Protocol interface manual

PROFIBUS-FMS

Hilscher Gesellschaft für Systemautomation mbH
Rheinstraße 78
D-65795 Hattersheim
Germany

Tel. +49 (0) 6190 / 9907 - 0
Fax. +49 (0) 6190 / 9907 - 50

Sales: +49 (0) 6190 / 9907 - 0
Hotline and Support: +49 (0) 6190 / 9907 - 99

e-mail: hilscher@hilscher.com
Homepage: http://www.hilscher.com

Index	Date	Version	Chapter	Revision
7	06. 05. 98	V3.000	all	Review for firmware version 3.000 This version is upward compatibel to V2.100, without the database. All applications for V2.100 can work with this firmware but they must use SyCon V2.000 to create the configuration database.
8	24.06.99	V3.000	3.2 3.2.2 3.3	Error numbers added Description of the client message added Define Variable List, Delete Variable List added
9	23.09.99	V3.000	3.3.10	Structure of Variable-List Object description added

Although this protocol implementation has been developed with great care and intensively tested, Hilscher Gesellschaft für Systemautomation mbH cannot guarantee the suitability of this protocol implementation for any purpose not confirmed by us in writing.

Guarantee claims shall be limited to the right to require rectification. Liability for any damages which may have arisen from the use of this protocol implementation or its documentation shall be limited to cases of intent.

We reserve the right to modify our products and their specifiactions at any time in as far as this contributes to technical progress. The version of the manual supplied with the protocol implementation applies.

1 General

This manual describes the protocol interface of PROFIBUS-FMS for our Communication Interfaces and the Communication Modules. The aim of this manual is to support the integration of these devices into own applications based on the Device Driver or direct access into the dual-port memory.

The general mechnism for the data transfer is protocol independent and common for all our devices. Therefore this general functions are described in separate manuals:

Device Driver	This manual describes how to use our Device Driver to communicate with our devices. The Device Driver offers a DLL with a C function interface and header files and can used under the operating systems Windows 95 and Windows NT.
Tool Kit-Manual	This manual describes the data structure of the dual-port memory and its handling. If you want to write your own device driver or you are working with an other operating system which we don't support this is the right choice. Contents of the Tool Kit is a C-Library in source code and header files which you can use directly under MS/DOS and Windows 3.11. It can also be used as basis to modify and create your own driver.

This manual, named protocol interface, defines the structures of the messages which you must send to the device or you will receive from the device to handle the PROFIBUS-FMS services. Furthermore it defines the sequence of the messages and the error numbers. You must used these definitions to create the messages for the Device Driver or to write it directly in the dual-port memory as described in the Tool Kit.

Please look at first in the other manuals to understand the principle things before you continue.

We assume that you are familiar with the principle functions of PROFIBUS. You can find an introduction in our protocol manual or in the following book:

- PROFIBUS specification EN50170, part 2.

This manual explain the PROFIBUS-FMS functions only if it is neccessary to understand how we use it with our application programmers interface.

2 Spezification and Protocol Fundamentals

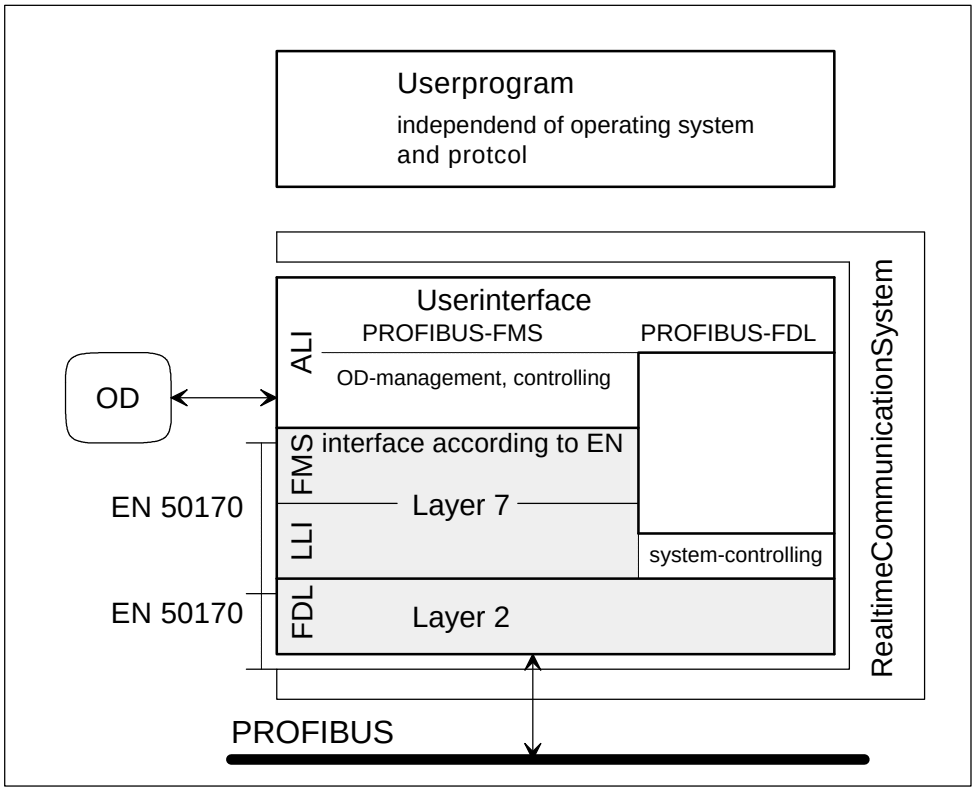
The implementation is according to EN 50170 part 2.

2.1 PROFIBUS-FMS

2.1.1 Structure

The function of data transfer is divided into individual layers with the PROFIBUS. Each layer has a function defined in EN 50170 part 2. In our PROFIBUS implementation, each layer is implemented by a separate task, coordinated by a real-time operating system. The PROFIBUS interface proper to EN50170 has been used to integrate the PROFIBUS application layer, the application layer interface (ALI), in the software. The ALI manages the object directory and images the PROFIBUS DIN functionality on a protocol independent user interface. The ALI interacts with the PROFIBUS Field bus Message Specification (FMS) interface to DIN, or alternatively with the Field Data Link (FDL).

The illustration below shows the relationships between the individual layers (tasks). The PROFIBUS section to EN 50170 is shown in grey.



Structure of the PROFIBUS software, links between the layers

Field bus Data Link Layer (FDL)
The function of the FDL is that of transferring the data from one device to another. This comprises the addressing and synchronisation of the stations on the bus (token passing). The FDL layer does not define any data contents or protocol sequences.

Lower Layer Interface (LLI)
The LLI regulates the sequence or data transfer. Among other functions, it decides how the software has to react in the case of a communication error or time-out. Furthermore, the LLI is the interface between the logical (communications reference) and physical addressing systems (station address, service access point).

Fieldbus Message Specification (FMS)
The FMS layer is the PROFIBUS application layer to EN 50170 Part 2. It includes the user data in a PROFIBUS compatible message format and makes the services accessible to the user available.

The FMS also defines the format of the user data. A uniform data format is necessary for the communicating stations to be able to interpret the data they exchange. PROFIBUS FMS, for example, specifies the word format as high-byte/low-byte, whereas data words in a PC are stored as low-byte/high-byte (Intel format). Further information can be found in the chapter data format.

Application Layer Interface (ALI)
The ALI no longer belongs to the PROFIBUS layer to EN standard. The ALI itself is the application layer of the PROFIBUS. It interacts with the service made available by the PROFIBUS. The ALI in conjunction with the user program organises the PROFIBUS communications data, objects and functions.

The ALI is already integrated in our devices, in order to implement as simple a user interface as possible.

3 The PROFIBUS Communication Services

Data transfer on the PROFIBUS is effected by means of services which the PROFIBUS provides to you. One example is the read service, with which you can read data made available by another device.

The services we support are listed below:

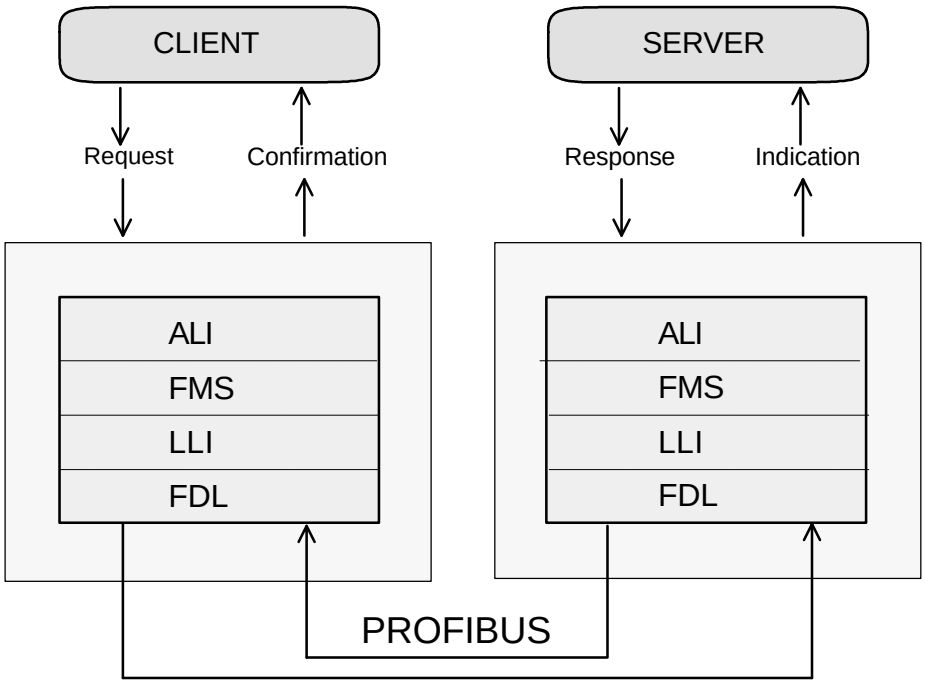
Identify	Information to identify a Virtual Field Device is read with this service.
Status	The device or the user state is read with this service.
Get OD	With the GetOD service one or several Object Description are read.
Read	The values of Simple Vaiable objects, arrays of the communication partner may be read by this service.
Write	Values are written in the objects Simple Variable, and Arrays with this service.
Event Notification	The Event Notification service allows the transmission of an Event Notification.
Information Report	Values of the objects Simple Variable and Arrays are transmitted by using this service.
Abort	This service shall be used to release an existing Communication Relationsship between the communication partners.
Define Variable List (Client)	This service define a variable list object.
Delete Variable List (Client)	This service delete a variable list object.

The service Initiate is used implicit if you activated a service on a Communication Relationship which is not established. This simplifies the application software essential, because you can activate a service in any case. The ALI on the device will send an Initiate for you if it is required.

According to the PROFIBUS conventions, the device which wishes to read must request the service. This device is named the 'client'. A second device whose data are to be read must respond to the read request. This device is known as the 'server'.
Depending on the location and the transfer direction the services are indicated as

Request	service which is sended from the client
Indication	service which is received by the server
Response	acknowledge of the service which is send from the server
Confirmation	acknowledge of the service which is received by the client

The following figure is intended to illustrate the relationship between the terms Request / Confirmation and Indication / Response. Understanding of these concepts is important for all the following chapters.



Some of the services are unconfirmed, which are used for broadcast and multicast telegrams on the PROFIBUS line. This means that the Client don't get a confirmation for its request and the Server must not send a reponse.
In our implementation you can get a confirmation from the local device for an unconfirmed service to handle these identical to the confirmed services.

