SIEMENS

SIMATIC NET

FDL Programming Interface

Volume 1 of 1

SIMATIC NET is a trademark of Siemens
Siemens Aktiengesellschaft

SIEMENS

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so daß wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in der Druckschrift werden jedoch regelmäßig überprüft. Notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.
Technische Änderungen vorbehalten.

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

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

Nous nous réservons le droit de modifier les caractéristiques techniques.

SIEMENS

SIMATIC NET

FDL Programming Interface

Description   C79000-B8976-C072/05

Note

We would point out that the contents of this product documentation shall not become a part of or modify any prior or existing agreement, commitment or legal relationship. The Purchase Agreement contains the complete and exclusive obligations of Siemens. Any statements contained in this documentation do not create new warranties or restrict the existing warranty.

We would further point out that, for reasons of clarity, these operating instructions cannot deal with every possible problem arising from the use of this device. Should you require further information or if any special problems arise which are not sufficiently dealt with in the operating instructions, please contact your local Siemens representative.

General

This device is electrically operated. In operation, certain parts of this device carry a dangerously high voltage.

WARNING !

Failure to heed warnings may result in serious physical injury and/or material damage.

Only appropriately qualified personnel may operate this equipment or work in its vicinity. Personnel must be thoroughly familiar with all warnings and maintenance measures in accordance with these operating instructions.

Correct and safe operation of this equipment requires proper transport, storage and assembly as well as careful operator control and maintenance.

Personnel qualification requirements

Qualified personnel as referred to in the operating instructions or in the warning notes are defined as persons who are familiar with the installation, assembly, startup and operation of this product and who posses the relevant qualifications for their work, e.g.:

− Training in or authorization for connecting up, grounding or labelling circuits and devices or systems in accordance with current standards in safety technology;

− Training in or authorization for the maintenance and use of suitable safety equipment in accordance with current standards in safety technology;

− First Aid qualification.

FDL Programming Interface

This manual describes the FDL programming interface and the services available on the PROFIBUS Layer 2.

The FDL protocol (**F**ield **D**ata **L**ink protocol) is suitable for PROFIBUS in the open, heterogeneous SIMATIC NET communication system for the cell and field area and is specially intended for an industrial environment.

Known under the name PROFIBUS (**Pro**cess **Fi**eld **Bus**), PROFIBUS is based on the PROFIBUS EN 50 170 Vol. 2 standard and is oriented on the ISO/OSI reference model.

By complying with the requirements of EN 50 170 Vol. 2, PROFIBUSguarantees an open system for the attachment of components of other vendors that comply with the standard.

PROFIBUS is the network for the middle range of performance. The maximum off 127 stations that can be connected opens up a wide spectrum of automation tasks. Different data rates can be selected using the software.

1   Introduction to the FDL Programming Interface

This chapter introduces the concept of the FDL programming interface. It explains the basic mechanisms you require to be able to program an application.

You require the information in this chapter as a basis for the chapters that follow.

Introduction   The program that uses the layer 2 services is known as an FDL application.

An FDL application can be created in the programming language C or C++. To allow access to the CP, include files and libraries are supplied on the diskette.
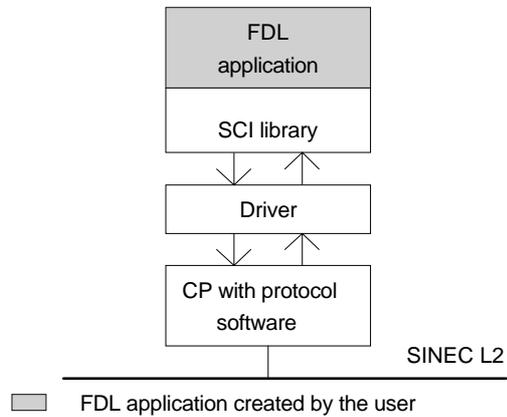
Fig. 1.1: Communications Architecture of the FDL Programming Interface

