Service Data Unit represented by the variable `abSDU[]`) is delivered to the user. The length of the Service Data Unit is limited to at most 246 bytes

The local confirmation message of this command informs just the end of the transfer, but not about successful reception of the data. Within the confirmation packet, a value of `ulSta` equal to `TLR_S_OK` (0x00000000) indicates successful receipt of the SDN message: All other values indicate the failure of the SDN service.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DATA_REQ_Ttag {
    TLR_UINT8 bSrvCls; /* Service Class */
    TLR_UINT8 bDstAddr; /* Destination address */
    TLR_UINT8 bDstSAPIdx; /* Destination Service Access Point */
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point */
    TLR_UINT8 abPad[4]; /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_DATA_REQ_T;

#define PROFIBUS_DL_DATA_REQ_SIZE (sizeof(PROFIBUS_DL_DATA_REQ_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Request-Packet for unacknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_DATA_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_REQ_T tData; /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DATA_REQ_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_REQ_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	8 + n	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task“ <table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table> Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
ulCmd	UINT32	0x10A	PROFIBUS_DL_CMD_DATA_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_REQ_T							
	bSrvCls	UINT8	0,1	Service Class (indicating the priority)				
	bDstAddr	UINT8	0-126, 127 (= Broadcast)	Destination address within the PROFIBUS network Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bDstSAPIdx	UINT8	0-63, 64 = NIL SAP	Destination Service Access Point				
	bSrcSAPIdx	UINT8	0-62, 64 = NIL SAP	Source Service Access Point				
	abPad	UINT8[4]	0	Padding bytes				
	abSDU	UINT8[]		Service Data Unit				

Table 58: PROFIBUS_DL_CMD_DATA_REQ - Send SDN Service Request

Additional explanations:

1. The parameter `bSrvCls` defines the FDL priority for the data transfer. Two priorities are possible in this context:
 - `PROFIBUS_DL_SERVICE_CLASS_LOW = 0`
 - `PROFIBUS_DL_SERVICE_CLASS_HIGH = 1`
2. The parameter `bDstAddr` defines the FDL address of the remote station. The allowed range for this parameter extends from 0 to 126.
3. The parameter `bDstSAPIdx` defines the service access point of the remote user. The value 64 represents the NIL SAP.
4. The parameter `bSrcSAPIdx` defines the service access point of the local user. The allowed range for this parameter extends from 0 to 62. A value `bSrcSAPIdx` of 63 is not permitted in this command. The value 64 again represents the NIL SAP.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DATA_CNF_Ttag {
    TLR_UINT8 bSrvCls;      /* Service Class */
    TLR_UINT8 bDstAddr;     /* Destination address */
    TLR_UINT8 bDstSAPIdx;   /* Destination Service Access Point */
    TLR_UINT8 bSrcSAPIdx;   /* Source Service Access Point */
} PROFIBUS_DL_DATA_CNF_T;

/* Confirmation-Packet for unacknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_DATA_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_CNF_T tData; /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DATA_CNF_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle, unchanged
	ulSrc	UINT32		Source queue handle, unchanged
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged
	ulLen	UINT32	4	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See Table 60: PROFIBUS_DL_CMD_DATA_CNF - Packet Status/Error
	ulCmd	UINT32	0x10B	PROFIBUS_DL_CMD_DATA_CNF - Command
	ulExt	UINT32	0	Extension, unchanged
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_DL_DATA_CNF_T			
	bSrcCls	UINT8	0,1	Service Class
	bDstAddr	UINT8	0-126, 127 (= Broadcast)	Destination address Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.
	bDstSAPIdx	UINT8	0-63, 64 = NIL SAP	Destination Service Access Point
	bSrcSAPIdx	UINT8	0-62, 64 = NIL SAP	Source Service Access Point

Table 59: PROFIBUS_DL_CMD_DATA_CNF – Confirmation of Send SDN Service Request

Packet Status/Error

Definition / (Value)	Description
TLR_S_OK (0x00000000)	Status ok
TLR_E_PROFIBUS_DL_ACK_RR (0xC0060081L)	The remote station the service has been sent to indicates a Resource Error as service acknowledgment
TLR_E_PROFIBUS_DL_ACK_NA (0xC0060088L)	The remote station the service has been sent to shows no or no plausible reaction at all.
TLR_E_PROFIBUS_DL_ACK_LS (0xC006008AL)	The requested service is not activated within the local SAP configuration.
TLR_E_PROFIBUS_DL_ACK_LR (0xC006008BL)	The local resources needed to execute the requested service are not available or not sufficient.
TLR_E_PROFIBUS_DL_ACK_DS (0xC006008CL)	Master not into the logical token ring

Table 60: PROFIBUS_DL_CMD_DATA_CNF - Packet Status/Error

For the following causes of failures some hints can be given:

Error Definition / (Value)	Error source	Description of problem and necessary actions
TLR_E_PROFIBUS_DL_ACK_RR (0xC0060081L)	Slave	Some resource is not available at the remote station as indicated by the remote station, for instance the slave has no left buffer space for the requested service. <u>Action:</u> Increase buffer space or reduce amount of required buffer space.
TLR_E_PROFIBUS_DL_ACK_DS (0xC006008CL)	Network in general	The master is not included into the logical token ring. <u>Action:</u> Check master DP-Address or highest-station-address of other masters Examine bus wiring to avoid bus short circuits.

Table 61: Reported Errors, their Sources and Possible Actions

4.3.7 PROFIBUS_DL_CMD_DATA_IND - Receive SDN Service Indication

This indication is passed from the remote FDL controller to the local host, when a SDN (Send Data with Acknowledge) frame has been received. The reception of the indication message needs no acknowledgment message to the local FDL.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DATA_IND_Ttag {
    TLR_UINT8 bSrvCls;      /* Service Class */
    TLR_UINT8 bDstAddr;     /* Destination address */
    TLR_UINT8 bDstSAPIdx;   /* Destination Service Access Point */
    TLR_UINT8 bSrcAddr;     /* Source address */
    TLR_UINT8 bSrcSAPIdx;   /* Source Service Access Point */
    TLR_UINT8 abPad[3];     /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_DATA_IND_T;

#define PROFIBUS_DL_DATA_IND_SIZE (sizeof(PROFIBUS_DL_DATA_IND_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Indication Packet for unacknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_DATA_IND_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_IND_T tData;
} PROFIBUS_DL_PACKET_DATA_IND_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_IND_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	8 + n	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task“ <table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table> Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
ulCmd	UINT32	0x10C	PROFIBUS_DL_CMD_DATA_IND - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_IND_T							
	bSrcCls	UINT8	0,1	Service Class				
	bDstAddr	UINT8	0-126, 127 (= Broadcast)	Destination address Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bDstSAPIdx	UINT8	0-63, 64 (= NIL SAP)	Destination Service Access Point				
	bSrcAddr	UINT8	0-126	Source address Note: Mask out the highest bit when working with bSrcAddr in order to avoid disturbances as this bit stores a separate information.				
	bSrcSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Source Service Access Point				
	abPad	UINT8[3]	0	Padding bytes				

	abSDU	UINT8[]		Service Data Unit
--	-------	---------	--	-------------------

Table 62: PROFIBUS_DL_CMD_DATA_IND - Receive SDN Service Indication

Additional explanations:

1. The parameter `bSrcSAPIdx` and `bDstSAPIdx` specify the source and destination service access points of the received SDN frame. The allowed range for this parameter extends from 0 to 62. The value 64 is also allowed. It represents the NIL SAP. A value `bSrcSAPIdx` of 63 is not permissible in this command because it represents the global access.
2. The parameter `bSrcAddr` defines the FDL address of the local station. Values between 0 and 126 are allowed.
3. The parameter `bDstAddr` defines the FDL address of the remote station, which has sent this service. Values between 0 and 126 are allowed. 127 signifies a broadcast.
4. The parameter `bSrvCls` defines the FDL priority for the data transfer. Two priorities are possible in this context:
 - PROFIBUS_DL_SERVICE_CLASS_LOW = 0
 - PROFIBUS_DL_SERVICE_CLASS_HIGH = 1

4.3.8 PROFIBUS_DL_CMD_DATA_REPLY_REQ/CNF - Send SRD Service

This packet shall be used to send the service SRD (Send and Request with Reply) for bidirectional connectionless data exchange. It allows transferring data to a single remote station and at the same time to request data that was made available by the remote user at an earlier time. The data of the response frame is sent back in the response message transparently.

This packet provides the SRD (Send and Request with Reply) service for data transfer with reply. It allows transferring data to a single remote station in the PROFIBUS network and at the same time to request data that was made available by the remote user at an earlier time. At the remote station, the user data (which is contained in the Service Data Unit represented by the variable `abSDU[]`) is delivered to the user. The length of the Service Data Unit is limited to at most 246 bytes



Note: This service also allows to request data from the remote station without sending data to the remote station (in this case `abSDU` should be empty).