Unconfirmed services are:

Abort

Event Notification

Information Report.

3.1 General Structure of a Send/Receive Message

The communication with PROFIBUS-FMS is message oriented and take place via the `DevMailbox` and the `HostMailbox` which are located in the dual-port memory of the device. Please look at the Tool Kit or the Device Driver manual for details, how to send or receive a message.

This chapter described the general structure of the messages which containts the PROFIBUS services. You will find explanatory description for every service on the following pages.

The message structure is common to all our devices. Therefore it is composed of 16 bytes of header data and, where appropriate user data. The 16 bytes of header data is divided into two blocks of 8 bytes each. The first 8 bytes are designated the message header. They contain parameter required for transport of the message (e.g. sender, receiver, ...). The following 8 bytes are designated the telegram header. These contain parameter required by the protocol in this case the PROFIBUS service parameters.

For PROFIBUS-FMS all messages are exchanged between the ALI and the USER. Depending on the direction *msg.rx* and *msg.tx* containes only ALI and User.
The parameter *msg.ln* defines the length of the message including the eight bytes of the telegram header.
The message identification *msg.nr* can be a constant value like 0, if there are no parallel services. If the user activates more then one service at a time all services must get a different message identification number. The confirmation created by the ALI will get the same message identification as the corresponding request. Also the response which the User send back to the ALI must get the same message identification as the corresponding indication.
The Command Identification *msg.b* and the Reply Identification *msg.a* is defined as TASK_B_11 for all PROFIBUS-FMS services.
If *msg.f* is not equal 0 then the message must be interpreted as an error message. Also the user can answer with an error number at *msg.f* in his response to the ALI at a servcer. This number becomes an Additional Detail in the negativ confirmation for the client.
With the extension *msg.e* the user has the possibility to suppress a positive confirmation for an unconfirmed service.

The structure of the message is shown on the following table. An explanation of the parameters for the message header can be found in both manual mentioned above.

	Parameter	Type	Define	Value	Description
Message header	msg.rx	unsign char	 ALI USER	 1 16	Identification ot the receiver ALI (request / response) User application (indication / confirmation)
	msg.tx	unsign char	 USER ALI	 16 1	Identification of the sender User application (request / response) ALI (indication / confirmation)
	msg.ln	unsign char		 8...249	Message length Number of bytes started at bCommunicationRef
	msg.nr	unsign char		0...255	Message identification Unique number (Invoke ID)
	msg.a	unsign char	 TASK_B_11	 0 17	Reply identification Request / Indication Response / confirmation
	msg.f	unsign char	 see ALI error table	 0 1...255	Error number No error Error number (response / confirmation)
	msg.b	unsign char	 TASK_B_11	 17 0	Command identification Request / indication Response / confirmation
	msg.e	unsign char		 0 2	Extention Standard No reply expected
Telegram header	bCommunicationRef	unsign char		1...32	Communication reference
	bAccess	unsign char	ACCESS_INDEX	0	Access mode
	usObjectIdx	unsign short		20...6553 5	Object index
	bSubObjectIdx	unsign char		 0 1...241	Object subindex Whole object Access by subindex (only arrays)
	bDataCnt	unsign char		0...241	Number of user data Of the type given in bDataType'
	bDataType	unsign char	 DT_BOOLEAN DT_INT8 DT_INT16 DT_INT32 DT_UINT8 DT_UINT16 DT_UINT32 DT_FLOAT DT_OC_STRING	 1 2 3 4 5 6 7 8 10	Data type of user data Boolean Integer 8 Bit Integer 16 Bit Integer 32 Bit Unsigned Integer 8 Bit Unsigned Integer 16 Bit Unsigned Integer 32 Bit Floating Point Octet String
	bService	unsign char	 TASK_TFC_READ TASK_TFC_WRITE STATUS_SERVICE IDENTIFY_SERVICE READ_SERVICE WRITE_SERVICE GET_OD_SERVICE GET_INFO_REPORT_SERVICE EVENT_NOTIFICATION_SERVICE ABORT_SERVICE	 1 2 0x80 0x81 0x82 0x83 0x84 0x9F 0xA1 0xA5	Service Read Write Status Identify Read Write GetOD Information Report Event Notification Abort
Data	abData[ ]	unsign char[ ]			User data If existing

The parameters of the telegram header are described in detail at the next page.

Communication reference (CR)
The value to be entered for the CR is the line in which the corresponding CR can be found in the communication relationship list (CRL). A value of 3 is therefore entered for CR 3, which is the CR in the third line of the CRL.
The Communication reference is used to define the communication partner for the actual service.

Object index and object subindex
The object index and object subindex designate the object to be read or written. A subindex of zero always refers to the entire object - thus permitting even a complete array to be read. A subindex not equal to zero refers to an element in an array and is not permissible for simple objects. Please note that the first element of an array has the subindex 1.

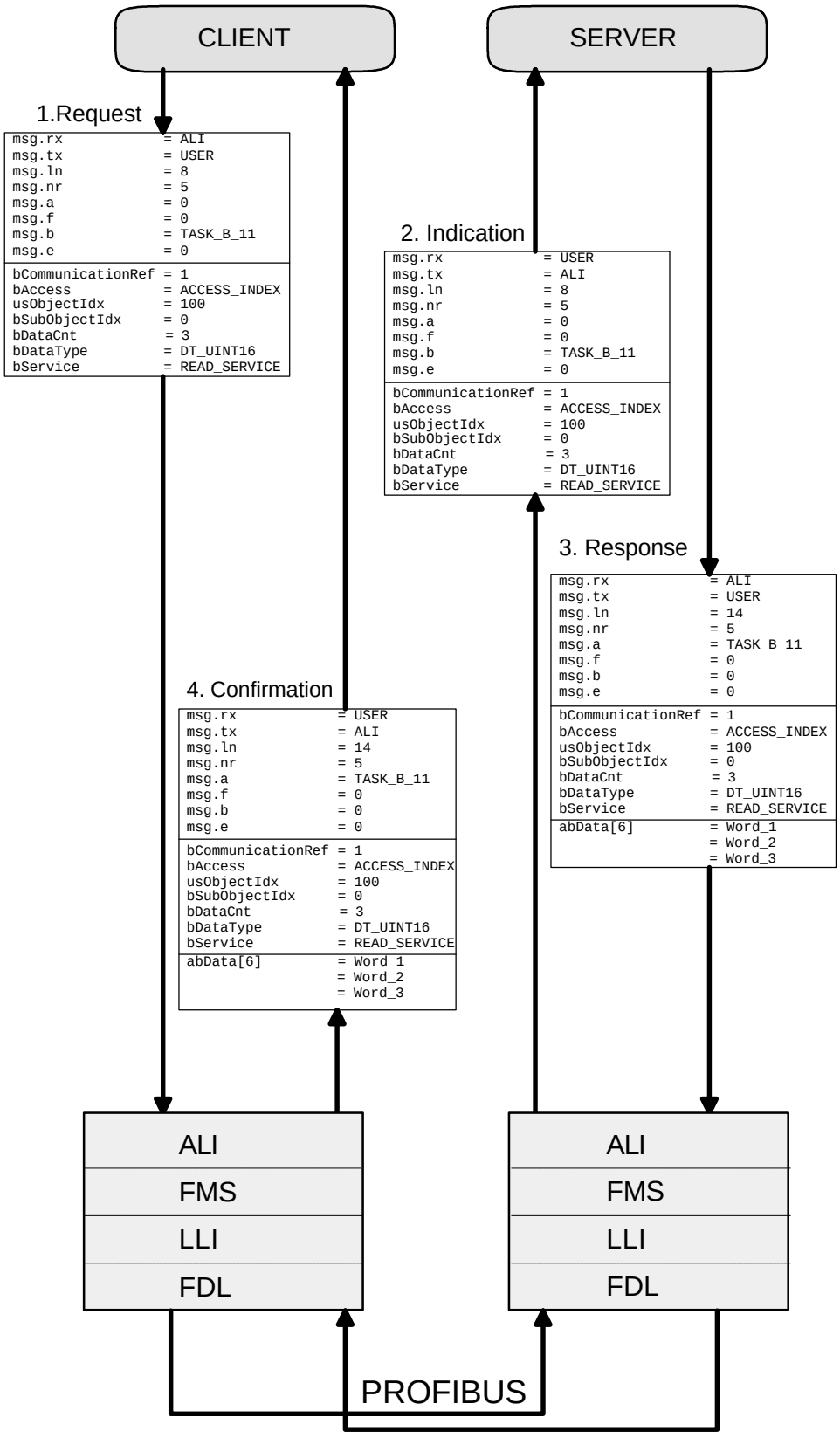
Number of user data and data type
The parameters *bDataCnt* and *bDataType* defines how much data should be read or written. It is important for write services that these parameters match the quantities of bytes in the user data and the parameter *msg.ln*.
The data type is defined in accordance with the PROFIBUS standard.
For simple Objects and array elements the number of user data is always 1 and for the whole array it is the number of elements of the array.
The octet string is an exception: here, the number of the user data - measured in bytes - is always stated.
On sending or receiving PROFIBUS read services, the number of user data and the data type defines the number and structure of the expected user data. These parameters are not a part of the PROFIBUS telegram. So the server sends as much as in its object directory for this object defined data back to the client.

Service
The service indicates what function the PROFIBUS has to perform. More detailed explanations of each service are presented on the following pages.

User data
The user data contain the useful data to be transmitted. In the configuration you can switch the format of the user-data between LSB-MSB (little endian) and MSB-LSB (big endian) for the 16 and 32 bit data types and floating point values. Please note that the parameter in the telegram header are always in the format LSB-MSB.

Note for FDL-defined communication:
All data from parameter *bCommunicationRef* are transmitted through the PROFIBUSBUS. Only the *bCommunicationRef* parameter is interpreted by the PROFIBUS software, in order to identify the target device. The physical target device address is determined from the communication relationship list by the communication reference. The user data which the communicating partner has returned as a response to a read request are located below the parameter *bCommunicationRef* in the response. The PROFIBUS software does not interpret this data. The communication partner, preferably an S5-95U, is therefore fully responsible for the contents of the data.

The following diagram shows a read service at object 100 for 3 words.



Read the object 100 which is an array of 3 words

3.2 General Error Messages

The following message is returned to the user application instead of the expected confirmation if an error is detected:

	Parameter	Type	Define	Value	Description
Message header	msg.rx	unsign char	USER	16	Identification ot the receiver User application (confirmation)
	msg.tx	unsign char	ALI	1	Identification of the sender ALI (confirmation)
	msg.ln	unsign char		8	Message length Number of bytes started at bCommunicationRef
	msg.nr	unsign char		0...255	Message identification Unique number (Invoke ID)
	msg.a	unsign char	TASK_B_11	17	Reply identification Confirmation
	msg.f	unsign char	See ALI error table	1.255	Error number
	msg.b	unsign char		0	Command Identification Confirmation
	msg.e	unsign char		0	Extention Standard
Telegram header: The telegram header is the same as the belonging command	bCommunicationRef	unsign char		1...32	Communication reference
	bAccess	unsign char		0	Access mode
	usObjectIdx	unsign short		20...65535	Object index
	bSubObjectIdx	unsign char		0...241	Object subindex
	bDataCnt	unsign char		0...241	Number of user data
	bDataType	unsign char		1-8, 10	Data type of user data
	bService	unsign char		1, 2, 0x80...	Service

The user must check every confirmation if there is an error number at *msg.f*. If this is not equal zero an error is detected.