Structure of Layer 2   The layer 2 protocol software of PROFIBUS can be divided into three entities, FLC (Fieldbus Link Control), FMA (Fieldbus Management), MAC (Media Access Control).
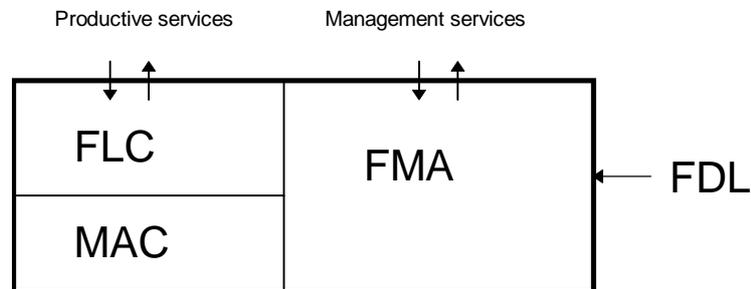
Fig. 1.2: Structure of the Layer 2 Protocol Software (FDL)

Using the FLC and FMA entities, the FDL application can transfer jobs to layer 2 that, if applicable, are passed on to the physical medium by the MAC entity. In the opposite direction, the MAC entity receives frames on the bus that can then be transferred to the FLC or FMA entities of the FDL application.

FLC   The FLC entity is responsible for receiving the services on the FDL programming interface as described in PROFIBUS (data transfer services, send and receive frames). The jobs of the FDL application are received, processed (frame processing etc.) and if applicable, passed on to the MAC entity via an internal interface.

FMA   The FMA entity is responsible for receiving the management services described in PROFIBUS (administrative services, parameter assignment, modifications to operating parameters etc.).

MAC   The MAC entity implements the complete bus access management according to DIN 19245 Part 1.

Mechanism of the Interface

The FDL programming interface uses request blocks (job blocks, RB) for processing jobs with the CP. A request block completely describes a job for the FDL programming interface. The request block is transferred to the CP with one of the functions of the SCI library and is then fetched later by a different function.

Request Block ID

In general, the FDL application transfers a request block to layer 2 with the **request** ID and, depending on the service, receives a request block back with the **confirm** ID or with the **indication** ID.

Request Block ID	Task of the Request Block
request	Jobs from the FDL application for the CP.
confirm	Acknowledgment of a request from the CP to the FDL application.
indication	Indication of an event from the CP to the FDL application.

Notes

2 The FDL Services

This chapter explains which services the FDL protocol provides for communication with other stations on the bus.

Description of the FDL Services

The services of layer 2 can be divided into productive services and management services. Productive services are used in the productive phase to send data frames. Management services are used to activate/deactivate local SAPs (Service Access Point), to provide resources for receiving data frames from other stations and other administrative services. The following tables contain an overview of the available FDL services.

Productive Services

SDA (send and request data with acknowledge)	The CP sends a data frame to a remote station. If this is successful, the addressed remote station returns an acknowledgment. If an error occurs, layer 2 generates a local error message.
SDN (send data with no acknowledge)	The CP sends a data frame to one or more remote stations. In contrast to the SDA service, the addressed remote stations do not return an acknowledgment. After the frame is sent, layer 2 generates a local acknowledgment.
SRD (send and request data)	The CP sends a data frame to a remote station. If this is successful, the addressed remote station returns an acknowledgment. In contrast to the SDA service, the remote station can also send user data in the reply frame. If an error occurs, layer 2 generates a local error message.
REPLY_UPDATE_SINGLE	With this call, data is transferred to layer 2 that can be read out by a remote station. The data is sent back to a remote station in the acknowledgment of an SRD frame. The data can only be read by the remote station **once**.
REPLY_UPDATE_MULTIPLE	Sequence as for REPLY_UPDATE_SINGLE. Difference: The data transferred to layer 2 can be read **more than once** by remote stations.