The local confirmation message of this command informs about the receipt or non-receipt of the user data in the remote station.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DATA_REPLY_REQ_Ttag {
    TLR_UINT8 bSrvCls;      /* Service Class */
    TLR_UINT8 bDstAddr;     /* Destination address */
    TLR_UINT8 bDstSAPIdx;   /* Destination Service Access Point */
    TLR_UINT8 bSrcSAPIdx;   /* Source Service Access Point */
    TLR_UINT8 abPad[4];     /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_DATA_REPLY_REQ_T;

#define PROFIBUS_DL_DATA_REPLY_REQ_SIZE (sizeof(PROFIBUS_DL_DATA_REPLY_REQ_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Request-Packet for two-way connectionless data exchange */
typedef struct PROFIBUS_DL_PACKET_DATA_REPLY_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_REPLY_REQ_T tData;      /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DATA_REPLY_REQ_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_REPLY_REQ_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, unchanged				
	ulLen	UINT32	8 + n	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x10E	PROFIBUS_DL_CMD_DATA_REPLY_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_REPLY_REQ_T							
	bSrvCls	UINT8	0,1	Service Class (indicating the priority)				
	bDstAddr	UINT8	0-126	Destination address within the Profibus network Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bDstSAPIdx	UINT8	0-62, 64 = NIL SAP	Destination Service Access Point				
	bSrcSAPIdx	UINT8	0-62, 64 = NIL SAP	Source Service Access Point				
	abPad	UINT8[4]	0	Padding bytes				
	abSDU	UINT8[]		Service Data Unit				

Table 63: PROFIBUS_DL_CMD_DATA_REPLY_REQ - Send SRD Service Request

Additional explanations:

1. The parameter `bDstSAPIdx` defines the service access point of the remote user. It is not recommended to use the value 63 in this context. The value 64 represents the NIL SAP.
2. The parameter `bSrcSAPIdx` defines the service access point of the local user. It is not recommended to use the value 63 in this context. The value 64 represents the NIL SAP.
3. The parameter `bDstAddr` defines the FDL address of the remote station. The allowed range for this parameter extends from 0 to 126.
4. The parameter `bSrvCls` defines the FDL priority for the data transfer. Two priorities are possible in this context:
 - `PROFIBUS_DL_SERVICE_CLASS_LOW` = 0
 - `PROFIBUS_DL_SERVICE_CLASS_HIGH` = 1

Packet Structure Reference

```

typedef struct PROFIBUS_DL_DATA_REPLY_CNF_Ttag {
    TLR_UINT8 bSrvCls;      /* Service Class */
    TLR_UINT8 bDstAddr;     /* Destination address */
    TLR_UINT8 bDstSAPIdx;   /* Destination Service Access Point */
    TLR_UINT8 bSrcSAPIdx;   /* Source Service Access Point */
    TLR_UINT8 abPad[4];     /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_DATA_REPLY_CNF_T;

#define PROFIBUS_DL_DATA_REPLY_CNF_SIZE (sizeof(PROFIBUS_DL_DATA_REPLY_CNF_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Confirmation-Packet for two-way connectionless data exchange */
typedef struct PROFIBUS_DL_PACKET_DATA_REPLY_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_REPLY_CNF_T tData;      /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DATA_REPLY_CNF_T;

```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_REPLY_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	8 + n	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See Table 65: PROFIBUS_DL_CMD_DATA_REPLY_CNF - Packet Status/Error
	ulCmd	UINT32	0x10F	PROFIBUS_DL_CMD_DATA_REPLY_CNF- Command
	ulExt	UINT32	0	Extension, unchanged
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_DL_DATA_REPLY_CNF_T			
	bSrvCls	UINT8	0,1	Service Class
	bDstAddr	UINT8	0-126	Destination address Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.
	bDstSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Destination Service Access Point
	bSrcSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Source Service Access Point
	abPad	UINT8[4]	0	Padding bytes
	abSDU	UINT8[]		Service Data Unit

Table 64: PROFIBUS_DL_CMD_DATA_REPLY_CNF – Confirmation of Send SRD Service Request

Packet Status/Error

Definition / (Value)	Description
TLR_S_OK (0x00000000)	Status ok
TLR_E_PROFIBUS_DL_ACK_DL (0xC0060087L)	The remote station the service has been sent to, confirms its reception positively and has returned low priority data in the response.
TLR_E_PROFIBUS_DL_ACK_RR (0xC0060081L)	The remote station the service has been sent to indicate a Resource Error as service acknowledgment
TLR_E_PROFIBUS_DL_ACK_NA (0xC0060088L)	The remote station the service has been sent to shows no or no plausible reaction at all.
TLR_E_PROFIBUS_DL_ACK_LS (0xC006008AL)	The requested service is not activated within the local SAP configuration.
TLR_E_PROFIBUS_DL_ACK_LR (0xC006008BL)	The local resources needed to execute the requested service are not available or not sufficient.
TLR_E_PROFIBUS_DL_ACK_DS (0xC006008CL)	Master not into the logical token ring
TLR_E_PROFIBUS_DL_ACK_IV (0xC006008DL)	Invalid parameter detected in the requested service.

Table 65: PROFIBUS_DL_CMD_DATA_REPLY_CNF - Packet Status/Error

For the following possible causes of failures some hints can be given:

Error Definition / (Value)	Error source	Description of problem and necessary actions
TLR_E_PROFIBUS_DL_ACK_RR (0xC0060081L)	Slave	Some resource is not available at the remote station as indicated by the remote station, for instance the slave has no left buffer space for the requested service. <u>Action:</u> Increase buffer space or reduce amount of required buffer space.
TLR_E_PROFIBUS_DL_ACK_LR (0xC006008BL)	Slave	The local resources needed to execute the requested service are not available or not sufficient for instance the slave has no left buffer space for the requested service. <u>Action:</u> Increase buffer space or reduce amount of required buffer space.
TLR_E_PROFIBUS_DL_ACK_LS (0xC006008AL)	Slave	The requested function of master is not activated within the slave, i.e. the slave does not support the service which had been requested. <u>Action:</u> If possible, use another slave providing the requested functionality.
TLR_E_PROFIBUS_DL_ACK_DH (0xC0060086L)	No error	The slave has responded with data in a low priority telegram
TLR_E_PROFIBUS_DL_ACK_NR (0xC0060083L)	Slave	No answer-data available, as the slave did not sent back any data. <u>Action:</u> Check slave for correct operation.
TLR_E_PROFIBUS_DL_ACK_NA (0xC0060088L)	Slave	No response of the station at all <u>Action:</u> Check for correct network wiring, also check the bus address of slave or baud rate support.
TLR_E_PROFIBUS_DL_ACK_DS (0xC006008CL)	Network in general	The master is not included into the logical token ring. <u>Action:</u> Check master DP-Address or highest-station-address of other masters Examine bus wiring to avoid bus short circuits.

Table 66: Reported Errors, their Sources and Possible Actions

4.3.9 PROFIBUS_DL_CMD_DATA_REPLY_IND - Receive SRD Service Indication

This indication is sent, when the service SRD (Send and Request with Reply) has been received from another station within the Profibus network.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DATA_REPLY_IND_Ttag {
    TLR_UINT8 bSrvCls;    /* Service Class */
    TLR_UINT8 bDstAddr;   /* Destination address */
    TLR_UINT8 bDstSAPIdx; /* Destination Service Access Point */
    TLR_UINT8 bSrcAddr;   /* Source address */
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point */
    TLR_UINT8 bUpdateStatus; /* Indicates whether or not response data have been
passed */
    TLR_UINT8 abPad[2];    /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_DATA_REPLY_IND_T;

#define PROFIBUS_DL_DATA_REPLY_IND_SIZE (sizeof(PROFIBUS_DL_DATA_REPLY_IND_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Indication Packet for acknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_DATA_REPLY_IND_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DATA_REPLY_IND_T tData;
} PROFIBUS_DL_PACKET_DATA_REPLY_IND_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_REPLY_IND_T								
Type: Indication								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	8 + n	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
				Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.				
ulCmd	UINT32	0x110	PROFIBUS_DL_CMD_DATA_REPLY_IND - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_REPLY_IND_T							
	bSrvCls	UINT8	0,1	Service Class				
	bDstAddr	UINT8	0-126	Destination Address Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bDstSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Destination Service Access Point				
	bSrcAddr	UINT8	0-126	Source address Note: Mask out the highest bit when working with bDstAddr in order to avoid disturbances as this bit stores a separate information.				
	bSrcSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Source Service Access Point				
	bUpdateStatus	UINT8	0,1	Indicates whether or not response data has been passed				

	abPad	UINT8[2]	0	Padding bytes
	abSDU	UINT8[]		Service Data Unit

Table 67: PROFIBUS_DL_CMD_DATA_REPLY_IND - Receive SRD Service Indication

Additional explanations:

1. The parameters bSrcSAPIdx and bDstSAPIdx specify the source and destination service access points of the received SRD frame. The allowed range for these parameters extends from 0 to 62. The value 64 is also allowed. It represents the NIL SAP. However, a value bSrcSAPIdx of 63 is not permitted in this command because it represents the global access.
2. The parameter bSrcAddr defines the FDL address of the local station. Values between 0 and 126 are allowed.
3. The parameter bDstAddr defines the FDL address of the remote station, which has sent this service. Values between 0 and 126 are allowed.
4. The parameter bSrvCls specifies the FDL priority of the received SRD frame. Two priorities are possible in this context:
 - PROFIBUS_DL_SERVICE_CLASS_LOW = 0
 - PROFIBUS_DL_SERVICE_CLASS_HIGH = 1
5. The parameter bUpdateStatus indicates whether or not the response data have been passed to the local FDL controller. The response data can be set up to with the reply update service (PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_REQ).

The following table shows the allowed values of the parameter bUpdateStatus and their meanings:

Name	Value	Meaning
PROFIBUS_DL_UPDATE_STATUS_NO	0x00	No data transmitted in response
PROFIBUS_DL_UPDATE_STATUS_LO	0x01	Low priority data transmitted in response
PROFIBUS_DL_UPDATE_STATUS_HI	0x02	High priority data transmitted in response

Table 68: Allowed Values and their Meanings for Variable bUpdateStatus

4.3.10 PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_REQ/CNF - Reply Update Service

The service is used to update the reply buffer of the SRD service (Send and Request with Reply).

The host is responsible for valid data in the device if any data should be requested from a remote station issuing an SRD or MSRD service request. By this command, the host may initiate the loading of this data to a specific SAP. Upon completion of loading the data area, the device informs the host by the confirmation message.