An error message will received only after a service request is activated. Both have the same message identification and the same service parameter in the first eight bytes after the message header. In order with these information the user can assign the error message to the belonging service request, if parallel services are used.

If the connection is aborted by the remote station, there will be generated an error message for every outstanding service to the local user. If the abort is generated from the local user he will get only the confirmation of his abort service. All outstanding services will be deleted in the local ALI without any error message to the local user.
If there are no outstanding services and the remote station sends an abort, the local user will not get any information about this event, because there are no service parameters which can be send back to the user. It is also not necessary, because the local ALI will try to establish the connection by itself with the next service request from the user.

Definition	No. hex	No. dez	Description
	0x00	0	No error
ALI_ABORT_ERR	0x42	66	Connection has been aborted A local PROFIBUS-layer (FDL, LLI, FMS, ALI) or the remote partner has closed the connection. All requests by the client will be confirmed with this error number. All indications by the server will be lost. By sending a new request on this CR, the firmware will try to open the connection automatically by sending an initiate-telegram. **See additional details at message data to get more information, descriped in 'Connection aborted'**
ALI_REJECT_PAR_SRV	0x43	67	Too many parallel services on one CR ALI received Reject service with Reject Code for Max-Service-Overflow. This means confirmed Service.Request received from the user and Out-standing Services Counter Client **≥** Outstanding Services Client. Check setting of ´SCC´ or 'SAC' in CRL. On FDL-communication there can be generally sended only one open service, independend of the SCC/SAC parameters in the CRL.
ALI_INVOKE_ID_ERR	0x44	68	Invoke-ID in use for running service Check the consistence of your user-program according to the messagenumber. When you send a seccond telegram with the same messagenumber before receiving the answer of the first one, this error will be received.
ALI_REJECT_PDU_LENGTH	0x45	69	Telegram too long to send ALI received Reject service with Reject Code for PDU-Size. This means PDU length exceeds maximum PDU length. Check telegram length or value of ´PDU-Length´ in CRL.
ALI_REJECT_SRV_NOT_SUPP	0x46	70	Client-Service not supported ALI received Reject service with Reject Code for Feature-Not-Supported-Connection-Oriented or Feature-Not-Supported-Connectionless. This means Service Request received from the user and service or option not supported as a client. Check client services in CRL.
ALI_REJECT_OTHER	0x47	71	Telegram cannot be evaluated by local FMS ALI received Reject service with Reject Code for Other.
ALI_CON_RR	0x48	72	Negative acknowledgement, resource of the remote FDL controller not available or not sufficent and reply data not available (RR error from FDL) The remote-FDL will send this error-number, when there is no segment or receivebuffer free to receive this telegram. It's a temporary error. By receiving this error, the local LLI will disconnect the connection.
ALI_CON_NA	0x49	73	No or no plausible reaction of remote FDL (NA error from FDL) There is no connection between the local and the remote FDL. Possibilities are, that the stations are not connected, switched off or the bus-parameters are invalid or inconsistent. Check with default-BUS-parameters.
ALI_CLOSED_FMS_SERVICE	0x4A	74	FMS service received at closed Communication Reference
ALI_FDL_PAR_SRV	0x50	80	Parallel service during FDL defined FDL defined don't support parallel services.
ALI_FDL_ABORT	0x55	85	Abort is send from user during FDL defined
ALI_REJECT_INVOKE_ID_EXIST	0x56	86	Invoke Id already exists ALI received Reject service with Reject Code for Invoke-Id-Exists. This means confirmed Service Request received from the user and Invoke ID already exits.

Definition	No. hex	No. dez	Description
ALI_FDL_ERR			FDL error received This range containes the most used FDL errors.
	0x60	96	All other FDL errors without the following.
	0x61	97	UE Negative acknowledgement, remote User interface error.
	0x62	98	RR Negative acknowledgement, no remote resources available.
	0x63	99	RS No service or no remote address activated at Remote Service Access point.
	0x64	100	RA Negative acknowledgement, access-point blocked
	0x68	104	DL Positive acknowledgement for sent data, reply data with low priority available.
	0x69	105	NR Positive acknowledgement for sent data, negative acknowledgement for reply data, as not available to the remote FDL controller
	0x6A	106	DH Positive acknowledgement for sent data, reply data with high priority available.
	0x6C	108	RD Negative acknowledgement for sent data, resource of the remote FDL controller not available or not sufficent, reply data with low priority available.
	0x6D	109	RDH Negative acknowledgement for sent data, resource of the remote FDL controller not available or not sufficent, reply data with high priority available.
	0x70	112	LS Service at local SAP or local SAP not activated.
	0x71	113	NA No reaction (ACK/RES) from remote station
	0x72	114	DS Local FDL/PHY controller not in logical token ring or disconnected from line.
	0x73	115	NO No reply data transmitted.
	0x74	116	LR Resource of the local FDL controller not available or not sifficient.
	0x75	117	IV Invalid parameter in request.
	0x76	118	BR
	0x77	119	NM
	0x78	120	NC
	0x79	121	NP
	0x7A	122	NL
	0x7B	123	RD
	0x7C	124	SV
ALI_INITIATE_ERR	0x80	128	Connection could not be opened By the first request the connection has to be opened by sending an initiate-telegram. If the remote-partner confirmed this initiate negative, the connection could not be opened and the request will be reject with this error. Please check the local and remote configuration of this CR in KBL. **See error code at message data to get more information, described in chapter 'Connection could not be established'.**
ALI_REMOTE_ERR	0x81	129	Error in application of remote-partner The communication partner (server) has reject the request with an error. Possible reasons: a) Access on an non existing object b) Data-length of sending data is not consistent to data-length of object c) The user-program has reject the service **See error type at message data to get more information, described in chapter 'Negative response from remote station - Error-Type'**.
ALI_F_VFD_WRONG_STATE	0x87	135	Local state does not allow to send You can configure, that there can only be send telegrams after be taken up in the logical token-ring. Before taken up in the logical token-ring, all commissions will be rejected with this error.

Definition	No. hex	No. dez	Description
ALI_FMS_MSG_F	0x88	136	FMS reply contains error number at msg.f
ALI_TASK_NOT_INIT	0x89	137	The ALI-task is not initialized Check configuration and download.
ALI_DATA_TYPE_INVALID	0x8A	138	Invalid datatype Check datatype-unsign char of your commission. On FDL-transparent there is only allowed the OCTET-STRING
ALI_DATA_CNT_INVALID	0x8C	140	Invalid data count Check consistence of msg.ln-unsign char to len-unsign char.
ALI_UNKNOWN_FMS_ SERVICE	0x8D	141	Invalid FMS service
ALI_USER_UNKNOWN_ID	0x8E	142	Unknown Msg.nr / Invoke ID get from user
ALI_USER_ANSWER_ NON_EXPECTED	0x8F	143	Non expected reply received from user
ALI_OD_ACCESS_TYPE	0x90	144	Unknown bAccessSpec
ALI_OD_ACCESS_MODE	0x91	145	Unknown AccessMode
ALI_OD_ACCESS_ERR	0x92	146	OV_NEXT without OV_FIRST
ALI_OD_OBJ_ERR	0x93	147	Access error for local object
ALI_OD_ACCESS_OBJ	0x94	148	Objet non exsistent on local station
ALI_USER_DATA_ERR	0x95	149	To less data in reply from user
ALI_FMS_UNKNOWN_ID	0x96	150	Invalid Invoke ID from FMS
ALI_CR_INVALID	0x97	151	Invalid Communication Reference
ALI_MESSAGE_LEN	0x98	152	Message length is less than telegramm header
ALI_SEGMENT_MISSING	0x9A	154	No free segments available
ALI_UNKNOWN_SERVICE	0x9B	155	Invalid PROFIBUS service Check service-unsign char of your commission
ALI_B_UNKNOWN	0x9C	156	Invalid command from user Check msg.b.
ALI_A_UNKNOWN	0x9D	157	Invalid answer from user Check Msg.a.
ALI_FDL_UNKNOWN_SERVICE	0xA0	160	Unknown service for FDL defined
ALI_FDL_DATA_TYPE	0xA1	161	Invalid bDataType for FDL defined
ALI_FDL_DATA_CNT	0xA2	162	Invalid bDataCnt for FDL defined
ALI_LOCAL_SAP_INVALID	0xA3	163	Local SAP non existing on actual CR for FDL communication
ALI_FDL_ID_ERROR	0xA4	164	Unknown FDL reply
ALI_REMOTE_DATA_ERR	0xA5	165	Error at reply from FDL defined
ALI_FDL_TO_LESS_DATA	0xA6	166	Reply from FDL defined containes to less data
ALI_VFD_F_LOGICAL	0xB0	172	Invalid logical state
ALI_VFD_F_PHYSICAL	0xB1	173	Invalid physical state
ALI_VFD_F_DEATIL_LEN	0xB2	174	Invalid DetailLen by setting VFD state
ALI_INIT_DBM_ERR	0xC1	193	Database error during initialisation
ALI_INIT_OC_NOT_ SUPPORTED	0xC2	194	Not supported object code at object directory
ALI_MAX_KR_ERR	0xC4	196	To many Communication References
ALI_F_DBM	0xC6	198	Error during access at the database

There are some error message which contains more information about the error in the message data. We recommondate urgently to evalute these data and make it visuable to the user otherwise it is very difficult to detect the real reason of an error between two PROFIBUS-FMS stations.
The following error messages have additional information available:

ALI_INITIATE_ERR	This error message is created, based of the internal generated initiate service.
ALI_REMOTE_ERR	This is the standard error message if the remote station has detected an error during the execution of the received service.
ALI_ABORT_ERR	The remote station has canceled the connection and there is a service active.

The structure of these additional information and their values are defined in the following chapters.

To handle the error message in the user application we suggest the following:

ALI_OK	0x00 No error. Send the next service.
ALI_TEMPORARY_ERR	0x01...0x42 Errors which could be happend temporarily. Please activate the last service once more.
ALI_RECOVERY_ERR	0x43...0x88 Different error reason. You can send an Abort to disconnect the communication relationship and activate the last service once more. This will establish the communication relationship anew if it is possible. Such an error could also happened because an error in the user application or in the driver. Therefore the error number should display or put into a buffer.
ALI_MESSAGE_ERR	0x89...0xBF Error in the message from the user. It makes no sense to reply the service. There is a principle problem or error in the message which comes from the user into the device.May be in relation to actual FMS configuration. You should display the error number, to fix the error in the user application or in the configuration.
All other	0xC0...0xFF System error. Should not happend - if neverthelesshappened please call us.

3.2.1 Connection could not be established - Error 0x80, 128

If the connection is not established then the ALI generates with the first service received from the user an Initiate service on this communication reference. If the remote station answeres with a negative result, then the following error message will be send back to the user.

	Parameter	Type	Define	Value	Description
Message header	msg.rx	unsign char	USER	16	Identification ot the receiver User application (confirmation)
	msg.tx	unsign char	ALI	1	Identification of the sender ALI (confirmation)
	msg.ln	unsign char		9	Message length Number of bytes started at bCommunicationRef
	msg.nr	unsign char		0...255	Message identification Unique number (Invoke ID)
	msg.a	unsign char	TASK_B_11	17	Reply identification Confirmation
	msg.f	unsign char	ALI_INITATE_ERR	0x80	Error number Connection could not be established
	msg.b	unsign char		0	Command Identification Confirmation
	msg.e	unsign char		0	Extention Standard
Telegram header: The telegram header is the same as the belonging command. Not possible for Abort.	bCommunicationRef	unsign char		1...32	Communication reference
	bAccess	unsign char		0	Access mode
	usObjectIdx	unsign short		20...65535	Object index
	bSubObjectIdx	unsign char		0...241	Object subindex
	bDataCnt	unsign char		0...241	Number of user data
	bDataType	unsign char		1-8, 10	Data type of user data
	bService	unsign char		1, 2, 0x80...	Service
	bErrorCode	uint8	See following table		Error code

Initiate Result(-) - structure Msg.Err of MSG_INITIATE

If this error occur the connection is not established.