Management Services	SAP_ACTIVATE	With this service, an SAP (Service Access Point) can be activated at layer 2. This must be activated before data frames can be sent or received.
	RSAP_ACTIVATE	Corresponds to the SAP_ACTIVATE call Difference: With this RSAP_ACTIVATE service, an SAP cannot be initialized for active sending of data frames.
	SAP_DEACTIVATE	With this service, an SAP activated with (R)SAP_ACTIVATE can be deactivated again. Following this, no further data transfer is possible with this SAP.
	AWAIT_INDICATION	With this service, a receive buffer can be transferred to an SAP. Only then is it possible for a call frame (SDA, SDN, SRD) to be received from a remote station. After receiving a remote call frame, a new receive buffer must be transferred to the SAP.
	WITHDRAW_INDICATION	With this service, receive buffers transferred to an SAP with AWAIT_INDICATION, can be fetched back.
	LSAP_STATUS	This service checks the configuration of an SAP of the local station.
	FDL_IDENT	This service checks the identification of the local or a remote station.
	FDL_LIFE_LIST_CREATE_LOCAL	This service provides a list of **active** and some of the **passive** stations on the bus. The list is generated using only local information within layer 2. No additional frames are sent on the bus.
	FDL_LIFE_LIST_CREATE_REMOTE	This service provides a list of the active and passive stations on the bus. In contrast to FDL_LIFE_LIST_CREATE_LOCAL a status frame is requested from all possible stations (extra load on the bus).
	FDL_READ_STATISTIC_ CTR	This service is used to read out bus-specific statistical values (invalid frames etc.).
	FDL_EVENT	With this service, the FDL application is informed of layer 2 events.
	FDL_READ_VALUE	With this service, the current parameter assignment data of layer 2 can be read out.

2.1   Transfer Mechanisms

Transfer to the FDL Application

An FDL application communicates with layer 2 using three different transfer mechanisms:

1)   FDL application request to layer 2

2)   Layer 2 confirmation to the FDL application

3)   Layer 2 indication to the FDL application

Request

A request involves the FDL application requesting a service at layer 2. The request is transferred to the CP with an SCP_send call (see Section 7.2) or ihi_write call (see Section 6.2).

The call parameters of the function are a 'handle' and the address of the pointer that points to a request block structure *). The entries must be made in the request block before the call in keeping with the service description.

The return value of the SCP_send or ihi_write call relates only to the correct transfer of the request to layer 2 by the driver.

Only the corresponding layer 2 confirmation shows whether a request was processed without errors by layer 2.

Confirmation

In a confirmation, layer 2 informs the FDL application of the result of a processed request. The confirmation must be read out using an SCP_receive call (see Section 7.3) or an ihi_read call (see Section 6.3) to the CP.

The return value of the SCP_receive or ihi_read call relates only to the correct transfer of the data to the driver. The result of the request processing is contained in the request block returned with ihi_read.

Indication

Using an indication, layer 2 informs the FDL application that a call frame (SDA, SRD or SDN) has been received from a remote station. The indication must be read out with an SCP_receive call or an ihi_read call to the CP *).

The return value of the SCP_receive call or ihi_read call relates only to the correct transfer of the data to the driver. The type and content of the indication is contained in the request block that is returned with SCP_receive or ihi_read.

*) see 3.1 Data Structures of the Productive Services

Requester and Responder

Requester   The station that triggers job processing and waits to receive the confirmation.

Responder   The station that receives a data frame from a remote station and returns an acknowledgment frame.

The following table shows the possible transfer mechanisms for the available productive and management services for requesters and responders.