This packet is only appropriate if the intended role in service (value of the `bRoleInService` parameter) is not "Initiator" (`PROFIBUS_DL_SERVICE_ROLE_INITIATOR`). In this special case you are recommended to use the packet "PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ/CNF - Activate an SAP for Request" described in the previous section.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_REPLY_UPDATE_REQ_Ttag {
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point */
    TLR_UINT8 bSrvCls;    /* Service Class */
    TLR_UINT8 bTransmitStrategy; /* One shot or multiple transmissions */
    TLR_UINT8 abPad[1]; /* Padding needed to bring SDU to a DWORD boundary and to
right position */
    TLR_UINT8 abSDU[PROFIBUS_DL_MAX_DLPDU_SIZE]; /* Service Data Unit */
} PROFIBUS_DL_REPLY_UPDATE_REQ_T;

#define PROFIBUS_DL_REPLY_UPDATE_REQ_SIZE (sizeof(PROFIBUS_DL_REPLY_UPDATE_REQ_T)-
PROFIBUS_DL_MAX_DLPDU_SIZE)

/* Request-Packet for updating the reply update buffer of a two-way connectionless
data exchange */
typedef struct PROFIBUS_DL_PACKET_REPLY_UPDATE_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_REPLY_UPDATE_REQ_T tData; /* Packet Data Unit */
} PROFIBUS_DL_PACKET_REPLY_UPDATE_REQ_T;
```


Packet Description

structure PROFIBUS_DL_PACKET_DATA_REPLY_UPDATE_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	4 + n	Packet data length in bytes
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task
			Hexadecimal Value	Definition Description
			0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.			
	ulCmd	UINT32	0x112	PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_REQ - Command
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)	
ulRout	UINT32	x	Routing, do not change	
tData	structure PROFIBUS_DL_DATA_REPLY_UPDATE_REQ_T			
	bSrcSAPIdx	UINT8	0-62, 64 (= NIL SAP)	Source Service Access Point
	bSrvCls	UINT8	0,1	Service Class
	bTransmitStrategy	UINT8	0,1	One shot or multiple transmissions
	abPad	UINT8[1]	0	Padding bytes
	abSDU	UINT8[]		Service Data Unit

Table 69: PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_REQ - Reply Update Service

Additional explanations:

1. The parameter `bSrcSAPIdx` specifies the service access point of the remote user that carries out this request and which data area should be updated by the `abSDU`. A `bSrcSAPIdx` value of 63 is not permitted. A value of 64 represents the NIL SAP.
2. The parameter `bSrvCls` defines the FDL priority for the data transfer. Two priorities are possible:
 - `PROFIBUS_DL_SERVICE_CLASS_LOW` = 0 and
 - `PROFIBUS_DL_SERVICE_CLASS_HIGH` = 1.
3. The parameter `bTransmitStrategy` specifies whether the update is transmitted only once or many times, in the case of "multiple" the data area is transferred again for each subsequent SRD service. This is done by choosing between
 - `PROFIBUS_DL_REPLY_UPDATE_SINGLE` = 0 and
 - `PROFIBUS_DL_REPLY_UPDATE_MULTIPLE` = 1

Packet Structure Reference

```
typedef struct PROFIBUS_DL_REPLY_UPDATE_CNF_Ttag {
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point */
    TLR_UINT8 bSrvCls;    /* Service Class */
} PROFIBUS_DL_REPLY_UPDATE_CNF_T;

/* Confirmation-Packet for after updating the reply update buffer of a two-way
connectionless data exchange */
typedef struct PROFIBUS_DL_PACKET_REPLY_UPDATE_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_REPLY_UPDATE_CNF_T tData; /* Packet Data Unit */
} PROFIBUS_DL_PACKET_REPLY_UPDATE_CNF_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DATA_REPLY_UPDATE_CNF_T								
Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	2	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.							
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x113	PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DATA_REPLY_UPDATE_CNF_T							
	bSrcSAPIdx	UINT8	0-62, 64 = NIL SAP	Source Service Access Point				
	bSrvCls	UINT8	0,1	Service Class				

Table 70: PROFIBUS_DL_CMD_DATA_REPLY_UPDATE_CNF – Confirmation of Reply Update Service

4.3.11 PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ/CNF - Activate an SAP for Request

This packet allows configuring and activating a local service access point.

A service access point is a means for managing communication between different communication layers correctly, it is identified by a non-negative integer numeric value. You can find a detailed description in section 3.4.2 “*Management of Services Access Points*” on page 26 of this document.

This packet provides the host with the possibility to configure a local service access point (LSAP) for most of the individual FDL services. However, there is a special case requiring additional information if responding functionality is needed for services with integrated reply capabilities such as SRD and related ones. In this case the packet described here is not applicable, so there is a separate packet for configuring and activating the responder function for the Reply services (SRD, MSRD) that are activated by means of the RSAP activate service, see next the section

The parameter `bSrcSAPIdx` specifies the local LSAP that is to be activated and configured. The allowed values are 0 to 63 and NIL is represented by the value 64.

After the SAP has successfully been configured and activated, a confirmation packet containing the numerical value of the SAP (contained in variable `bSrcSAPIdx`) and the access protection mode will be sent back to the application. After that, the SAP may be used in FDL communication function such as the ones described in the sections 4.3.4, 4.3.6 or 4.3.8. If later on this SAP is not needed any longer, it may be deactivated using the packet “SAP Deactivate” described in section 3.4.2.4 „SAP Deactivate“ of this manual.

If a SAP should be configured for a responder service (i.e. the SRD or the MSRD service), this packet is only appropriate if the intended role in service (represented by the value of the `bRoleInService` parameter) is “Initiator” (PROFIBUS_DL_SERVICE_ROLE_INITIATOR). Otherwise use the packet “[PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ](#)” described in the next section.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DLSAP_ACTIVATE_REQ_Ttag {
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point to be activated */
    TLR_UINT8 bAcc; /* access protection to specify if all stations or
individual station may access */
    TLR_UINT8 bParallelServices; /* Number of parallel Services that shall be
activated the same time */
    TLR_UINT8 bServiceActivate; /* services to be activated for that SAPIdx
SDA, SDN, SRD, MSRD, CS */
    TLR_UINT8 bRoleInService; /* service configuration INITIATOR, RESPONDER, BOTH */
    TLR_UINT8 bMaxDLSDUlenReqLow; /* DLSDU length list */
    TLR_UINT8 bMaxDLSDUlenReqHigh;
    TLR_UINT8 bMaxDLSDUlenIndCnfLow;
    TLR_UINT8 bMaxDLSDUlenIndCnfHigh;
} PROFIBUS_DL_DLSAP_ACTIVATE_REQ_T;

typedef struct PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DLSAP_ACTIVATE_REQ_T tData;
} PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_REQ_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_REQ_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	9	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.							
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x120	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DLSAP_ACTIVATE_REQ_T							
	bSrcSAPIdx	UINT8	0-63, 64=NIL SAP	Source Service Access Point to be activated				
	bAcc	UINT8	0-126,255	Access protection to specify if all stations or individual station may access				
	bParallelServices	UINT8	1-4	Number of parallel Services that shall be activated the same time				
	bServiceActivate	UINT8	0-31	Services to be activated for that SAPIdx SDA,SDN,SRD,MSRD,CS				
	bRoleInService	UINT8	1-3	Service configuration INITIATOR, RESPONDER,BOTH				
	bMaxDLSDULenReqLow	UINT8	1-246	Maximum length of DLSDU list for requests with low priority				
	bMaxDLSDULenReqHigh	UINT8	1-246	Maximum length of DLSDU list for requests with high priority				

	bMaxDLSDULenI ndCnfLow	UINT8	1-246	Maximum length of DLSDU list for indications and confirmations with low priority
	bMaxDLSDULenI ndCnfHigh	UINT8	1-246	Maximum length of DLSDU list for indications and confirmations with high priority

Table 71: PROFIBUS_DL_CMD_DLSDAP_ACTIVATE_REQ - Activate a SAP for Request

Additional explanations:

- 1) The parameter `bSrcSAPIdx` specifies the source service access point of the received SRD frame. The allowed range for this parameter extends from 0 to 62. The value 64 is also allowed. It represents the NIL SAP. A value `bSrcSAPIdx` of 63 is not permitted in this command because it represents the global access.
- 2) The parameter `bAcc` is used for specifying the access protection for responder functions. It has the allowed values ALL = 255 or 0 to 126 and specifies whether all remote stations (ALL) or only an individual remote station identified by its station number (0 to 126) may have access to the LSAP. This parameter is only valid for responder functions, i.e. `bRoleInService` = RESPONDER, BOTH. If the access value is ALL, the device automatically sets the buffer length to the maximum value 244 and the services to all supported and the role in service to both independent what the rest command message parameter are.
- 3) The parameter `bParallelServices` specifies the upper limit for the number of parallel services that can be activated at the same time. Up to 4 parallel services are possible.
- 4) The parameter `bServiceActivate` contains the FDL service that shall be activated for this service according to the following table:

`bServiceActivate` parameter

Name	Value	Service
PROFIBUS_DL_SERVICE_SDA	0x01	SDA Service.
PROFIBUS_DL_SERVICE_SDN	0x02	SDN Service.
PROFIBUS_DL_SERVICE_SRD	0x04	SRD Service
PROFIBUS_DL_SERVICE_MSRD	0x08	MSRD Service
PROFIBUS_DL_SERVICE_CS	0x10	CS Service

Table 72: Values of `bServiceActivate` Parameter and the according Service

- 5) The parameter `bRoleInService` specifies the configuration for the service to be activated. The following values are specified:

`bRoleInService` parameter

Name	Value	Operation
PROFIBUS_DL_SERVICE_ROLE_INITIATOR	1	The device initiates the respective service exclusively.
PROFIBUS_DL_SERVICE_ROLE_RESPONDER	2	The device responds to the service exclusively. Not allowed for SRD.
PROFIBUS_DL_SERVICE_ROLE_BOTH	3	The device initiates and responds to the service. This option is not permissible for SRD.