The Error Codes are defined in accordance EN 50170-6-2:1996, page 438 and are described in the following table.

Error code		Description
ERR_INITIATE_OTHER	0	Other Reason other than any of those identified below.
ERR_INITIATE_MAX_PDU_SIZE_UNSUFFICIENT	1	Max PDU size insufficient The remote-FMS will generate this error, when the configured max. receive-PDU-length is lower then the local max send-PDU-length
ERR_INITIATE_FEATURE_NOT_SUPPORTED	2	Feature not supported The requested service or option is not supported by the server. Check client- and server-services in the Communication Reference list.
ERR_INITIATE_VERSION_OD_INCOMPATIBLE	3	The OD incompatible The versions of the Object Dictionaries (local OD and remote OD) of the communication partners are not compatibel.
ERR_INITIATE_USER_INITIATE_DENIED	4	User Initiate denied The FMS user from the remote station refuses to establish the connection
ERR_INITIATE_PASSWORD	5	Password error There is already a communication relationship established with the same password.
ERR_INITIATE_PROFILE_NUMBER_INCOMPATIBLE	6	Profile number incompatible The client's profile is not supported by the server.

3.2.2 Negative response from Remote Station - Error Type

This is the most frequently error which is generated if the server detects an error during the execution of a service and answers mit a negativ result.

You will find the additional error information in the PROFIBUS standard as Error Type in the Result(-) of all confirmed services.

The server sends the following answer with error to the ALI:

	Parameter	Type	Define	Value	Description
Message header	msg.rx	unsign char	ALI	1	Identification ot the receiver ALI (response)
	msg.tx	unsign char	USER	16	Identification of the sender User application (response)
	msg.ln	unsign char		13...242	Message length Number of bytes started at bCommunicationRef
	msg.nr	unsign char		0...255	Message identification Unique number (Invoke ID)
	msg.a	unsign char	TASK_B_11	17	Reply identification Response
	msg.f	unsign char	ALI_REMOTE_ERR	0x81	Error number Negative response from this server station
	msg.b	unsign char		0	Command Identification Response
	msg.e	unsign char		0	Extention Standard
Telegram header: The telegram header is the same as the belonging command.	bCommunicationRef	unsign char		1...32	Communication reference
	bAccess	unsign char		0	Access mode
	usObjectIdx	unsign short		20...65535	Object index
	bSubObjectIdx	unsign char		0...241	Object subindex
	bDataCnt	unsign char		0...241	Number of user data
	bDataType	unsign char		1-8, 10	Data type of user data
Only possible for these services.	bService	unsign char	TASK_TFC_READ TASK_TFC_WRITE IDX_STATUS_CON IDX_IDENTIFY_CON IDX_READ_CON IDX_WRITE_CON IDX_GET_OD_CON IDX_DEFINE_VAR_LIST_CON IDX_DELETE_VAR_LIST_CON	1 2 0x80 0x81 0x82 0x83 0x84 x087 0x88	Service Read, with same function as other protocols Write, with same function as other protocols Status Identify Read Write GetOD Define Variable List Delete Variable List
Data	chErrorClass	int8		xx	unused
	chErrorCode	int8		xx	unused
	sAddCode	int16		xx	unused
	abAddDescription[ ]	visible	Zero terminated string Length (without termination): 0..228		Additional description User defined

Negative server response to the ALI of all confirmed services

The Client (USER) receives the following confirmation from the ALI:

	Parameter	Type	Define	Value	Description
Message header	msg.rx	unsign char	USER	16	Identification ot the receiver User application (confirmation)
	msg.tx	unsign char	ALI	1	Identification of the sender ALI (confirmation)
	msg.ln	unsign char		13...242	Message length Number of bytes started at bCommunicationRef
	msg.nr	unsign char		0...255	Message identification Unique number (Invoke ID)
	msg.a	unsign char	TASK_B_11	17	Reply identification Confirmation
	msg.f	unsign char	ALI_REMOTE_ERR	0x81	Error number Negative response from the remote station
	msg.b	unsign char		0	Command Identification Confirmation
	msg.e	unsign char		0	Extention Standard
Telegram header: The telegram header is the same as the belonging command.	bCommunicationRef	unsign char		1...32	Communication reference
	bAccess	unsign char		0	Access mode
	usObjectIdx	unsign short		20...65535	Object index
	bSubObjectIdx	unsign char		0...241	Object subindex
	bDataCnt	unsign char		0...241	Number of user data
	bDataType	unsign char		1-8, 10	Data type of user data
Only possible for these services.	bService	unsign char	TASK_TFC_READ TASK_TFC_WRITE IDX_STATUS_CON IDX_IDENTIFY_CON IDX_READ_CON IDX_WRITE_CON IDX_GET_OD_CON IDX_DEFINE_VAR_LIST_CON IDX_DELETE_VAR_LIST_CON	1 2 0x80 0x81 0x82 0x83 0x84 0x87 0x88	Service Read, with same function as other protocols Write, with same function as other protocols Status Identify Read Write GetOD Define Variable List Delete Variable List
Data	chErrorClass	int8		6 8	Error class No additional description from remote station Additional description from remote station
	chErrorCode	int8		0	Error Code
	sAddCode	int16		1..255	Additional Code User defined. This value is taken from msg.f of the server answer.
	abAddDescription[ ]	visible	Zero terminated string Length: 0..229		Additional description User defined

Error Type of Result(-) - structure Msg.Err of all confirmed services

In case of this error the connection will not be closed.

The errors are divided in Error classes with own Error Codes. These are defined in accordance EN 50170-6-2:1996, page 432 and are described in the following table.

Error class		Error code		Description
ERR_TYPE_CLASS_VFD_STATE	1			This error class is returned whenever the state of the VFD is such that the request service my not be executed.
		ERR_TYPE_CODE_OTHER	0	Other error as described below.
ERR_TYPE_CLASS_APP_ REFERENCE	2			This error class is related to the communication relationship on which the service is executed.
		ERR_TYPE_CODE_OTHER	0	Other error as described below
		ERR_TYPE_CODE_APP_UNREACHABLE	1	The application process is not reachable.
ERR_TYPE_CLASS_DEFINITION	3			This Error Class is returned when there are problems with object definitions.
		ERR_TYPE_CODE_OTHER	0	Other error as described below
		ERR_TYPE_CODE_OBJ_UNDEFINED	1	The required objects do not exist.
		ERR_TYPE_CODE_OBJ_ATTRIBUTE_ INCONSISTEND	2	The indicated objects are defined with inconsistent attributes.
		ERR_TYPE_CODE_NAME_ALREADY_EXIST	3	The name exists already.
ERR_TYPE_CLASS_RESOURCE	4			This Error Class is returned when available resources are exceeded.
		ERR_TYPE_CODE_OTHER	0	Other error as described below
		ERR_TYPE_CODE_MEMORY_ UNAVAILABLE	1	There is no more memory for the execution of this service.
ERR_TYPE_CLASS_SERVICE	5			This Error Class is returned whenever there are problems with the service itself.
		ERR_TYPE_CODE_OTHER	0	Other error as described below
		ERR_TYPE_OBJ_STATE_CONFLICT	1	The current state of the object does not permit execution of the service.
		ERR_TYPE_PDU_SIZE	2	Response-PDU > Max-PDU-Sending-Low-PRIO.
		ERR_TYPE_CODE_OBJ_CONSTRAINT_ CONFLICT	3	The execution of the service is not possible at this time.
		ERR_TYPE_PARAMETER_INCONSISTENT	4	The service contains inconsistent parameters.
		ERR_TYPE_ILLEGAL_PARAMETER	5	A parameter has an illegal value.
ERR_TYPE_CLASS_ACCESS	6			This Error Class is returned whenever an access is faulty.
		ERR_TYPE_CODE_OTHER	0	Other error as described below
		ERR_TYPE_CODE_OBJ_INVALIDATED	1	The access refers to a defined object with an undefined Reference attribute.
		ERR_TYPE_CODE_HARDWARE_FAULT	2	The access to the object failed because of a hardware error.
		ERR_TYPE_CODE_OBJ_ACCESS_DENIED	3	The FMS client has not sufficient Access Rights for the object.
		ERR_TYPE_CODE_INVALID_ADDRESS	4	The indicated Address is out of the legal range.
		ERR_TYPE_OBJ_ATTRIBUTE_ INCONSISTENT	5	The attributes of the objects are inconsistent.
		ERR_TYPE_CODE_OBJ_ACCESS_ UNSUPPORTED	6	Object has not been defined for this access.
		ERR_TYPE_CODE_OBJ_NON_EXISTENT	7	The object does not exist.
		ERR_TYPE_CODE_TYPE_CONFLICT	8	The access is rejected because of an incorrect Data Type.
		ERR_TYPE_NAMED_ACCESS_ UNSUPPORTED	9	The access with names is not supported.

Error class		Error code		Description
ERR_TYPE_CLASS_OD	7			This Eror Class is returned whenever an incorrect change to the OD is made.
		ERR_TYPE_CODE_OTHER	0	Other error as described below
		ERR_TYPE_CODE_NAME_LENGTH_ OVERFLOW	1	The legal length of the name is exceeded.
		ERR_TYPE_CODE_OD_OVERFLOW	2	The legal length of the OD is exceeded.
		ERR_TYPE_CODE_OD_WRITE_ PROTECTED	3	The OD is write protected.
		ERR_TYPE_CODE_EXTENSION_LENGTH_ OVERFLOW	4	The legal length of the Extension is exceeded.
		ERR_TYPE_CODE_OD_DESCRIPTION_ LENGTH_OVERFLOW	5	The legal length of a single Object Description is exceeded.
		ERR_TYPE_CODE_OPERATIONAL_ PROBLEM	6	The currently loaded OD is incorrect.
ERR_TYPE_CLASS_OTHER	8			This Error Class is returned to a reason other than any of those listed above.
		ERR_TYPE_CODE_OTHER	0	Other error as described below

3.2.3 Connection Aborted, Error 0x42, 66

If the remote station sends an Abort on an established communication relationship the local ALI create a confirmation with the error ALI_ABORT_ERR for all outstanding services. The reason for the Abort is signalled in the message data as described in the PROFIBUS standard for the Abort indication.