Service	Requester		Responder
	Request	Confirmation	Indication
SDA (Send data with acknowledge)	yes	yes	yes
SDN (Send data with no acknowledge)	yes	yes	yes
SRD (Send and request data)	yes	yes	yes
REPLY UPDATE SINGLE	yes	yes	no
REPLY UPDATE MULTIPLE	yes	yes	no
SAP ACTIVATE	yes	yes	no
RSAP ACTIVATE	yes	yes	no
SAP DEACTIVATE	yes	yes	no
AWAIT_INDICATION	yes	Success: no Error: yes	no
WITHDRAW_INDICATION	yes	yes	no
LSAP_STATUS	yes	yes	no
FDL_IDENT	yes	yes	no
FDL_LIFE_LIST_CREATE_ REMOTE	yes	yes	no
FDL_LIFE_LIST_CREATE_LOCAL	yes	yes	no
FDL_READ_STATISTIC_ COUNTER	yes	yes	no
FDL_EVENT	no	no	yes
FDL_READ_VALUE	yes	yes	no ⸙

Notes

3   Productive Services

This chapter explains the principles of productive services.

The chapter covers the following topics:

➢   The data structures of the productive services

➢   The request blocks of the productive service

The following productive services are dealt with in detail:

➢   SDA (send data with acknowledge)

➢   SDN (send data with no acknowledge)

➢   SRD (send and request data)

➢   REPLY_UPDATE_SINGLE

➢   REPLY_UPDATE_MULTIPLE

3.1   Data Structures of the Productive Services

Request Block Structure

The data structures are defined in the **"fdl_rb.h" include file**.

The "fdl_rb" structure described below is the request block assigned to the ihi functions as a parameter.

```
typedef struct
{
  rb2_header_type            rb2_header;
  struct application_block application_block;
  UBYTE   reserved  [12];
  UBYTE   reference [2];
  UBYTE   user_data_1 [260];
  UBYTE   user_data_2 [260];
} fdl_rb;
```

Description of the Parameters

rb2_header	Request block header. General, non-service-specific parameters
application_block	Argument area. FDL parameters.
reference	ID of the FDL application.
user_data_1	User data, dependent on particular job.
user_data_2	User data, dependent on particular job.

Substructure Request Block Header

```
typedef  struct
{
  UWORD      reserved [2];
  UBYTE      length;
  UWORD      user;
  UBYTE      rb_type;
  UBYTE      priority;
  UBYTE      reserved_1;
  UWORD      reserved_2;
  UBYTE      subsystem;
  UBYTE      opcode;
  UWORD      response;
  UWORD      fill_length_1;
  UBYTE      reserved_3;
  UWORD      seg_length_1;
  UWORD      offset_1;
  UWORD      reserved_4;
  UWORD      fill_length_2;
  UBYTE      reserved_5;
  UWORD      seg_length_2;
  UWORD      offset_2;
  UWORD      reserved_6;
} rb2_header_type;
```

Description of the Parameters	length	Length of the request block without "user_data_1" and "user_data_2" (= 80 bytes).
	user	Available for the FDL application
	rb_type	Type of request block used (= 2).
	priority	Priority of the job (identical to the "serv_class" parameter in the application block).
	subsystem	Selects the communications layer. (FDL = 22h).
	opcode	Request, confirm, indication (same as the "opcode" parameter in the application block).
	response	Return parameter (same as the "l_status" parameter in the application block).
	fill_length_1	Number of relevant bytes in data buffer 1.
	seg_length_1	Actual length of data buffer 1.
	offset_1	Offset of data buffer 1 relative to the start of the request block.
	fill_length_2	Number of relevant bytes in data buffer 2.
	seg_length_2	Actual length of data buffer 2.
	offset_2	Offset of data buffer 2 relative to the start of the request block.

**Substructure
Argument Area**

```
struct application_block
{
  UBYTE                               opcode;
  UBYTE                               subsystem;
  UWORD                               id;
  struct    service                   service;
  struct    remote_address            loc_add;
  UBYTE                               ssap;
  UBYTE                               dsap;
  struct    remote_address            rem_add;
  enum      service_class             serv_class;
  struct    link_service_data_unit    receive_l_sdu;
  UBYTE                               reserved_1;
  UBYTE                               reserved;
  struct    link_service_data_unit    send_l_sdu;
  enum      link_status               l_status;
  UWORD                               reserved_2 [2];
};

struct service
{
  enum      service_code              code;
};
struct remote_address
{
  UBYTE                               station;
  UBYTE                               segment;
};
struct link_service_data_unit
{
  void      far *                     buffer_ptr;
  UBYTE                               length;
};
```