Table 73: Values of `bRoleInService` Parameter and the according Operation

- 6) The parameters `bMaxDLSDULenReqLow`, `bMaxDLSDULenReqHigh`, `bMaxDLSDULenIndCnfLow` and `bMaxDLSDULenIndCnfHigh` specify the maximum length of the SDU for transparent data transfer depending on
- whether low (`bMaxDLSDULenXXXLow`) or high priority (`bMaxDLSDULenXXXHigh`) is intended to be applied.
 - whether data are sent (in case of a request, `bMaxDLSDULenReqYYY`) or received (in case of either a confirmation or an indication, `bMaxDLSDULenIndCnfYYY`)

Additional rules apply for all of these parameters whether they must be equal to 0 or must not be equal to 0 depending on :

- the chosen service
- the role in service (Initiator, responder or both))

These rules can be expressed as tables. For each parameter and role in service intended to use in conjunction with this parameter there are a few allowed combinations whether to set the parameter to a zero- or non-zero-value which are represented by columns. If for a parameter the value 0 can be found it should be set to 0 otherwise (value 'x') it should be set to a non-zero value not exceeding 242:

`bMaxDLSDULenReqLow`, `bMaxDLSDULenReqHigh`, `bMaxDLSDULenIndCnfLow` and `bMaxDLSDULenIndCnfHigh` parameters for services SDA and SDN

Services SDA and SDN															
Name	Role in service														
	Initiator			Responder			Both								
<code>bMaxDLSDULenReqLow</code>	x	0	x	0	0	0	x	0	x	0	x	0	x	x	x
<code>bMaxDLSDULenReqHigh</code>	0	x	x	0	0	0	0	x	0	x	0	x	x	x	x
<code>bMaxDLSDULenIndCnfLow</code>	0	0	0	x	0	x	x	0	0	x	x	x	x	0	x
<code>bMaxDLSDULenIndCnfHigh</code>	0	0	0	0	x	x	0	x	x	0	x	x	0	x	x

Table 74: Allowed combinations of `bMaxDLSDULenReqLow`, `bMaxDLSDULenReqHigh`, `bMaxDLSDULenIndCnfLow` and `bMaxDLSDULenIndCnfHigh` parameters for services SDA and SDN

`bMaxDLSDULenReqLow`, `bMaxDLSDULenReqHigh`, `bMaxDLSDULenIndCnfLow` and `bMaxDLSDULenIndCnfHigh` parameters for services SRD and MSRD

Services SRD and MSRD													
Name	Role in service												
	Initiator												
<code>bMaxDLSDULenReqLow</code>	0	0	0	x	0	0	x	x	x	x	0	x	
<code>bMaxDLSDULenReqHigh</code>	0	0	0	0	x	x	0	x	x	0	x	x	
<code>bMaxDLSDULenIndCnfLow</code>	x	0	x	x	0	x	0	x	0	x	x	x	
<code>bMaxDLSDULenIndCnfHigh</code>	0	x	x	0	x	0	x	0	x	x	x	x	

Table 75: Allowed combinations of `bMaxDLSDULenReqLow`, `bMaxDLSDULenReqHigh`, `bMaxDLSDULenIndCnfLow` and `bMaxDLSDULenIndCnfHigh` parameters for services SRD and MSRD

For the roles in service *Responder* and *Both* no values are given as this packet is not applicable for the reply services SRD and MSRD, there is a special packet, please see the next section

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DLSAP_ACTIVATE_CNF_Ttag {
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point to be activated */
    TLR_UINT8 bAcc;       /* access protection to specify if all stations or
individual station may access */
} PROFIBUS_DL_DLSAP_ACTIVATE_CNF_T;

/* Confirmation-Packet for SAP activation */
typedef struct PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DLSAP_ACTIVATE_CNF_T tData; /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_CNF_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_CNF_T								
Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	2	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
				Hexadecimal Value	Definition Description			
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.								
ulCmd	UINT32	0x121	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DLSAP_ACTIVATE_CNF_T							
	bSrcSAPIdx	UINT8	0-63,64	Source Service Access Point to be activated				
	bAcc	UINT8	0-126,255	Access protection to specify if all stations or individual station may access				

Table 76: PROFIBUS_DL_CMD_DLSAP_ACTIVATE_CNF - Confirmation of Activate a SAP for Request

4.3.12 PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ/CNF - Activate an SAP for Responder Functionality

This packet allows configuring a local service access point for an SRD or MSRD service in a responder role. As outlined in section 3.4.2 “*Management of Services Access Points*”, this special case requires a separate packet, namely this one.

Have in mind that this packet is only appropriate if the intended role in service (represented by the value of the `bRoleInService` parameter) is not “Initiator” (`PROFIBUS_DL_SERVICE_ROLE_INITIATOR`). Otherwise use the packet “PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ/CNF - Activate an SAP for Request” described in the previous section.

For a more precise description of service access points refer to the section 3.4.2 “*Management of Services Access Points*” of this manual.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_REQ_Ttag {
    TLR_UINT8 bSrcSAPIdx;          /* Source Service Access Point to be activated */
    TLR_UINT8 bAcc;                /* access protection to specify if all
                                   stations or individual station may access */
    TLR_UINT8 bParallelServices; /* Number of parallel Services that shall be
                                   activated the same time */
    TLR_UINT8 bMaxDLSDUlenReqLow; /* DLSDU length list */
    TLR_UINT8 bMaxDLSDUlenReqHigh;
    TLR_UINT8 bMaxDLSDUlenIndLow;
    TLR_UINT8 bMaxDLSDUlenIndHigh;
    TLR_UINT8 bIndicationMode;
    TLR_BOOLEAN8 fPublisherEnabled;
} PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_REQ_T;

typedef struct PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_RESPONDER_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_REQ_T tData;
} PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_RESPONDER_REQ_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_RESPONDER_REQ_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	9	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.							
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
	ulCmd	UINT32	0x122	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ - Command				
	ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)				
	ulRout	UINT32	x	Routing, do not change				
tData	structure PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_REQ_T							
	bSrcSAPIdx	UINT8	0-63, 64 (=NIL SAP)	Source Service Access Point to be activated				
	bAcc	UINT8	0-126,255	access protection to specify if all stations or individual station may access				
	bParallelServices	UINT8	1-4	Number of parallel Services that shall be activated the same time				
	bMaxDLSDULenReqLow	UINT8	1-246	DLSDU length list				
	bMaxDLSDULenReqHigh	UINT8	1-246	DLSDU length list				
	bMaxDLSDULenIndLow	UINT8	1-246	DLSDU length list				
	bMaxDLSDULenIndHigh	UINT8	1-246	DLSDU length list				
	bIndicationMode	UINT8	0-2	Indication mode				

	fPublisherEnabled	BOOLEAN	0,1	Enable or disable publisher functionality
--	-------------------	---------	-----	---

Table 77: PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_REQ - Activate a SAP for Responder Functionality

Additional explanations:

1. The parameter bSrcSAPIdx specifies the local LSAP for the data area and the user that receives the indication primitive if a SRD is received on this "DSAP". The value can have a range from 0 to 63 and NIL is represented by 64.
2. The parameter bAcc with the values ALL = 255 or 0 to 126 is used for access protection and specifies whether all remote stations (all) or only an individual remote station (0 to 126) may have access to the LSAP. This parameter is only valid for responder functions, i.e. bRoleInService = RESPONDER or BOTH.
3. The parameter bParallelServices specifies the upper limit for the number of parallel services that can be activated at the same time. Up to 4 parallel services are possible.
4. The parameters bMaxDLSDULenReqLow, bMaxDLSDULenReqHigh, bMaxDLSDULenIndLow and bMaxDLSDULenIndHigh specify the maximum length of the SDU for transparent data transfer depending on
 - whether low (bMaxDLSDULenXXXLow) or high priority (bMaxDLSDULenXXXHigh) is intended to be applied.
 - whether data are sent (in case of a request, bMaxDLSDULenReqYYY) or received (in case of either a confirmation or an indication, bMaxDLSDULenIndYYY)
 - Additional rules apply for all of these parameters whether they must be equal to 0 or must not be equal to 0. These rules can be expressed as tables. For each parameter and role in service intended to use in conjunction with this parameter there are a few allowed combinations whether to set the parameter to a zero- or non-zero-value which are represented by columns. If for a parameter the value 0 can be found it should be set to 0 otherwise (value 'x') it should be set to a non-zero value not exceeding 242:

bMaxDLSDULenReqLow, bMaxDLSDULenReqHigh, bMaxDLSDULenIndLow and bMaxDLSDULenIndHigh parameters for services SRD and MSRD in 'Responder' role

Services SRD and MSRD												
Name	Role in service											
	Responder or both											
bMaxDLSDULenReqLow	x	0	x	x	0	0	x	x	x	x	0	x
bMaxDLSDULenReqHigh	0	x	x	0	x	x	0	x	x	0	x	x
bMaxDLSDULenIndLow	0	0	0	x	0	x	0	x	0	x	x	x
bMaxDLSDULenIndHigh	0	0	0	0	x	0	x	0	x	x	x	x

Table 78: Allowed combinations of bMaxDLSDULenReqLow, bMaxDLSDULenReqHigh, bMaxDLSDULenIndLow and bMaxDLSDULenIndHigh parameters for service SRD with role in service 'Responder'.

5. The parameter `bIndicationMode` specifies when to create an indication at the receiver of the packet. The following table shows the allowed values of the parameter `bUpdateStatus` and their meanings:

Name	Value	Meaning
PROFIBUS_DL_INDICATION_MODE_ALL	0x00	An indication is generated always, even in case of zero data length.
PROFIBUS_DL_INDICATION_MODE_DATA	0x01	An indication is generated if data transfer really happens, i.e. only in case of non-zero data length of at least either the request or the reply data.
PROFIBUS_DL_INDICATION_MODE_UNCHANGED	0x02	This option is used only for changing the <code>bAcc</code> parameter. It is used in the following manner: <ul style="list-style-type: none"> ■ The <code>bAcc</code> parameter will be changed according to its setting within the packet. ■ All other parameters will completely remain unchanged.

Table 79: Allowed Values and their Meanings for Variable `bUpdateStatus`

- The parameter `fPublisherEnabled` specifies the reply update mode, i.e. whether or not the service using the RSAP may act as publisher or not. The possible values in this context are TRUE(1) or FALSE.(0). However, this parameter is currently not supported.