	Parameter	Type	Define	Value	Description
Message header	msg.rx	unsign char	USER	16	Identification ot the receiver User application (confirmation)
	msg.tx	unsign char	ALI	1	Identification of the sender ALI (confirmation)
	msg.ln	unsign char		13...29	Message length Number of bytes started at bCommunicationRef
	msg.nr	unsign char		0...255	Message identification Unique number (Invoke ID)
	msg.a	unsign char	TASK_B_11	17	Reply identification Confirmation
	msg.f	unsign char	ALI_ABORT_ERR	0x42	Error number Connection has been aborted
	msg.b	unsign char		0	Command Identification Confirmation
	msg.e	unsign char		0	Extention Standard
Telegram header: The telegram header is the same as the belonging command. Not possible for Abort:	bCommunicationRef	unsign char		1...32	Communication reference
	bAccess	unsign char		0	Access mode
	usObjectIdx	unsign short		20...65535	Object index
	bSubObjectIdx	unsign char		0...241	Object subindex
	bDataCnt	unsign char		0...241	Number of user data
	bDataType	unsign char		1-8, 10	Data type of user data
	bService	unsign char		1, 2, 0x80...	Service
Data	bInvokeId	uint8			Error code
	bLocallyGenerated	boolean	FALSE TRUE	0x00 0xff	Abort generated remote local
	bAbortIdentifier	uint8	USER_ABT_IDENTIFIER FMS_ABT_IDENTIFIER LLI_ABT_IDENTIFIER FDL_ABT_IDENTIFIER	0 1 2 3	Abort Identifier ALI / User FMS LLI FDL
	bReasonCode	uint8	See following table	0...	Reason Code
	bAbortDetailLen	uns.char		0...16	Length of Abort Detail
	abAbortDetail[16]	octet			Abort Detail Defined by user

Abort Indication - structure Msg.Ind of MSG_ABORT

After receiving an Abort the Communication relationship is closed.
The Abort Identifier defines which Layer has generated the Abort and the Reason Code defines the error itself in accordance EN 50170-5-2:1996, page 308 and EN 50170-6-2:1996, page 684. The Abort can be generated from the local or the remote station, which is signaled in the parameter bLocallyGenerated.
In some case there are additional details available which are defined by the user.

Abort Identifier		Reason Code		Description
USER_ABT_IDENTIFIER	0	USER_ABT_RC1	0	ALI-timer overrun on this communication reference The ALI-timer controls the time between sending a request and receiving the confirmation of this request. Check configuration in KBL.
		USER_ABT_RC2	1	Version OD incompatible The versions of the OD (source OD and remote OD) of the communication partners are not compatible.
		USER_ABT_RC3	2	Password Error There is already a communication relationship established with the same password.
		USER_ABT_RC4	3	Profile Number incompatible The server's profile is not supported by the client.
		USER_ABT_RC5	4	Limited Services Permitted The VFD is in the Logical Status LIMITED-SERVICES-PERMITTED.
		USER_ABT_RC6	5	OD loading interacting The PutOD Service which is not free of interaction is currently active.
FMS_ABT_IDENTIFIER	1	FMS_ABT_RC1	0	FMS CRL Error Faulty FMS CRL Entry.
		FMS_ABT_RC2	1	User Error Improper, unknown or faulty service primitive received from the FMS user.
		FMS_ABT_RC3	2	FMS PDU Error Unknown or faulty FMS PDU received from the LLI.
		FMS_ABT_RC4	3	Connection State Conflict LLI Improper LLI service primitive.
		FMS_ABT_RC5	4	LLI Error Unknown or faulty LLI service primitive.
		FMS_ABT_RC6	5	PDU Size PDU length exceeds maximum PDU length.
		FMS_ABT_RC7	6	Feature not supported SERVICE_REQ_PDU received from LLI and service or option not supported as a server.
		FMS_ABT_RC8	7	Invoke ID Error Response Confirmed service.res received from the FMS user and Invoke ID does not exist.
		FMS_ABT_RC9	8	Max Service Overflow CONFIRMED-SERVICE_REQ_PDU received from the LLI and Outstanding Services Counter Server $\geq$ Outstanding Services Servcer.
		FMS_ABT_RC10	9	Connection State Conflict FMS INITIATE_REQ_PDU or INITIATE_RES_PDU received from the LLI.
		FMS_ABT_RC11	10	Service Error The service in the response does not match the service in the indication or the service in the confimation does not match the service in the request.
		FMS_ABT_RC12	11	Invoke ID Error Request CONFIRMED-SERVICE_REQ_PDU received from LLI and Invoke ID already exists.
		FMS_ABT_RC13	12	FMS disabled FMS is not ready for data transmission.

Abort Identifier		Reason Code		Description
LLI_ABT_IDENTIFIER	2	LLI_ABT_RC1	0	LLI-LLI context-check negative The configuration concerning the CRL of the two partners is inconsistent.
		LLI_ABT_RC2	1	Unallowed LLI PDU received in the connection establishment phase or release phase This error happens often, when the communication-partner has been switched off/on during datatransfer.
		LLI_ABT_RC3	2	Unallowed LLI PDU received in the data transfer phase This error happens often, when the communication-partner has been switched off/on during datatransfer.
		LLI_ABT_RC4	3	Unknown or faulty LLI PDU received
		LLI_ABT_RC5	4	DTA_ACK_PDU received and SAC = 0 Check parameter SAC/RAC in KBL.
		LLI_ABT_RC6	5	Number of parallel services exceeded
		LLI_ABT_RC7	6	Invoke ID unknown
		LLI_ABT_RC8	7	Priority error
		LLI_ABT_RC9	8	Local error in the remote station Read manual of communication-partner. If our firmware sends this error, there is a system-error. Please contact your distributor.
		LLI_ABT_RC10	9	Timer T1 expired (connection establishment) Remote-partner did not answer during a fixed time. If this time is too short, please contact our hotline.
		LLI_ABT_RC11	10	Timer T3 expired (LLI-timer control interval, connection monitoring) No answer received for LLI-IDLE-telegram from remote-partner. Check local and remote LLI-timer in CRL.
		LLI_ABT_RC12	11	RTimer expired (LLI-receive-timer control interval) No LLI-IDLE-telegram from remote-partner received. Check local and remote LLI-timer in CRL.
		LLI_ABT_RC13	12	Error in the LSAP actication
		LLI_ABT_RC14	13	Unallowed FDL primitive in the connection establishment phase or release phase
		LLI_ABT_RC15	14	Unallowed FDL primitive in the data transfer phase
		LLI_ABT_RC16	15	Unknown FDL primitive
		LLI_ABT_RC17	16	Unknown LLI primitive
		LLI_ABT_RC18	17	Unallowed LLI primitive in the connection establishment phase or release phase
		LLI_ABT_RC19	18	Unallowed LLI primitive in the data transfer phase
		LLI_ABT_RC20	19	CRL entry not OK
		LLI_ABT_RC21	20	Conflict case in the connection establishment phhase; local address > remote address This error can only happen on MMAZ-connections. If both masters send at the same time initiate-telegrams, the master with the higher address has to send an abort.
		LLI_ABT_RC22	21	Execution error in cyclic data transfer
		LLI_ABT_RC23	22	Number of parallel services exceeded
		LLI_ABT_RC24	23	CRL is being loaded by FMA7, LLI disabled
		LLI_ABT_RC25	24	Confirm / indication mode error
		LLI_ABT_RC26	25	Unallowed FMA1/2 primitive received
		LLI_ABT_RC27	26	Unallowed FMS service on connection for cyclic data transfer
		LLI_ABT_RC28	27	FSM PDU size exceeded on connection for cyclic data transfer

Abort Identifier		Reason Code		Description
FDL_ABT_IDENTIFIER	3	FDL_ABT_OK	0	No error occured
		FDL_ABT_UE	1	Remote user (LLI) or interface-error
		FDL_ABT_RR	2	Recources of Remote-FDL-not available or exhausted The remote-FDL will send this error-number, when there is no segment or receivebuffer free to receive this telegram. It's a temporary error. By receiving this error, the local LLI will disconnect the connection.
		FDL_ABT_RS	3	Service, remote-address or remote-SAP is by remote-FDL not active
		FDL_ABT_RDL	12	Recources of the remote-FDL-controller/user not available or exhausted
		FDL_ABT_RDH	13	Recources of the remote-FDL-controller/user not available or exhausted
		FDL_ABT_LS	16	Service or local SAP not active
		FDL_ABT_NA	17	No or no logical reaction of remote FDL There is no connection between the local and the remote FDL. Possibilities are, that the stations are not connected, switched off or the bus-parameters are invalid or inconsistent. Check with default-BUS-parameters.
		FDL_ABT_DS	18	Local FDL not in logical token-ring or disconnected from bus
		FDL_ABT_NO	19	No reply data transmitted
		FDL_ABT_LR	20	Recources of the remote-FDL-controller/user not available or exhausted
		FDL_ABT_IV	21	Invalid parameter in request

3.3 Client Services

Client services are divided into requests and confirmations. A request is a demand for a service which is compiled at a client and a confirmation is a response to a request received by the client.

Compilation of the client services is described in this chapter.

3.3.1 Read Request

Read request of an object with index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays or variable list objects)
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	1 / 0x82	Service Read

Read request - structure Msg.Req of MSG_READ

Remark:

- If this service is carried out on a **variable list object**, the parameter *bDataType* has to be set to 10 (Octet string). The parameter *bDataCnt* has to be set to the number of bytes of the requestet user data. The parameter *bSubObjectIdx* has to be set to zero, if the whole variable list object is requested. Otherwise, this parameter refers to an element (object) of the variable list object. The first element of the variable list object has the subindex *bSubObjectIdx* 1.

3.3.2 Read Confirmation

Confirmation of a read request of a variable with index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Confirmation
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays or variable list objects)
	bDataCnt	unsign char	0...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	1 / 0x82	Service Read
Data	abData[ ]	unsign char[ ]		User data

Read confirmation - structure Msg.Con of MSG_READ

Remarks:

- In the case of an error, an additional detail can be transmitted. Such additional details are informative user data transferred in place of the user data. The number of data and their structure depend on the error. For further information see chapter 'Negative response from remote station - Error-Type'.
- If this service is carried out on a **variable list object**, the user data are transfered in a **transparent** way because of the data type octet string. A swapping of the data elements (for example words or long words) must be done by the user application.

3.3.3 Write Request

Write request of a variable with index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays or variable list objects)
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	2 / 0x83	Service Write
	abData[ ]	unsign char[ ]		User data

Write request - structure Msg.Req of MSG_WRITE

Remark:

If this service is carried out on a **variable list object**, the parameter *bDataType* has to be set to 10 (Octet string). The parameter *bDataCnt* has to be set to the number of bytes of the user data. The parameter *bSubObjectIdx* has to be set to zero, if the whole variable list object is sended. Otherwise, this parameter refers to an element (object) of the variable list object. The first element of the variable list object has the subindex *bSubObjectIdx* 1. Because of the data type octet string, the user data are transfered in a **transparent** way. A swapping of the data elements (for example words or long words) must be done by the user application.

3.3.4 Write Confirmation

Confirmation of a write request of a variable with index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Confirmation
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays or variable list objects)
	bDataCnt	unsign char	0...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	2 / 0x83	Service Write

Write confirmation - structure Msg.Con of MSG_WRITE

Remark:

- In the case of an error, an additional detail can be transmitted. Such additional details are informative user data transferred in place of the user data. The number of data and their structure depend on the error. For further information see chapter 'Negative response from remote station - Error-Type'.