**Description of the
Parameters**

opcode	Request, confirm, indication
subsystem	Reserved for the CP.
id	Reserved for the CP.
service.code	sda, sdn, sdn_broadcast , srd, reply_update_single, reply_update_multiple
loc_add.station	Local address 0 to 126; for SDN: 127 = MULTICAST/ BROADCAST
loc_add.segment	Reserved
ssap	Source service access point, 0 to 62
dsap	Destination service access point, 0 to 63
rem_add.station	Remote address, 0 to 126; for SDN : 127 = MULTICAST/BROADCAST
rem_add.segment	Reserved
serv_class	Priority of the service (low or high)
receive_l_sdu.buffer_ptr	Reserved for the CP.
receive_l_sdu.length	Buffer length, 32 to 255 (for request); user data length for confirm, indication
send_l_sdu.buffer_ptr	Reserved for the CP.
send_l_sdu.length	User data length of the send frame.
l_status	Return parameter, link_status

**Constants for the
Application Block**

The constants used in this chapter that are available to the FDL application are as follows:

DEFAULT_SAP	FFH	Default SAP ID
NO_SEGMENT	FFH	Segment invalid
BROADCAST	127	Global address
MULTICAST	127	Global address
LEN_MAX_RECEIVE_BUFFER	255	Max. receive buffer
LEN_MIN_RECEIVE_BUFFER	32	Min. receive buffer
LEN_DATA_OVERHEAD	14	Length of maximum frame header + trailer

Declaration

The following tables indicate which parameters are mandatory (m), optional (o), don't care (x), returned (r) for the request blocks of the productive services:

Request

request	sda	sdn	srd	reply_update	
length	m	m	m	m	
user	x	x	x	x	
rb_type	m	m	m	m	
priority	m	m	m	m	Request
subsystem	m	m	m	m	block
opcode	m	m	m	m	header
response	x	x	x	x	
fill_length_1	m	m	m	m	
seg_length_1	m	m	m	m	
offset_1	m	m	m	m	
fill_length_2	m	m	m	m	
seg_length_2	m	m	m	m	
offset_2	x	x	m	x	
opcode	m	m	m	m	
subsystem	x	x	x	x	
id	x	x	x	x	Application
service.code	m	m	m	m	block
loc_add.station	x	o	x	x	
loc_add.segment	x	x	x	x	
ssap	m	m	m	m	
dsap	m	m	m	m	
rem_add.station	m	m	m	m	
rem_add.segment	x	x	x	x	
serv_class	m	m	m	m	
receive_l_sdu.length	x	x	m	x	
send_l_sdu.length	m	m	m	m	
l_status	x	x	x	x	
user_data_1	m	m	m	m	User data 1
user_data_2	x	x	m	x	User data 2

Confirmation

confirm	sda	sdn	srd	reply_update	
length	r	r	r	r	
user	x	x	x	x	
rb_type	r	r	r	r	
priority	r	r	r	r	Request
subsystem	r	r	r	r	block
opcode	r	r	r	r	header
response	r	r	r	r	
fill_length_1	r	r	r	r	
seg_length_1	x	x	x	x	
offset_1	r	r	r	r	
fill_length_2	x	x	r	x	
seg_length_2	x	x	r	x	
offset_2	x	x	r	r	
opcode	r	r	r	r	
subsystem	x	x	x	x	
id	x	x	x	x	Application
service.code	r	r	r	r	block
loc_add.station	x	o	x	x	
loc_add.segment	x	x	x	x	
ssap	r	r	r	r	
dsap	r	r	r	r	
rem_add.station	r	r	r	r	
rem_add.segment	x	x	x	x	
serv_class	r	r	r	x	
receive_l_sdu.length	x	x	r	x	
send_l_sdu.length	r	r	r	r	
l_status	r	r	r	r	
user_data_1	x	x	x	x	User data 1
user_data_2	x	x	r	x	User data 2