Packet Structure Reference

```
typedef struct PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_CNF_Ttag {
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point to be activated */
    TLR_UINT8 bAcc;       /* access protection to specify if all stations or
individual station may access */
} PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_CNF_T;

/* Request-Packet for SAP responder activation */
typedef struct PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_RESPONDER_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_CNF_T tData; /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_RESPONDER_CNF_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DLSAP_ACTIVATE_RESPONDER_CNF_T								
Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	2	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x123	PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DLSAP_ACTIVATE_RESPONDER_CNF_T							
	bSrcSAPIdx	UINT8	0-63. 64 = NIL SAP	Source Service Access Point to be activated				
	bAcc	UINT8	0-126,255	access protection to specify if all stations or individual station may access				

Table 80: PROFIBUS_DL_CMD_DLSAP_ACTIVATE_RESPONDER_CNF – Confirmation of Activate a SAP for Responder Functionality

4.3.13 PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ/CNF - Deactivate an SAP for Request

This service is used to deactivate a local service access point which has previously been defined either with the

- “PROFIBUS_DL_CMD_DLSAP_ACTIVATE_REQ/CNF - Activate an SAP for Request” packet described in section 4.3.11 or with the
- “PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ/CNF - Deactivate an SAP for Request” packet described in section 4.3.12.

Only the numerical value assigned with the SAP or the RSAP to be deactivated needs to be specified.

Packet Structure Reference

```
/* Request-Packet for SAP deactivation */
typedef struct PROFIBUS_DL_DLSAP_DEACTIVATE_REQ_Ttag {
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point to be activated */
} PROFIBUS_DL_DLSAP_DEACTIVATE_REQ_T;

typedef struct PROFIBUS_DL_PACKET_DLSAP_DEACTIVATE_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DLSAP_DEACTIVATE_REQ_T tData;
} PROFIBUS_DL_PACKET_DLSAP_DEACTIVATE_REQ_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_DLSAP_DEACTIVATE_REQ_T;								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	1	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
				Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.				
ulCmd	UINT32	0x128	PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DLSAP_DEACTIVATE_REQ_T							
	bSrcSAPIdx	UINT8	0-63, 64	Source Service Access Point to be deactivated				

Table 81: PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_REQ - Deactivate an SAP for Request

Packet Structure Reference

```

typedef struct PROFIBUS_DL_DLSAP_DEACTIVATE_CNF_Ttag {
    TLR_UINT8 bSrcSAPIdx; /* Source Service Access Point to be activated */
} PROFIBUS_DL_DLSAP_DEACTIVATE_CNF_T;

/* Confirmation-Packet for SAP deactivation */
typedef struct PROFIBUS_DL_PACKET_DLSAP_DEACTIVATE_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_DLSAP_DEACTIVATE_CNF_T tData; /* Packet Data Unit */
} PROFIBUS_DL_PACKET_DLSAP_DEACTIVATE_CNF_T;

```

Packet Description

structure PROFIBUS_DL_PACKET_DLSAP_DEACTIVATE_CNF_T								
Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	1	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x129	PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_CNF - Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_DLSAP_DEACTIVATE_CNF_T							
	bSrcSAPIdx	UINT8	0-63, 64	Source Service Access Point which has been deactivated in case of successful execution				

Table 82: PROFIBUS_DL_CMD_DLSAP_DEACTIVATE_CNF – Confirmation of Deactivate an SAP for Request

4.3.14 PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ/CNF - Load the Bus Parameter Set

This packet can be applied to load a bus parameter set to the communication channel. A detailed description of all bus parameters is given in section 2.1.1 “*Detailed Description of Bus Parameters*” of this document.

A data structure containing the configured bus and master parameter set is returned along with the confirmation packet.



Note: This packet belongs to Layer 2 (Data Link Layer) of the OSI-layer model for networks.



Important: All parameters of structure PROFIBUS_DL_BUS_PARAMETER_SET_T following after parameter bHSA are not relevant for PROFIBUS MPI and should therefore be set to 0.

Packet Structure Reference

```
#define BIG_ENDIAN      0
#define LITTLE_ENDIAN  1

typedef struct PROFIBUS_DL_PACKET_SET_BUS_PARAMETER_SET_REQ_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_BUS_PARAMETER_SET_T tData;
} PROFIBUS_DL_PACKET_SET_BUS_PARAMETER_SET_REQ_T;

typedef struct PROFIBUS_DL_BUS_PARAMETER_SET_Ttag {
    TLR_UINT16 usBus_Para_Len; /* Contains the length of the Bus_Para inclusive the
field Bus_Para_Len itself */
    TLR_UINT8  bDL_Add; /* Contains the own address of the PROFIBUS Device */
    TLR_UINT8  bData_rate; /* Contains the Transmission speed */
    TLR_UINT16 usTSL; /* slot-time */
    TLR_UINT16 usMin_TSDR; /* min. station delay responder */
    TLR_UINT16 usMax_TSDR; /* max. station delay responder */
    TLR_UINT8  btQUI; /* quiet time */
    TLR_UINT8  btSET; /* setup time */
    TLR_UINT32 ultTR; /* target rotation time */
    TLR_UINT8  bG; /* Gap update factor */
    TLR_UINT8  bHSA; /* Highest station address */
    TLR_UINT8  bMax_Retry_Limit; /* retries if error occurs */
    TLR_UINT8  bBp_Flag;
    TLR_UINT16 usMin_Slave_Interval; /* Minimum Slave Interval Time */
    TLR_UINT16 usPoll_Timeout; /* Class2 Poll timeout */
    TLR_UINT16 usData_Control_Time; /* Data Control Time */
    TLR_UINT8  bMasterSetting; /* 0 == big Endian, 1 == little Endian (swapping
should be done) */
    TLR_UINT8  bMax_User_Global_Control; /* Maximum allowed parallel active USER
Global Control Commands */
    TLR_UINT8  abReserved[4]; /* 4 reserved Octets */
    TLR_UINT16 usMaster_User_Data_Len; /* Contains the length of the USER Master data
*/
    TLR_UINT8  abMaster_Class2_Name[32]; /* Name of the Master */
    TLR_UINT8  abMaster_User_Data[32]; /* USER specific Parameter data */
    TLR_UINT32 ultCL; /* Isochronous cycle time */
    TLR_UINT8  bMax_TSH; /* Maximum Shift Time */
} PROFIBUS_DL_BUS_PARAMETER_SET_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_SET_BUS_PARAMETER_SET_REQ_T					
Type: Request					
Area	Variable	Type	Value / Range	Description	
tHead	structure TLR_PACKET_HEADER_T				
	ulDest	UINT32		Destination queue handle	
	ulSrc	UINT32		Source queue handle	
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet	
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process	
	ulLen	UINT32	>=19	Packet data length in bytes	
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet	
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task	
			Hexadecimal Value	Definition Description	
			0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.	
Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x126	PROFIBUS_DL_CMD_SET_BUS_PARAMETER_SET_REQ - Command		
ulExt	UINT32	0	Extension, unchanged		
ulRout	UINT32	x	Routing, do not change		
tData	structure PROFIBUS_DL_BUS_PARAMETER_SET_T				
	usBus_Para_Len	UINT16	35	Contains the length of the Bus_Para including the field usBus_Para_Len itself sizeof(PROFIBUS_DL_BUS_PARAMETER_SET_T)	
	Note: This parameter is not evaluated by the PROFIBUS DP Master firmware. It is only present for compatibility reasons.				
	bDL_Add	UINT8	0-126	Contains the own address of the PROFIBUS Device	
	bData_rate	UINT8	0-11 Default: 3	Contains the Transmission speed (see Table 4: Supported Baud Rates) The value 3 denotes a baudrate of 187.5 kBit/s which is normally used at PROFIBUS MPI.	
	usTSL	UINT16	37-16383 Default: 415	Slot-time Use the default value of 415 when working with the default baudrate of 187.5 kBit/s	
usMin_TSDR	UINT16	1-1023	Min. station delay responder		

			Default: 60	
	usMax_TSDR	UINT16	1-1023 Default: 400	Max. station delay responder
	bTQUI	UINT8	0-127 Default: 1	Quiet time
	bTSET	UINT8	1-255 Default: 1	Setup time
	uITTR	UINT32	>= 255 Default: 10000	Target rotation time
	bG	UINT8	1-255 Default: 20	Gap update factor
	bHSA	UINT8	1-126 Default: 31	Highest station address
	bMax_Retry_Limit	UINT8	0	Retries if error occurs
	bBp_Flag	UINT8	0	Bp flag
	usMin_Slave_Interval	UINT16	0	Minimum Slave Interval Time
	usPoll_Timeout	UINT16	0	Class2 Poll timeout
	usData_Control_Time	UINT16	0	Data Control Time
	bMasterSetting	UINT8	0,1	Master Setting 0 == BIG_ENDIAN, 1 == LITTLE_ENDIAN (swapping required)
	bMax_User_Global_Control	UINT8	0	Maximum allowed parallel active USER Global Control Commands
	abReserved	UINT8[4]	0	4 reserved Octets
	usMaster_User_Data_Len	UINT16	0	Contains the length of the USER Master data
	abMaster_Class2_Name[32]	UINT8[32]	0	Name of the Master
	abMaster_User_Data[32];	UINT8[32]	0	USER specific Parameter data
	uITCL	UINT32	0	Isochronous cycle time
	bMax_TSH	UINT8	0	Maximum Shift Time

Table 83: PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_REQ - Load the Bus Parameter Set

Packet Structure Reference

```

/* Confirmation-Packet for the setting the DL parameter */
typedef struct PROFIBUS_DL_PACKET_SET_BUS_PARAMETER_SET_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_BUS_PARAMETER_SET_T tData;
} PROFIBUS_DL_PACKET_SET_BUS_PARAMETER_SET_CNF_T;