3.3.5 Status Request

The device / user state is read with this service.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	0 1...32	Communication reference Local Remote
	bReserved1	unsign char	0	Reserved
	bReserved2	unsign short	0	Reserved
	bReserved3	unsign char	0	Reserved
	bReserved4	unsign char	0	Reserved
	bReserved5	unsign char	0	Reserved
	bService	unsign char	0x80	Service Status

Status request - structure Msg.Req of MSG_STATUS

3.3.6 Status Confirmation

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	11...14	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Confirmation
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	0 1...32	Communication reference Local Remote
	bReserved1	unsign char	0	Reserved
	bReserved2	unsign short	0	Reserved
	bReserved3	unsign char	0	Reserved
	bReserved4	unsign char	0	Reserved
	bReserved5	unsign char	0	Reserved
	bService	unsign char	0x80	Service Status
Data	bLogicalStatus	unsign char	see below	Logical device state
	bPhysicalStatus	unsign char	see below	Physical device state
	bLocalDetailLen	unsign char		Length of Local Detail
	abLocalDeatail[3]	bits		Local Detail

Status confirmation - structure Msg.Con of MSG_STATUS

Parameter	Type	Value		Description
bLogicalStatus	unsign char			Logical device-state
		VFD_STATE_CHANGES_ALLOWED	0	standard state after initialisation
		VFD_LIMITED_SERVICES_PERMITTED	2	not used by firmware
		VFD_OD_LOADING_NON_INTERACTING	4	not used by firmware
		VFD_OD_LOADING_INTERACTING	5	not used by firmware
bPhysicalStatus	unsign char			physical device-state
		VFD_OPERATIONAL	0	ALI initialized, device ready for communication
		VFD_PARTIALLY_OPERATIONAL	1	ALI initialized, device not ready for communication (FDL not in logical token-ring)
		VFD_INOPERATIONAL	2	not used by firmware
		VFD_NEEDS_COMMISIONING	3	ALI (device) not initialised
		VFD_WAIT_ALI_INIT	4	Wait until ALI is initialised
bLocalDetailLen	unsign char		0	Local Detail not available
			3	Local Detail available (not used by firmware)
abLocalDetail[3]	unsign char[3]		0	not used by firmware

3.3.7 Identify Request

Information to identify a VFD is read withthis service.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	0 1...32	Communication reference Local Remote
	bReserved1	unsign char	0	Reserved
	bReserved2	unsign short	0	Reserved
	bReserved3	unsign char	0	Reserved
	bReserved4	unsign char	0	Reserved
	bReserved5	unsign char	0	Reserved
	bService	unsign char	0x81	Service Identify

Identify request - structure Msg.Req of MSG_IDENTIFY

3.3.8 Identify Confirmation

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	107	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Confirmation
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	0 1...32	Communication reference Local Remote
	bReserved1	unsign char	0	Reserved
	bReserved2	unsign short	0	Reserved
	bReserved3	unsign char	0	Reserved
	bReserved4	unsign char	0	Reserved
	bReserved5	unsign char	0	Reserved
	bService	unsign char	0x81	Service Identify
Data	abVendorName[33]	visible		Name of the vendor Null terminated
	abModelName[33]	visible		Name of the model Null terminated
	abRevision[33]	visible		Revision Null terminated

Identify confirmation - structure Msg.Con of MSG_IDENTIFY

3.3.9 Get OD Request

With this request the user can read one or all object description from the local or remote object dictionary. Short and long form is supported. To address the local OD the communication reference 0 must be selected.
If the Access mode startindex is selected then the server sends as much object description as in the PDU fits or available are back.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0..255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	0 1...32	Communication reference Local Remote
	bAccess	unsign char	0x00 0x02 0x80 0x82	Access mode Short form by index Short form by startindex Long form by index Long form by startindex
	usObjectIdx	unsign short	0 1..8, 10 15..32.767 15..65.535	Object index OD object description Static type object of the ST-OD Static object of the S-OD Variable list object DV-OD
	bReserved1	unsign char	0	Reserved
	bReserved2	unsign short	0	Reserved
	bReserved3	unsign char	0	Reserved
	bService	unsign char	0x84	Service Get OD

Get OD request - structure Msg.Req of MSG_GET_OD

3.3.10 Get OD Confirmation

If bMore is set to true then there are more object description available. The number of object description in the message is stated in the parameter *bDescriptionNum*. The object description depends of the type of the object. The type can be taken from the OD object description with index 0. Furthermore, short and long form of object description is possible.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	15.240	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Confirmation
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	0 1...32	Communication reference Local Remote
	bAccess	unsign char	0x00 0x02 0x80 0x82	Access mode Short form by index Short form by startindex Long form by index Long form by startindex
	usObjectIdx	unsign short	0 1..8, 10 15..32.767 15..65.535	Object index OD object description Static type object of the ST-OD Static object of the S-OD Variable list object DV-OD
	bReserved1	unsign char	0	Reserved
	bReserved2	unsign char	0	Reserved
	bReserved3	unsign char	0	Reserved
	bService	unsign char	0x84	Service Get OD
Data	bMore	boolean	0x00 (False) 0xFF (True)	More follows OD description is complete OD description not complete
	bDescriptionNum	unsign char	1..N	Number of following descriptions
	bLen of tDescription 1	unsign char	4..48	Len of tDescription 1 in bytes
	tDescription 1	record		Object description
	bLen of tDescription 2	unsign char	4..48	Len of tDescription 2 in bytes
	tDescription 2	record		Object description
	..	..	..	..
	bLen of tDescription N	unsign char	4..48	Len of tDescription N in bytes
	tDescription N	record		Object description

Get OD confirmation - structure Msg.Con of MSG_GET_OD

After the element *bDescriptionNum*, 1..*N* object description follows. A object description consists of the byte *bLen of tDescription i* and the record *tDescription i* (*i* is the index 1..*N*). The first byte *bLen of tDescription i* contains the length of the following description *tDescripton i* in bytes. The start of the following object description *i*+1 is given by *bLen of tDescription i* + 1. The structure of the records *tDescription* depends of the type of the object and the short or long form. These structures are described below. The given values are valid for Hilscher componets, except the *Variable-List Object* description.

Element	Type	Value	Description
usObjectIdx	unsign short	0	Index
chRomRamFlag	boolean	0x00 (False)	ROM / RAM Flag
bNameLength	unsign char	32	Name Length
chAccessProtectionSupported	boolean	0x00 (False)	Access Protection Supported
sVersionOd	short	1	Version OD
sStOdLength	short	10	ST-OD Length
usFirstIndexSOd	unsign short	15	First Index S-OD
sSOdLength	short	32.767	S-OD Length
usFirstIndexDvOd	unsign short	0	First Index DV-OD
sDvOdLength	short	0	DV-OD Length
usFirstIndexDpOd	unsign short	0	First Index DP-OD
sDpOdLength	short	0	DP-OD Length

Structure of the OD object description, short form

Element	Type	Value	Description
usObjectIdx	unsign short	0	Index
chRomRamFlag	boolean	0x00 (False)	ROM / RAM Flag
bNameLength	unsign char	32	Name Length
chAccessProtectionSupported	boolean	0x00 (False)	Access Protection Supported
sVersionOd	short	1	Version OD
ulLocalAddressOd	UInt32	0xFFFFFFFF	Local Address OD-ODES
sStOdLength	short	10	ST-OD Length
usLocalAddressStOd	UInt32	0xFFFFFFFF	Local Address ST-OD
usFirstIndexSOd	unsign short	15	First Index S-OD
sSOdLength	short	32.767	S-OD Length
ulLocalAddressSOd	UInt32	0xFFFFFFFF	Local Address S-OD
usFirstIndexDvOd	unsign short	0	First Index DV-OD
sDvOdLength	short	0	DV-OD Length
ulLocalAddressDvOd	UInt32	0xFFFFFFFF	Local Address DV-OD
usFirstIndexDpOd	unsign short	0	First Index DP-OD
sDpOdLength	short	0	DP-OD Length
ulLocalAddressDpOd	UInt32	0xFFFFFFFF	Local Address DP-OD

Structure of the OD object description,long form

Element	Type	Value	Description
usObjectIdx	unsign short	1..8, 10	Index
bObjectCode	unsign char	5	Object code

Structure of Data Type Object in the ST-OD, short form

Element	Type	Value	Description
usObjectIdx	unsign short	1..8, 10	Index
bObjectCode	unsign char	5	Object code
abName[32]	unsign char[32]		Variable Name

Structure of Data Type Object in the ST-OD, long form

Element	Type	Value	Description
usObjectIdx	unsign short	15..32.767	Index
bObjectCode	unsign char	4	Object code
usIndexEventData	unsign short	1..8, 10	Index Event Data
bLength	unsign char	1..	Length

Structure of Event Object in the S-OD, short form

Element	Type	Value	Description
usObjectIdx	unsign short	15..32.767	Index
bObjectCode	unsign char	4	Object code
usIndexEventData	unsign short	1..8, 10	Index Event Data
bLength	unsign char	1..241	Length
bPassword	unsign char	0x00	Password
bAccessGroups	unsign char	0xFF	Access Groups
usAccessRights	unsign short	0x00FF	Access Rights
chEnabled	boolean	0xFF	Enabled
abVariableName[32]	unsign char[32]		Variable Name
bExtensionLen	unsign char	0	Extension

Structure of Event Object in the S-OD, long form

Element	Type	Value	Description
usObjectIdx	unsign short	15..32.767	Index
bObjectCode	unsign char	7	Object code
usDataTypeIndex	unsign short	1..8, 10	Data Type Index
bLength	unsign char	1..241	Length

Structure of Simple Variable Object in the S-OD, short form

Element	Type	Value	Description
usObjectIdx	unsign short	15..32.767	Index
bObjectCode	unsign char	7	Object code
usDataTypeIndex	unsign short	1..8, 10	Data Type Index
bLength	unsign char	1..241	Length
bPassword	unsign char	0x00	Password
bAccessGroups	unsign char	0xFF	Access Groups
usAccessRights	unsign short	0x00FF	Access Rights
ulLocalAddress	UInt32	0xFFFFFFFF	Local Address
abVariableName[32]	unsign char[32]		Variable Name
bExtensionLen	unsign char	0	Extension

Structure of Simple Variable Object in the S-OD, long form

Element	Type	Value	Description
usObjectIdx	unsign short	15..32.767	Index
bObjectCode	unsign char	8	Object code
usDataTypeIndex	unsign short	1..8, 10	Data Type Index
bLength	unsign char	1..241	Length
bElementNum	unsign char	1..241	Number Of Elements

Structure of Array Object in the S-OD, short form

Element	Type	Value	Description
usObjectIdx	unsign short	15..32.767	Index
bObjectCode	unsign char	7	Object code
usDataTypeIndex	unsign short	1..8, 10	Data Type Index
bLength	unsign char	1..241	Length
bElementNum	unsign char	1..241	Number Of Elements
bPassword	unsign char	0x00	Password
bAccessGroups	unsign char	0xFF	Access Groups
usAccessRights	unsign short	0x00FF	Access Rights
ulLocalAddress	UInt32	0xFFFFFFFF	Local Address
abVariableName[32]	unsign char[32]		Variable Name
bExtensionLen	unsign char	0	Extension

Structure of Array Object in the S-OD, long form

Element	Type	Value	Description
usObjectIdx	unsign short	15..65.535	Index
bObjectCode	unsign char	10	Object code
bElementNum	unsign char	1..76 *	Number Of Elements
bPassword	unsign char	xx	Password
bAccessGroups	unsign char	xx	Access Groups
usAccessRights	unsign short	xxxx	Access Rights
chDeletable	boolean	0xFF (True)	Deletable
usElementIdx[bElementNum]	unsign short [bElement Num]		Element Index (1) .. Element Index (bElementNum)
abVariableName[32]	unsign char[32]		Variable List Name
bExtensionLen	unsign char	0 **	Extension

Structure of Variable-List Object in the DV-OD, long form ***

Element	Type	Value	Description
usObjectIdx	unsign short	15..65.535	Index
bObjectCode	unsign char	10	Object code
bElementNum	unsign char	1..76 *	Number Of Elements
chDeletable	boolean	0xFF (True)	Deletable
usElementIdx[bElementNum]	unsign short [bElement Num]		Element Index (1) .. Element Index (bElementNum)

Structure of Variable-List Object in the DV-OD, short form ***

* This value is given by the maximum PDU size

** Only zero if there is no profile specific information

*** Variable-List Objects are not supported from the OD of Hilscher components. They are supported only as client functionallity. This means, Variable-List Objects in the DV-OD of a remote partner can be read.

3.3.11 Define Variable List Request

Define Variable List request of objects with index. Define a Variable List object consisting of the objects given by index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	10..108	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usReserved1	unsign short	0	Reserved
	bReserved2	unsign char	0	Reserved
	bDataCnt	unsign char	1...50	Number of objects Number of objects to define in the variable list
	bDataType	unsign char	6	Data type of user data Unsigned Integer 16 Bit
	bService	unsign char	0x87	Service Define Variable List
Data	ausObjectList[]	unsign short[]		List of objects to define in the Variable List object.

Define Variable List request - structure Msg.Req of MSG_DEFINE_VARIABLE_LIST

3.3.12 Define Variable List Confirmation

Define Variable List confirmation of objects with index. Define a Variable List object consisting of the objects given by index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	10	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Confirmation
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usReserved1	unsign short	0	Reserved
	bReserved2	unsign char	0	Reserved
	bDataCnt	unsign char	1..76	Number of user data Number of objects to define in the variable list
	bDataType	unsign char	6	Data type of user data Unsigned Integer 16 Bit
	bService	unsign char	0x87	Service Define Variable List
Data	usVariableListIndex	unsign short		Index of the Variable List object

Define Variable List confirmation - structure Msg.Con of MSG_DEFINE_VARIABLE_LIST

3.3.13 Delete Variable List Request

Delete Variable List request of objects with index. Delete a Variable List object given by the variable list index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	10	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usReserved1	unsign short	0	Reserved
	bReserved2	unsign char	0	Reserved
	bDataCnt	unsign char	1	Number of Variable List indizes
	bDataType	unsign char	6	Data type of user data Unsigned Integer 16 Bit
	bService	unsign char	0x88	Service Delete Variable List
Data	usVariableListIndex	unsign short		Variable List index

Delete Variable List request - structure Msg.Req of MSG_DELETE_VARIABLE_LIST

3.3.14 Delete Variable List Confirmation

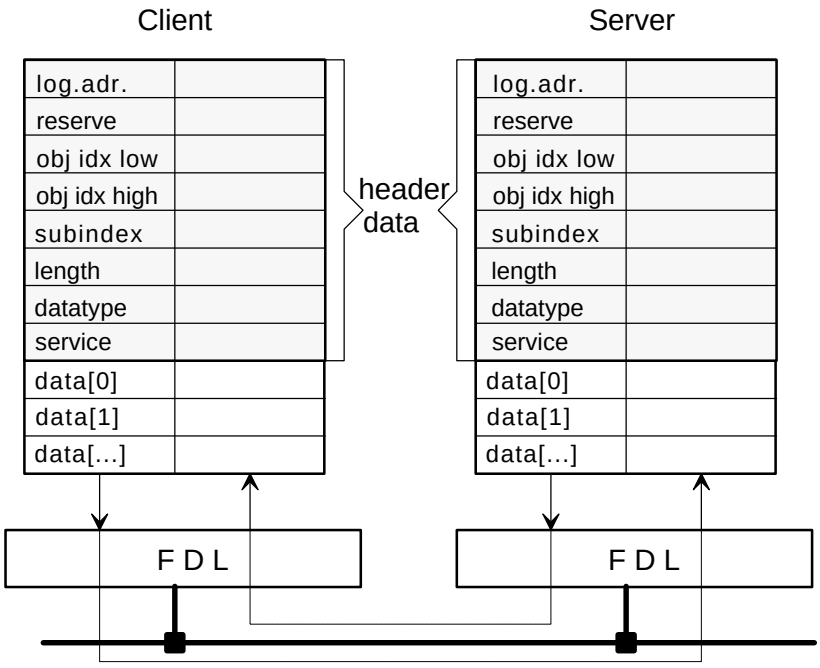
Delete Variable List confirmation of objects with index. Delete a Variable List object given by the variable list index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Confirmation
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usReserved1	unsign short	0	Reserved
	bReserved2	unsign char	0	Reserved
	bDataCnt	unsign char	1	Number of Variable List indizes
	bDataType	unsign char	6	Data type of user data Unsigned Integer 16 Bit
	bService	unsign char	0x88	Service Delete Variable List

Delete Variable List confirmation - structure Msg.Con of MSG_DELETE_VARIABLE_LIST

3.4 Server Services

Server services are divided into indications and responses. An indication is a receiving demand that the server has to reply, and a response is the reply that the server has created.



3.4.1 Read Indication

A read indication of a variable with index arrived.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Indication
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays)
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	1 / 0x82	Service Read

Read indication - structure Msg.Ind of MSG_READ

3.4.2 Read Response

Reply to a read indication of a variable with index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Response
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays)
	bDataCnt	unsign char	0...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	1 / 0x82	Service Read
Data	abData[ ]	unsign char[ ]		User data

Read response - structure Msg.Res of MSG_READ

For the response, the telegram header beginning with *bCommunicationRef* is taken from the indication without change. In the case of the message header the parameters *msg.rx* / *msg.tx* and *msg.a* / *msg.b* are swapped. The parameter *msg.ln* has to adapted on the size of the User data.

If an error number is returned in *msg.f*, the PROFIBUS sends this to the partner device in the form of an additional detail.

3.4.3 Write Indication

A write indication of a variable with index arrived.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Indication
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays)
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	2 / 0x83	Service Write
Data	abData[ ]	unsign char[ ]		User data

Write indication - structure Msg.Ind of MSG_WRITE

3.4.4 Write Response

Reply to a write indication of a variable with index.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	8	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification Response
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays)
	bDataCnt	unsign char	0...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	2 / 0x83	Service Write

Write response - structure Msg.Res of MSG_WRITE

For the response, the telegram header beginning with *bCommunicationRef* is taken from the indication without change. In the case of the message header the parameters *msg.rx* / *msg.tx* and *msg.a* / *msg.b* are swapped. The parameter *msg.ln* must be set to 8 according the size of the telegram header.

If an error number is returned in *msg.f*, the PROFIBUS sends this to the partner device in the form of an additional detail.

3.5 Unconfirmed Client Services

Unconfirmed services receive no acknowledgement from the communications partner on the FMS/ALI levels. On the user level, the user himself can decide whether an acknowledgement is to take place or not. This acknowledgement does not however guarantee transmission of the service, but only its receipt by the ALI. This facilitates uniform processing of the messages by the user.

3.5.1 Abort Request

The abort request service is used to terminate a communications relationship with a remote partner. On transmission of an abort, therefore, an open communication reference through which the data can be transmitted is closed. An acknowledgement of the service is sent to the user by the local ALI.

Please note that you must also delete all open services in your internal message managment if you send an Abort to the ALI.

If there are parallel services on this communication reference, they will be erased without a response to the user.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	10...26	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bReserved1	unsign char	0	Reserved
	bReserved2	unsign short	0	Reserved
	bReserved3	unsign char	0	Reserved
	bReserved4	unsign char	0	Reserved
	bReserved5	unsign char	0	Reserved
	bService	unsign char	0XA5	Service Abort
Data	bReasonCode	uint8	see table	Reason of abort
	bAbortDetailLen	uns.char	0...16	Number of data of abort detail
	abAbortDetail[16]	octet		Abort detail Defined by user

Abort request - - structure Msg.Req of MSG_ABORT

3.5.2 Send Information Report

Sends a write request for a variable with index.

The service corresponds to a write request without confirmation. This service can be transmitted on open links like Broadcast or Multicast. The service is not permissible on cyclical links.

This service is not supported by FDL communication.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays)
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	0x9F	Service Information report
Data	abData[ ]	unsign char[ ]		User data

Information Report request - structure Msg.Req of MSG_INFORMATION_REPORT

3.5.3 Send Event Notification

Objects from the device's **own** object directory can be sent with this service. Events at local objects are signalled. The object index in the request block therefore references an object in the station's own object directory and not, as usual, an object in the server's object directory.

The ALI does not maintain any data *contents* for the objects defined in the OD. For this reason, the event notification job contains the data to be sent with the object.

This service can be transmitted on open links such as Broadcast or Multicast. The service is not permissible on cyclical links. The event notification service is an unconfirmed service and is not acknowledged by the remote partner!

The service is not supported by FDL communications.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification This number is also used as event number
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index Attention: In this service the object index refer to the local object directory
	bSubObjectIdx	unsign char	0	Object subindex Whole object
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	0xA1	Service Event notification
Data	abData[ ]	unsign char[ ]		User data

Event Notification request - structure Msg.Req of MSG_EVENT_NOTIFICATION

3.6 Unconfirmed Server Services

An unconfirmed service arriving at the server is not acknowledged on the FMS/A-LI level. On the user level, the user can decide himself whether he wishes to compile an acknowledgement in order to implement uniform processing of the messages. If the user sends an acknowledgement, this is not further analysed by the ALI.

3.6.1 Receive Information Report

Receive write job for a variable with index.

The service corresponds to an unconfirmed write, i.e. a write indication that is not acknowledged. This service can be transmitted on open links such as Broadcast or Multicast. The service is not permissible on cyclical links.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Indication
	msg.e	unsign char	2	Extention No reply
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0 1...241	Object subindex Whole object Access by subindex (only arrays)
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	1...8, 10	Data type of user data
	bService	unsign char	0x9F	Service Information report
Data	abData[ ]	unsign char[ ]		User data

Information Report indication - structure Msg.Ind of MSG_INFORMATION_REPORT

3.6.2 Receive Event Notification

With this job, data are received from an object defined in a remote object directory. The object index in the request block therefore references an object in the remote object directory and not, as usual, an object in the local object directory.

This service can be transmitted on open links such as Broadcast or Multicast. The service is not permissible on cyclical links.

The service is not supported by FDL communication.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	9...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Indication
	msg.e	unsign char	2	Extention No reply
Telegram header	bCommunicationRef	unsign char	1...32	Communication reference
	bAccess	unsign char	0	Access mode By index
	usObjectIdx	unsign short	20...65535	Object index
	bSubObjectIdx	unsign char	0	Object subindex Whole object
	bDataCnt	unsign char	1...241	Number of user data
	bDataType	unsign char	10	Data type of user data Only octet string, because the data type is not know at the server. Attention: The abData[ ] are in the same order as on the PROFIBUS. They are not swapped in little endian mode!
	bService	unsign char	0xA1	Service Event Notification
Data	abData[ ]	unsign char[ ]		User data

Event Notification indication - structure Msg.Ind of MSG_EVENT_NOTIFICATION

3.7 Data Transmission by FDL-Communication

In contrast to FMS data transfer, no message contents are defined in FDL data transfer. The FDL cycle describes only the transfer of data from one device to another and the acknowledge of correct receiving. This is realized with the **FDL service 'SDA'** (Send Data with Acknowledge). There are no functions like read or write, only send. Therefore our PROFIBUS-firmware offers two interfaces for FDL-communication, which can be configured for each communications reference. So it is possible to run at the same time on one device data transmission on FMS, FDL-transparent and FDL-defined. The last ones will be described next chapters.

3.7.1 Transparent FDL Data Transmission

The transparent FDL data transmission sends a data-block from one PROFIBUS device to another. The contents of the data does not matter. If the partner is a PLC, mostly the data arrives in a receive data block (DB). If the device should receive data from a PLC, the PLC has to invoke the data transmission. Mostly the PLC puts the data in a datablock and sends it by using a SEND function block (SEND-FB). The device receives this data as user data.