typedef struct PROFIBUS_DL_BUS_PARAMETER_SET_Ttag {
    TLR_UINT16 usBus_Para_Len; /* Contains the length of the Bus_Para inclusive the
field Bus_Para_Len itself */
    TLR_UINT8 bDL_Add; /* Contains the own address of the PROFIBUS Device */
    TLR_UINT8 bData_rate; /* Contains the Transmission speed */
    TLR_UINT16 usTSL; /* slot-time */
    TLR_UINT16 usMin_TSDR; /* min. station delay responder */
    TLR_UINT16 usMax_TSDR; /* max. station delay responder */
    TLR_UINT8 btQUI; /* quite time */
    TLR_UINT8 btSET; /* setup time */
    TLR_UINT32 ulTTR; /* target rotation time */
    TLR_UINT8 bG; /* Gap update factor */
    TLR_UINT8 bHSA; /* Highest station address */
    TLR_UINT8 bMax_Retry_Limit; /* retries if error occurs */
    TLR_UINT8 bBp_Flag;
    TLR_UINT16 usMin_Slave_Interval; /* Minimum Slave Interval Time */
    TLR_UINT16 usPoll_Timeout; /* Class2 Poll timeout */
    TLR_UINT16 usData_Control_Time; /* Data Control Time */
    TLR_UINT8 bMasterSetting; /* 0 == big Endian, 1 == little Endian (swapping
should be done)*/
    TLR_UINT8 bMax_User_Global_Control; /* Maximum allowed parallel active USER
Global Control Commands */
    TLR_UINT8 abReserved[4]; /* 4 reserved Octets */
    TLR_UINT16 usMaster_User_Data_Len; /* Contains the length of the USER Master data
*/
    TLR_UINT8 abMaster_Class2_Name[32]; /* Name of the Master */
    TLR_UINT8 abMaster_User_Data[32]; /* USER specific Parameter data */
    TLR_UINT32 ulTCL; /* Isochronous cycle time */
    TLR_UINT8 bMax_TSH; /* Maximum Shift Time */
} PROFIBUS_DL_BUS_PARAMETER_SET_T;

```

Packet Description

structure PROFIBUS_DL_PACKET_SET_VALUE_BUS_PARAMETER_SET_CNF_T								
Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	103	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
				Hexadecimal Value	Definition Description			
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.								
ulCmd	UINT32	0x127	PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_CNF - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_BUS_PARAMETER_SET_T							
	usBus_Para_Len	UINT16	35	Contains the length of the Bus_Para including the field usBus_Para_Len itself				
	bDL_Add	UINT8	0-126	Contains the own address of the PROFIBUS Device				
	bData_rate	UINT8	0-11	Contains the Transmission speed				
	usTSL	UINT16	37-16383	slot-time				
	usMin_TSDR	UINT16	1-1023	min. station delay responder				
	usMax_TSDR	UINT16	1-1023	Max. station delay responder				
	bTQUI	UINT8	0-127	quiet time				
	bTSET	UINT8	1-255	setup time				
	ulTTR	UINT32	>= 255	target rotation time				
	bG	UINT8	1-255	GAP update factor				
	bHSA	UINT8	1-126	Highest station address				
	bMax_Retry_Limit	UINT8	0	retries if error occurs				
	bBp_Flag	UINT8	0					

usMin_Slave_Int erval	UINT16	0	Minimum Slave Interval Time
usPoll_Timeout	UINT16	0	Class2 Poll timeout
usData_Control_ Time	UINT16	0	Data Control Time
bMasterSetting	UINT8	0,1	Master Setting 0: big endian 1: little endian (swapping required)
bMax_User_Glob al_Control	UINT8	0	Maximum allowed parallel active USER Global Control Commands
abReserved	UINT8[4]	0	4 reserved Octets
usMaster_User_ Data_Len	UINT16	0	Contains the length of the USER Master data
abMaster_Class2 _Name[32]	UINT8[32]	0	Name of the Master
abMaster_User_ Data[32];	UINT8[32]	0	USER specific Parameter data
uITCL	UINT32	0	Isochronous cycle time
bMax_TSH	UINT8	0	Maximum Shift Time

Table 84: PROFIBUS_DL_CMD_SET_VALUE_BUS_PARAMETER_SET_CNF – Confirmation of Loading the Bus Parameter Set

4.3.15 PROFIBUS_DL_CMD_SET_VALUE_REQ/CNF - Set Value Service

This service sets a specific value within the variable set of the DL Layer.

The following system variables are available and supported by this packet:

1. Bus parameter

- minT_{SDR}

Indicates the smallest station delay response time. See section “usMin_TSDR” for detailed information. Allowed values range from 1 to 65535 ($= 2^{16}-1$).

2. Own Address

- DL_ADDR

Indicates the own address of the Profibus DP master that can be changed only by this method.

Packet Structure Reference

```
#define PROFIBUS_DL_SET_VALUE_MINTDSR (0)
#define PROFIBUS_DL_SET_VALUE_DLADDR (1)

typedef struct PROFIBUS_DL_SET_VALUE_REQ_Ttag
{
    TLR_UINT8 bvlu; /* Value to be set */
    union {
        TLR_UINT16 usMin_TSDR;
        TLR_UINT8 bDl_Add;
    } un;
}
PROFIBUS_DL_SET_VALUE_REQ_T;

/* Request-Packet for setting a DL value */
typedef struct PROFIBUS_DL_PACKET_SET_VALUE_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_SET_VALUE_REQ_T tData;
}
PROFIBUS_DL_PACKET_SET_VALUE_REQ_T;
```


Packet Description

structure PROFIBUS_DL_PACKET_SET_VALUE_REQ_T								
Type: Request								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle				
	ulSrc	UINT32		Source queue handle				
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet				
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process				
	ulLen	UINT32	2	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
			Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.					
ulCmd	UINT32	0x12A	PROFIBUS_DL_CMD_SET_VALUE_REQ - Command					
ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)					
ulRout	UINT32	x	Routing, do not change					
tData	structure PROFIBUS_DL_SET_VALUE_REQ_T							
	eVlu	ENUM	0,1	Value to be set: 0: MINTDSR 1: DLADDR				

Table 85: PROFIBUS_DL_CMD_SET_VALUE_REQ - Set Value Service

Packet Structure Reference

```
/* Confirmation-Packet for setting a DL value */
typedef struct PROFIBUS_DL_PACKET_SET_VALUE_CNF_Ttag {
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_SET_VALUE_CNF_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_SET_VALUE_CNF_T								
Type: Confirmation								
Area	Variable	Type	Value / Range	Description				
tHead	structure TLR_PACKET_HEADER_T							
	ulDest	UINT32		Destination queue handle, unchanged				
	ulSrc	UINT32		Source queue handle, unchanged				
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, unchanged				
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, unchanged				
	ulLen	UINT32	0	Packet data length in bytes				
	ulId	UINT32	0 ... 2 ³² -1	Packet identification as unique number generated by the Source process of the packet				
	ulSta	UINT32		See section „Error Codes of the MPI AP-Task				
				<table><tr><th>Hexadecimal Value</th><th>Definition Description</th></tr><tr><td>0xC0680001L</td><td>TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.</td></tr></table>	Hexadecimal Value	Definition Description	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.
	Hexadecimal Value	Definition Description						
	0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.						
				Table 99: Error Codes of the MPI AP-Task Error Codes of the DL-Task“.				
ulCmd	UINT32	0x12B	PROFIBUS_DL_CMD_SET_VALUE_CNF – Command					
ulExt	UINT32	0	Extension, unchanged					
ulRout	UINT32	x	Routing, do not change					

Table 86: PROFIBUS_DL_CMD_SET_VALUE_CNF – Confirmation of Set Value Service

4.3.16 PROFIBUS_DL_CMD_SEND_FDL_STATUS_REQ - Send FDL Status

This packet is used to request the FDL status (and station type, PROFIBUS Master or Slave) from any station on the PROFIBUS network. The FDL status indicates whether a Master is ready or not ready to participate in the token ring or has already been integrated in the token ring.

The request packet has one parameter, namely the destination address, i.e. the station address of the station to be examined according to its FDL status. The destination address may range from 0 to 126.

The confirmation packet has one parameter, the returned 8-bit value containing the requested FDL status information of the station of your choice..

The table below explains the coding of the FDL status information:

FDL Status value	Meaning
0	Passive station (i.e. PROFIBUS slave) at this address
1	Active station not ready for token ring at this address
2	Active station ready for token ring
3	Active station in token ring
4	No station available at this address

Table 87: Possible Values of the FDL Status and their Meanings



Note: Sending this packet to all possible destination addresses (from 0 to 126) allows you to produce a life list of the PROFIBUS network.

Packet Structure Reference

```

/* Request-Packet to get FDL status */
typedef struct PROFIBUS_DL_SEND_FDL_STATUS_REQ_Ttag
{
    TLR_UINT8 bDstAddr; /* Value to be set */
}
PROFIBUS_DL_SEND_FDL_STATUS_REQ_T;

typedef struct PROFIBUS_DL_PACKET_SEND_FDL_STATUS_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_SEND_FDL_STATUS_REQ_T tData;
}
PROFIBUS_DL_PACKET_SEND_FDL_STATUS_REQ_T;

#define PROFIBUS_DL_SEND_FDL_STATUS_REQ_SIZE
(sizeof(PROFIBUS_DL_SEND_FDL_STATUS_REQ_T))

```

Packet Description

structure PROFIBUS_DL_PACKET_SEND_FDL_STATUS_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	1	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0132	PROFIBUS_DL_CMD_SEND_FDL_STATUS_REQ - Command
	ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_DL_SEND_FDL_STATUS_REQ_T			
	bDstAddr	UINT8	0..126	Destination Address (value to be set)

Table 88: PROFIBUS_DL_CMD_SEND_FDL_STATUS_REQ - Send FDL Status

Packet Structure Reference

```

/* Confirmation-Packet for get FDL status*/
typedef struct PROFIBUS_DL_SEND_FDL_STATUS_CNF_Ttag
{
    TLR_UINT8 bStatus;
}
PROFIBUS_DL_SEND_FDL_STATUS_CNF_T;

typedef struct PROFIBUS_DL_PACKET_SEND_FDL_STATUS_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_SEND_FDL_STATUS_CNF_T tData;      /* Packet Data Unit */
}
PROFIBUS_DL_PACKET_SEND_FDL_STATUS_CNF_T;

#define PROFIBUS_DL_SEND_FDL_STATUS_CNF_SIZE
(sizeof(PROFIBUS_DL_SEND_FDL_STATUS_CNF_T))

```

Packet Description

structure PROFIBUS_DL_PACKET_SEND_FDL_STATUS_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	0	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0133	PROFIBUS_DL_CMD_SEND_FDL_STATUS_CNF - Command
	ulExt	UINT32	0	Extension, unchanged
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_DL_SEND_FDL_STATUS_REQ_T			
	bStatus	UINT8	0...4	FDL Status (including station type) of the addressed PROFIBUS station. Also see Table 87: Possible Values of the FDL Status and their Meanings.

Table 89: PROFIBUS_DL_CMD_SEND_FDL_STATUS_CNF – Confirmation of Send FDL Status Request

4.3.17 PROFIBUS_DL_CMD_GET_LMS_REQ/CNF - Get List of active Stations

This packet allows accessing the “list of master stations” (LMS) in the logical ring also often called “list of active stations” (LAS). This list is represented by an array of 128 elements, of which, however, at most 127 are used.

Each array element (i.e. entry within the list) represents a station address within the PROFIBUS network.

- If there is either no active station present at this station address or if there is a PROFIBUS Slave at this station address, the corresponding array element will contain the value 255 (0xFF).
- If there is a PROFIBUS Master at this station address, the corresponding array element will contain the FDL address of the PROFIBUS Master to which this Master will pass the token next.



Note: A valid station address is defined in PROFIBUS to be a value in the range between 0 and 126 (including both limiting values).

The request packet does not have any parameters. The confirmation packet has one parameter, the returned array `abLms[128]` containing the “list of master stations”.