3.7.2 Defined FDL Data Transmission

Because there is a difference between responding to the FMS- and FDL transparent - interfaces, we implemented the FDL-defined interface with the aim of a uniform user interface for FMS and FDL. These concern on the one hand the sequence of FDL communications and on the other hand a uniform message header data structure. The communicating station, for example a PLC S5-95U, has to construct an image of this sequence and evaluate the header data structure in the manner described.

Our FDL-defined user interface corresponds to that of the FMS. All data from the "logical address" parameter onwards are transferred to the communicating station.

The following table shows the general structure of the FDL message data. The header data must be present in every message!

Message data in FDL-defined data transfer

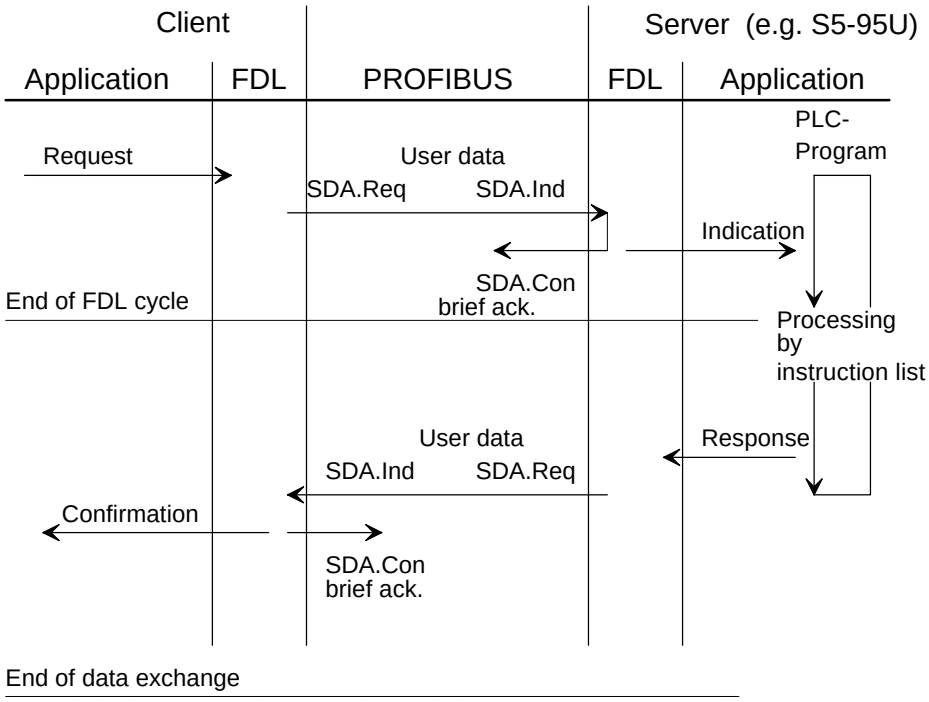
The functions of the parameters and values are shown in the following table:

parameter	description
log_adr	communications reference
reserved	reserved
objIdx low	data block (DB)
objIdx high	reserved
subindex	first data word (DW) to read/write
length	number of datawords to read/write
datatype	data type from PROFIBUS convention
service*	PROFIBUS service: 1 - read 2 - write
data[ ]	user data

Functions of the FDL-defined data
**In the response the bit 6 (0x40) of the parameter 'service' has to be set!*

Notice: In the response the bit 6 (0x40) of the parameter 'service' has to be set.

The time sequence for FDL data transfer looks as follows:



Time sequence of FDL data transfer

In the FDL - link between Siemens devices, an FDL send cycle is terminated on arrival of a brief acknowledgement from the remote FDL. This brief acknowledgement indicates that the data have been successfully transferred from A to B. From the point of view of the PROFIBUS-FMS/FDL user, however, the acknowledgement on the "higher" level, i.e. the confirmation, is missing. The confirmation is transmitted by the remote partner in a new FDL cycle. The message data for the confirmation/response are generated by the server from the indication received

In order to facilitate the distinction between a request and a response, Bit 6 in the service byte for the response is set. Bit 6 is automatically operated by the ALI in our PROFIBUS software, whereas it has to be operated by the FDL application program in the PLC or other PROFIBUS-FDL partners. See chapter Example of FDL cummunications with PLC S5-95U.

In contrast to FMS data transfer, no message contents are defined in FDL data transfer. The FDL-cycle describes only the transfer of data from one device to another and the acknowledge of correct receiving. This is realized with the **FDL-service 'SDA'** (Send Data with Acknowledge). There are no functions like read or write, only send. The transparent FDL data transmission sends a data-block from one PROFIBUS-device to another. The contents of the data does not matter.

3.7.3 Sending Data by FDL-Transparent

With this service the user data in the array abData[ ] will be sand as FDL-telegram over the PROFIBUS.
After the partner station received the data without error, the service will be acknowledged on FDL-Layer.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	8...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification Request
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1..32	Communication reference
	bAccess	unsign char	0	Access mode Not used
	usObjectIdx	unsign short	0	Object index Not used
	bSubObjectIdx	unsign char	0	Object subindex Not used
	bDataCnt	unsign char	0..241	Number of user data
	bDataType	unsign char	10	Data type of user data Only octet string valid
	bService	unsign char	2	Service Write
Data	abData[ ]	unsign char[ ]		User data

FDL-transparent send telegram

3.7.4 Reply to a FDL-Transparent Send Telegram

This reply tells, if the remote-partner received the data with or without error.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	8...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	17	Reply identification
	msg.f	unsign char	0 1...255	Error number No error Error number
	msg.b	unsign char	0	Command identification No command
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1..32	Communication reference
	bAccess	unsign char	0	Access mode Not used
	usObjectIdx	unsign short	0	Object index Not used
	bSubObjectIdx	unsign char	0	Object subindex Not used
	bDataCnt	unsign char	0..241	Number of user data
	bDataType	unsign char	10	Data type of user data Only octet string valid
	bService	unsign char	2	Service Write

FDL-transparent reply telegram

3.7.5 Receiving Data by FDL-Transparent

When the ALI sends an FDL-transparent receive telegram to the user, the ackowledge to the communications partner already has been sent. So it is not possible to the userprogram to reject the telegram with an error.

A reply to this telegram will be deleted by the ALI.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	16	Identification ot the receiver User application
	msg.tx	unsign char	1	Identification of the sender ALI
	msg.ln	unsign char	8...249	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	17	Command identification
	msg.e	unsign char	0	Extention Standard
Telegram header	bCommunicationRef	unsign char	1..32	Communication reference
	bAccess	unsign char	0	Access mode Not used
	usObjectIdx	unsign short	0	Object index Not used
	bSubObjectIdx	unsign char	0	Object subindex Not used
	bDataCnt	unsign char	0..241	Number of user data
	bDataType	unsign char	10	Data type of user data Only octet string valid
	bService	unsign char	2	Service Write
Data	abData[ ]	unsign char[ ]		User data

FDL-transparent receive telegram

3.8 Device Status

A PROFIBUS standard status field is maintained in the PROFIBUS device. The status field can be read by the remote partners with the PROFIBUS service "status".

After start up of the firmware, the initialisation status of the application layer (ALI) is entered in the status. Depending on the configuration, the ALI goes into *VFD_PARTIALLY_OPERATIONAL* status until the FDL signals inclusion in the token ring. In this state, all send jobs are acknowledged by the user with an error message.

After initialization, the status is no longer written by the PROFIBUS software. ON arrival of the PROFIBUS "STATUS" service, the ALI accesses this status field without informing the application.

Services are available to the local application program for reading or writing of the device status.

The status field has the following structure:

Parameter	Type	Value		Description
bLogicalStatus	unsign char			Logical device-state
		VFD_STATE_CHANGES_ALLOWED	0	standard state after initialisation
		VFD_LIMITED_SERVICES_PERMITTED	2	not used by firmware
		VFD_OD_LOADING_NON_INTERACTING	4	not used by firmware
		VFD_OD_LOADING_INTERACTING	5	not used by firmware
bPhysicalStatus	unsign char			physical device-state
		VFD_OPERATIONAL	0	ALI initialized, device ready for communication
		VFD_PARTIALLY_OPERATIONAL	1	ALI initialized, device not ready for communication (FDL not in logical token-ring)
		VFD_INOPERATIONAL	2	not used by firmware
		VFD_NEEDS_COMMISIONING	3	ALI (device) not initialised
		VFD_WAIT_ALI_INIT	4	Wait until ALI is initialised
bLocalDetailLen	unsign char		0	Local Detail not available
			3	Local Detail available (not used by firmware)
abLocalDetail[3]	unsign char[3]		0	not used by firmware

Structure of PROFIBUS-device state

3.8.1 Set the VFD Status

The status is set by the user with the command VFD_SET_STATUS. A response acknowledgement is not generally provided, but can be required to facilitate uniform processing of the messages by the user.

	Parameter	Type	Value	Description
Message header	msg.rx	unsign char	1	Identification ot the receiver ALI
	msg.tx	unsign char	16	Identification of the sender User application
	msg.ln	unsign char	3...6	Message length
	msg.nr	unsign char	0...255	Message identification
	msg.a	unsign char	0	Reply identification No reply
	msg.f	unsign char	0	Error number No error
	msg.b	unsign char	0x40	Command Identification ALI_B_SET_VFD_STATUS
	msg.e	unsign char	0 2	Extention Standard No reply expected
Data	bLogicalStatus	unsign char	0,2,4,5 see table above	Logical device-state
	bPhysicalStatus	unsign char	0..3 see table above	Physical device-state
	bLocalDetailLen	unsign char	0 3	Length of LocalDeatail Local Detail not existing Local Detail existing
	abLocalDetail[3]	unsign char[3]		User specific

Command ALI_B_SET_VFD_STATUS - structure MSG_SET_FVD_STATUS

4 Scope of Functions of the PROFIBUS-FMS/FDL Implementation

In the PROFIBUS standard, the functions of the PROFIBUS are divided into those which every PROFIBUS station must be able to perform and those which may be available if so required of the device. As the scope of these functions may deviate from device to device, this chapter deals with the scope of functions of our implementation.

4.1 Scope of Functions of Layer 2 - FDL

Scope of functions in accordance with EN 50170 part 2:

- active/passive station
- data rates of 9.6, 19.2, 93.75, 187.5, 500 kBit/s, 1.5, 3, 6 and 12 MBit/s.
- net data length up to 246 Bytes
- SDA, SRD, CSRD and SDN services
- management services.

All transmission services and an adapted selection of the management services are supported. The FDL communications services are not directly available to the user.

4.2 Scope of Functions of Layer 7 - LLI

The number of configurable communications references is 32. A maximum of 4 parallel services per communications reference is permissible.
The LLI provides the functions for

- acyclical master - master,
- acyclical master - slave,
- cyclical master - slave,
- Broadcast and
- Multicast

communications.

4.3 Scope of Functions of Layer 7 - FMS

All the FMS services described as 'mandatory' in EN 50170 are currently available, supplemented by READ, WRITE, EVENT NOTIFICATION, INFORMATION REPORT, STATUS, IDENTIFY and GET_OD.
As a result of the modular software structure, further services can be added within a short time.

Available services:

- Initiate - Establishment of a connection
- Read - Reading of an object
- Write - Writing of an object
- Event Notification - Notification of an event
- Information Report - Unconfirmed write
- Abort - Shutdown of the connection
- Reject - Rejection of a service
- Status - Device status
- Identify - Device identification
- Get-OD - Reading of an OD entry
- Define Variable List Define a variable list object (only client)
- Delete Variable List Delete a variable list object (only client).