Packet Structure Reference

```
/* Get List of active stations */
typedef struct PROFIBUS_DL_PACKET_GET_LMS_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
}
PROFIBUS_DL_PACKET_GET_LMS_REQ_T;

#define PROFIBUS_DL_GET_LMS_REQ_SIZE (0)
```

Packet Description

structure PROFIBUS_DL_PACKET_GET_LMS_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	0	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0134	PROFIBUS_DL_CMD_GET_LMS_REQ - Command
	ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)
	ulRout	UINT32	x	Routing, do not change

Table 90: PROFIBUS_DL_CMD_GET_LMS_REQ - Get List of active Stations

Packet Structure Reference

```

/* Confirmation Packet for get LMS */
typedef struct PROFIBUS_DL_GET_LMS_CNF_Ttag
{
    TLR_UINT8 abLms[128];
}
PROFIBUS_DL_GET_LMS_CNF_T;

typedef struct PROFIBUS_DL_PACKET_GET_LMS_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_GET_LMS_CNF_T tData;      /* Packet Data Unit */
} PROFIBUS_DL_PACKET_GET_LMS_CNF_T;

#define PROFIBUS_DL_GET_LMS_CNF_SIZE (sizeof(PROFIBUS_DL_GET_LMS_CNF_T))

```

Packet Description

structure PROFIBUS_DL_PACKET_GET_LMS_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	128	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0135	PROFIBUS_DL_CMD_GET_LMS_CNF - Command
	ulExt	UINT32	0	Extension, unchanged
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_DL_GET_LMS_CNF_T			
	abLms[128]	UINT8[]	0...126,255 (for each array element)	List of active Stations

Table 91: PROFIBUS_DL_CMD_GET_LMS_CNF - Confirmation of Get List of active Stations Request

4.3.18 PROFIBUS_DL_CMD_REGISTER_LMS_REQ/CNF - Register an Application to receive LMS Change Indications

This packet allows registering in order to be informed about current changes in the list of master stations (LMS) also often denominated as list of active stations (LAS). Every time a change within this list occurs, the PROFIBUS DL task of the protocol stack will receive an indication packet (PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for Change in LMS, see next subsection) indicating that a change has occurred and where this change has occurred.

Each array element (i.e. entry within the list) represents a station address within the PROFIBUS network.

For more information about the list, see the preceding subsection beginning on page 150 of this document (PROFIBUS_DL_CMD_GET_LMS_REQ/CNF - Get List of active Stations).

The confirmation packet does not have any parameters.

Packet Structure Reference

```
/* Register an application to receive LMS change indications*/
typedef struct PROFIBUS_DL_REGISTER_LMS_REQ_Ttag
{
    TLR_BOOLEAN8 bEnable; /* Value to be set */
}
PROFIBUS_DL_REGISTER_LMS_REQ_T;

typedef struct PROFIBUS_DL_PACKET_REGISTER_LMS_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_REGISTER_LMS_REQ_T tData;
}
PROFIBUS_DL_PACKET_REGISTER_LMS_REQ_T;

#define PROFIBUS_DL_REGISTER_LMS_REQ_SIZE (sizeof(PROFIBUS_DL_REGISTER_LMS_REQ_T))
```

Packet Description

structure PROFIBUS_DL_PACKET_REGISTER_LMS_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	0	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0136	PROFIBUS_DL_CMD_REGISTER_LMS_REQ - Command
	ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_DL_REGISTER_LMS_REQ_T			
	bEnable	BOOLE AN8	0,1	Value to be set 0: Indications are switched off 1: Indications are switched on

Table 92: PROFIBUS_DL_CMD_REGISTER_LMS_REQ - Register an Application to receive LMS Change Indications

Packet Structure Reference

```
/* Register an application to receive LMS change indications*/
typedef struct PROFIBUS_DL_PACKET_REGISTER_LMS_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
} PROFIBUS_DL_PACKET_REGISTER_LMS_CNF_T;

#define PROFIBUS_DL_REGISTER_LMS_CNF_SIZE (0)
```

Packet Description

structure PROFIBUS_DL_PACKET_REGISTER_LMS_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulAPM0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulDL0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	0	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0137	PROFIBUS_DL_CMD_REGISTER_LMS_CNF-Command
	ulExt	UINT32	0	Extension, unchanged
	ulRout	UINT32	x	Routing, do not change

Table 93: PROFIBUS_DL_CMD_REGISTER_LMS_CNF – Confirmation of Register an Application to receive LMS Change Indications Request

4.3.19 PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for Change in LMS

Every time a change within the list of master stations (LMS) occurs, the PROFIBUS DL task of the protocol stack will receive this indication packet indicating that a change has occurred and where this change has occurred.

Each array element (i.e. entry within the list) contains the FDL address of a currently active PROFIBUS Master within the PROFIBUS network.

For more information about the list, refer to subsection PROFIBUS_DL_CMD_GET_LMS_REQ/CNF - Get List of active Stations beginning on page 150 of this document.

Receiving indications requires registration via the packet PROFIBUS_DL_CMD_REGISTER_LMS_REQ/CNF - Register an Application to receive LMS Change Indications. For more information about the list, see the preceding subsection beginning on page 153 of this document.

The indication packet has one parameter, the array `abLms[128]` containing the “list of master stations”.

Packet Structure Reference

```
/* Indication for changed LMS */
typedef struct PROFIBUS_DL_LMS_CHANGED_IND_Ttag
{
    TLR_UINT8 abLms[128];
}
PROFIBUS_DL_LMS_CHANGED_IND_T;

#define PROFIBUS_DL_LMS_CHANGED_IND_SIZE (sizeof(PROFIBUS_DL_LMS_CHANGED_IND_T))

/* Indication Packet for acknowledged connectionless data transfer */
typedef struct PROFIBUS_DL_PACKET_LMS_CHANGED_IND_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_LMS_CHANGED_IND_T tData;
}
PROFIBUS_DL_PACKET_LMS_CHANGED_IND_T;
```

Packet Description

structure PROFIBUS_DL_PACKET_LMS_CHANGED_IND_T				
Type: Indication				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination queue handle
	ulSrc	UINT32		Source queue handle
	ulDestId	UINT32	ulDL0Id	Destination end point identifier, specifying the final receiver of the packet within the destination process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulAPMS0Id	Source end point identifier, specifying the origin of the packet inside the source process
	ulLen	UINT32	128	Packet data length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet identification as unique number generated by the Source process of the packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0138	PROFIBUS_DL_CMD_LMS_CHANGED_IND - Command
	ulExt	UINT32	0	Extension (not in use, set to zero for compatibility reasons)
	ulRout	UINT32	x	Routing, do not change
tData	structure PROFIBUS_DL_DATA_REPLY_REQ_T			
	abLms[128]	UINT8[]	0...126,255 (for each array element)	List of active Stations

Table 94: PROFIBUS_DL_CMD_LMS_CHANGED_IND - Indication for acknowledged connectionless Data Transfer

4.3.20 PROFIBUS_DL_CMD_GET_LL_REQ/CNF – Get the Life List (List of Active or Passive Stations)

This packet allows to retrieve the life list, i.e. a list indicating which stations are currently active and which are passive.

Each array element (i.e. entry within the list) represents a station address within the PROFIBUS network.



Note: A valid station address is defined in PROFIBUS to be a value in the range between 0 and 126 (including both limiting values).

The single bytes in array `abLL[128]` in the response have the following meaning:

Value	Symbolic name	Meaning
0	PROFIBUS_DL_STATION_TYPE_NOT_EXISTANT	Station does not exist
1	PROFIBUS_DL_STATION_TYPE_PASSIVE	Passive station (i.e. PROFIBUS slave) at this address
2	PROFIBUS_DL_STATION_TYPE_READY_NO_TOKEN	Active station not ready for token ring at this address
3	PROFIBUS_DL_STATION_TYPE_ACTIVE	Active station

Table 95: Possible Values of Single Bytes in Response

Packet Structure Reference

```
/* Request-Packet to get list of active station */
typedef struct PROFIBUS_DL_PACKET_GET_LL_REQ_Ttag
{
    TLR_PACKET_HEADER_T tHead;
}
PROFIBUS_DL_PACKET_GET_LL_REQ_T;
#define PROFIBUS_DL_GET_LL_REQ_SIZE (0)
```

Packet Description

structure PROFIBUS_DL_PACKET_GET_LL_REQ_T				
Type: Request				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32	ulDL0Id	Destination End Point Identifier, specifying the final receiver of the packet within the Destination Process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulAPMS0Id	Source End Point Identifier, specifying the origin of the packet inside the Source Process
	ulLen	UINT32	0	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification as unique number generated by the Source Process of the Packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0130	PROFIBUS_DL_CMD_GET_LL_REQ - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch

Table 96: PROFIBUS_DL_CMD_GET_LL_REQ - Get Life List Request

Packet Structure Reference

```

/* Confirmation-Packet for get list of active station */
typedef struct PROFIBUS_DL_GET_LL_CNF_Ttag
{
    TLR_UINT8 abLL[128];
}
PROFIBUS_DL_GET_LL_CNF_T;

typedef struct PROFIBUS_DL_PACKET_GET_LL_CNF_Ttag
{
    TLR_PACKET_HEADER_T tHead;
    PROFIBUS_DL_GET_LL_CNF_T tData;      /* Packet Data Unit */
}
PROFIBUS_DL_PACKET_GET_LL_CNF_T;

```

Packet Description

structure PROFIBUS_DL_PACKET_GET_LL_CNF_T				
Type: Confirmation				
Area	Variable	Type	Value / Range	Description
tHead	structure TLR_PACKET_HEADER_T			
	ulDest	UINT32		Destination Queue-Handle
	ulSrc	UINT32		Source Queue-Handle
	ulDestId	UINT32	ulAPM0Id	Destination End Point Identifier, specifying the final receiver of the packet within the Destination Process. Set to 0 for the Initialization Packet
	ulSrcId	UINT32	ulDL0Id	Source End Point Identifier, specifying the origin of the packet inside the Source Process
	ulLen	UINT32	128	Packet Data Length in bytes
	ulId	UINT32	0 ... $2^{32}-1$	Packet Identification as unique number generated by the Source Process of the Packet
	ulSta	UINT32		See section „Error Codes of the DL-Task“.
	ulCmd	UINT32	0x0131	PROFIBUS_DL_CMD_GET_LL_CNF - Command
	ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
	ulRout	UINT32	x	Routing, do not touch
tData	structure PROFIBUS_DL_GET_LL_CNF_T			
	abLL[128]	UINT8[]	0..3	Life List (see explanation above)

Table 97: PROFIBUS_DL_CMD_GET_LL_CNF - Confirmation to Get Life List Request

5 Status/Error Codes Overview

If a error occurred there is no guaranty that the data which the caller was send have valid values on return. Normally there two cases of error.

First one occurred if the packet has wrong content, for example Station address is out of range. In this case the field ulLen within the packet header is not touched. In this case the caller is responsible that the data are valid.

The second error can occur while the packet is in processing. Then the task will determine which data are valid, and set the ulLen field to this value. So it is possible, that the ulLen field is set to 0. Is the caller need the information within the packet he must save this data locally and must refer to this packet. He can use the ulId field or the ulSrcId field for this purpose.

5.1 PROFIBUS MPI-Error Codes

Hexadecimal Value	Definition Description
0xC0670005	TLR_E_PROFIBUS_DATA_CNT Data Counter Error.
0xC0670006	TLR_E_PROFIBUS_MPI_ILLEGAL_STATION_ADDRESS Station Address is invalid.
0xC0670007	TLR_E_PROFIBUS_MPI_ILLEGAL_SOCKET_NUMBER Socket number is invalid.
0xC0670008	TLR_E_PROFIBUS_MPI_COMMUNICATION_ABORTED Communication is aborted by the remote station.
0xC0670009	TLR_E_PROFIBUS_MPI_COMMUNICATION_REFUSED Communication is refused by the remote station.
0xC067000A	TLR_E_PROFIBUS_MPI_ERROR_WHILE_BARGAIN_LEN Communication error while bargain max data len.
0xC067000B	TLR_E_PROFIBUS_MPI_DATA_OUT_OF_ORDER Received packet has wrong sequence number.
0xC067000C	TLR_E_PROFIBUS_MPI_DISCONNECT_REQUEST Host application has send an disconnect request.
0xC0670130	TLR_E_PROFIBUS_MPI_CON_TO Time out.
0xC0670139	TLR_E_PROFIBUS_MPI_CON_SE Sequence error.
0xC0670203	TLR_E_PROFIBUS_MPI_ILLEGAL_FUNCTION_NUMBER Illegal function number
0xC0670204	TLR_E_PROFIBUS_MPI_RESET_IN_PROGRESS Reset in progress.
0xC0670205	TLR_E_PROFIBUS_MPI_TOO_MANY_REQ_IN_PROGRESS

Hexadecimal Value	Definition Description
	Too many requests in progress.
0xC0670206	TLR_E_PROFIBUS_MPI_DENIED_BY_WATCHDOG_TO No access because of Watchdog Timeout.
0xC0670207	TLR_E_PROFIBUS_MPI_ILLEGAL_WATCHDOG_TIME Illegal watchdog time.
0xC0670208	TLR_E_PROFIBUS_MPI_CON_IN_PROGRESS Connection to PLC in progress.
0xC0670209	TLR_E_PROFIBUS_MPI_BUS_ALREADY_CONFIGURED Bus already configured.
0xC0670281	TLR_E_PROFIBUS_MPI_REJ_SE Device stopped communication or is not in Open State.
0xC0670282	TLR_E_PROFIBUS_MPI_REJ_ABORT Device aborts communication.
0xC0670283	TLR_E_PROFIBUS_MPI_REJ_PS Previous Service still in Progress
0xC0670284	TLR_E_PROFIBUS_MPI_REJ_LE Length Error.
0xC0670285	TLR_E_PROFIBUS_MPI_REJ_IV Specified offset out of limits or not known to remote station.
0xC0670286	TLR_E_PROFIBUS_MPI_REJ_PDU Wrong PDU coding.
0xC0670287	TLR_E_PROFIBUS_MPI_REJ_OP Specified Length to read or write out of limits.
0xC0670288	TLR_E_PROFIBUS_MPI_REJ_HW Specified address not defined in remote station.
0xC0670289	TLR_E_PROFIBUS_MPI_REJ_MODE Remote station not in right operational mode.
0xC0670290	TLR_E_PROFIBUS_MPI_UNKNOWN_ERROR Unknown error.

Table 98: MPI Error Codes

5.2 Error Codes of the MPI AP-Task

Hexadecimal Value	Definition Description
0xC0680001L	TLR_E_PROFIBUS_MPI_AP_COMMAND_INVALID Invalid command received.

Table 99: Error Codes of the MPI AP-Task

5.3 Error Codes of the DL-Task

Hexadecimal Value	Definition Description
0xC0060001	TLR_E_PROFIBUS_DL_COMMAND_INVALID Invalid command received.
0xC0060040	TLR_E_PROFIBUS_DL_XC_INVALID The assigned XC-Data Link Layer is not installed or has a pending error.
0xC0060041	TLR_E_PROFIBUS_DL_BAUDRATE_INVALID The specified baudrate option is not supported and is out of range.
0xC0060042	TLR_E_PROFIBUS_DL_GAP_UPDATE_INVALID The specified GAP update factor option is not supported and is out of range 1-100.
0xC0060043	TLR_E_PROFIBUS_DL_DL_ADDR_INVALID The specified local Profibus address option is not supported and is out of range 0-125.
0xC0060044	TLR_E_PROFIBUS_DL_RETRY_LIMIT The specified retry limit option is not supported and is zero.
0xC0060045	TLR_E_PROFIBUS_DL_HSA_INVALID The specified highest station address option is not supported and is out of range 0-126.
0xC0060046	TLR_E_PROFIBUS_DL_NO_BUS_PARAMETER_SET The service cannot be executed, there are no bus parameters specified yet.
0xC0060047	TLR_E_PROFIBUS_DL_DLE_NOT_RESPONDING The service has detected a timeout at the connected XC-Data Link Layer entity.
0xC0060048	TLR_E_PROFIBUS_DL_NO_DL_RESOURCE There are no further resource blocks available to execute the service within the connected XC-Data Link Layer entity.
0xC0060049	TLR_E_PROFIBUS_DL_FATAL_DL_RESOURCE There are no further resource blocks available to execute the service within the connected XC-Data Link Layer entity.
0xC0060050	TLR_E_PROFIBUS_DL_STOPPED Profibus is stopped command cannot be handled.
0xC0060051	TLR_E_PROFIBUS_DL_PENDING_PACKET Previous pending packet is returned. It could not be handled.
0xC0060052	TLR_E_PROFIBUS_DL_SLAVE_MODE Command could not be executed, DL-task is running at slave mode.
0xC0060080	TLR_E_PROFIBUS_DL_ACK_UE The remote station the service has been sent to indicates a User Error as service acknowledgement.
0xC0060081	TLR_E_PROFIBUS_DL_ACK_RR The remote station the service has been sent to indicates a Resource Error as service acknowledgement.
0xC0060082	TLR_E_PROFIBUS_DL_ACK_RS The remote station the service has been sent to indicates a Service Access Point Error as service acknowledgement.
0xC0060083	TLR_E_PROFIBUS_DL_ACK_NR The remote station the service has been sent to confirms its positive reception but has no data to confirm.
0xC0060084	TLR_E_PROFIBUS_DL_ACK_RDH The remote station the service has been sent to, confirms its reception negatively but has returned low priority data in the response.

Hexadecimal Value	Definition Description
0xC0060085	TLR_E_PROFIBUS_DL_ACK_RDL The remote station the service has been sent to, confirms its reception negatively but has returned high priority data in the response.
0xC0060086	TLR_E_PROFIBUS_DL_ACK_DH The remote station the service has been sent to, confirms its reception positively and has returned low priority data in the response.
0xC0060087	TLR_E_PROFIBUS_DL_ACK_DL The remote station the service has been sent to, confirms its reception positively and has returned high priority data in the response.
0xC0060088	TLR_E_PROFIBUS_DL_ACK_NA The remote station the service has been sent to shows no or no plausible reaction at all.
0xC0060089	TLR_E_PROFIBUS_DL_ACK_UNKNOWN The remote station the service has been sent has returned an unknown acknowledgement code.
0xC006008A	TLR_E_PROFIBUS_DL_ACK_LS The requested service is not activated within the local SAP configuration.
0xC006008B	TLR_E_PROFIBUS_DL_ACK_LR The local resources needed to execute the requested service are not available or not sufficient.
0xC006008C	TLR_E_PROFIBUS_DL_ACK_DS The local data link layer is not in the logical token ring or disconnected from the network.
0xC006008D	TLR_E_PROFIBUS_DL_ACK_IV Invalid parameter detected in the requested service.
0xC006008E	TLR_E_PROFIBUS_DL_ACK_NO The local SAP is not activated because it has been activated already or resources are not sufficient.
0xC006008F	TLR_E_PROFIBUS_DL_ACK_NO_SET The variable to be set does not exist.
0xC0060090	TLR_E_PROFIBUS_DL_ACK_RE Format error of the telegram.
0xC0060091	TLR_E_PROFIBUS_DL_TSET_INVALID The specified parameter TSET is out of range 1-255.

Table 100: Error Messages of the DL-Task

5.3.1 Diagnostic Codes of the DL-Task

Hexadecimal Value	Definition Description
0xC0060001L	TLR_DIAG_E_PROFIBUS_DL_DATA_ACK_RES Invalid "Data Ack (SDA)" response packet received. Response does not match to internally reserved communication block
0xC0060002L	TLR_DIAG_E_PROFIBUS_DL_DATA_RES Invalid "Data (SDN)" response packet received. Response does not match to internally reserved communication block
0xC0060003L	TLR_DIAG_E_PROFIBUS_DL_DATA_REPLY_RES Invalid "Data Reply (SRD)" response packet received. Response does not match to internally reserved communication block

Table 101: Diagnostic Messages of the DL-Task

6 Contact

Headquarters

Germany

Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support

Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China

Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support

Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France

Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support

Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India

Hilscher India Pvt. Ltd.
New Delhi - 110 025
Phone: +91 11 40515640
E-Mail: info@hilscher.in

Italy

Hilscher Italia srl
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support

Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan

Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support

Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea

Hilscher Korea Inc.
Suwon, 443-810
Phone: +82-31-204-6190
E-Mail: info@hilscher.kr

Switzerland

Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support

Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA

Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support

Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com