Indication

indication	sda	sdn	srd	sdn_broadcast	
length	r	r	r	r	
user	x	x	x	x	
rb_type	r	r	r	r	
priority	r	r	r	r	Request
subsystem	r	r	r	r	block
opcode	r	r	r	r	header
response	r	r	r	r	
fill_length_1	r	r	r	r	
seg_length_1	x	x	x	x	
offset_1	r	r	r	r	
fill_length_2	x	x	x	x	
seg_length_2	x	x	x	x	
offset_2	x	x	x	x	
opcode	r	r	r	r	
subsystem	x	x	x	x	
id	x	x	x	x	Application
service.code	r	r	r	r	block
loc_add.station	x	o	x	x	
loc_add.segment	x	x	x	x	
ssap	r	r	r	r	
dsap	r	r	r	r	
rem_add.station	r	r	r	r	
rem_add.segment	x	x	x	x	
serv_class	r	r	r	r	
receive_l_sdu.length	r	r	r	r	
send_l_sdu.length	x	x	x	x	
l_status	x	x	r	x	
user_data_1	r	r	r	r	User data 1
user_data_2	x	x	x	x	User data 2

3.2   Request Blocks of the Productive Services

3.2.1 SDA (send data with acknowledge)

Request   The local station sends data to a remote station and receives a confirmation of the correct or incorrect data transfer.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the send frame low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	Length of the data 13 to 258
seg_length_1	Length of the buffer used 15..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	sda
loc_add.station	No significance
loc_add.segment	No significance
ssap	0 to 62 or DEFAULT_SAP
dsap	0 to 63 or DEFAULT_SAP
rem_add.station	0 to 126
rem_add.segment	No significance
serv_class	Priority of the send frame low/high
receive_l_sdu.length	No significance
send_l_sdu.length	Number of net bytes to be transferred 1 to 246
l_status	No significance

The following diagram shows the structure of the data of the SDA frame. This data is contained in the user_data_1 structure element of the request block.

The total length of the structure element is fixed at 260 bytes in the header file "fdl_rb.h".

The offset byte and the user data must be entered in the data buffer by the FDL application.

Recommendation   Select 12 for the offset byte.

Select 260 as the value for seg_length_1.

Select 12 + the length of the user data as the value for fill_length_1.

Structure of the
Send Buffer

```
┌────────┬──────────────┬──────────────────────────┬────────┐
│ Offset │   Reserved   │        User data         │        │
└────────┴──────────────┴──────────────────────────┴────────┘
```

$12 \leq \text{Offset} \leq 120$

fill_length_1

≤ 2

seg_length_1
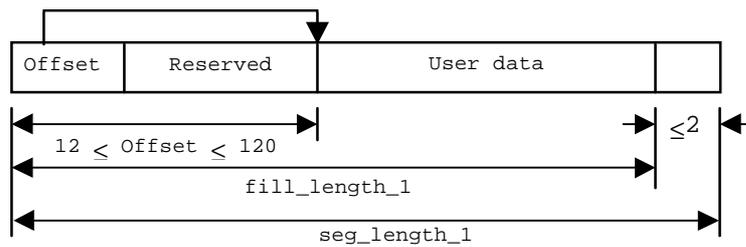
Address Extension The use of address extensions reduces the maximum number of net bytes that can be transmitted by up to 2 bytes.

Address extensions occur when an SAP other than the default SAP is used for dsap and/or ssap.

Confirm The SDA confirmation confirms execution of the SDA request.

The result of the service is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, rr, ue, rs, ls, na, ds, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	Unchanged from request
service.code	sda
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	May be different from request
send_l_sdu.length	Unchanged from request
l_status	ok, rr, ue, rs, ls, na, ds, iv

l_status Values	ok	=	Positive acknowledgment, service executed.
	rr	=	Negative acknowledgment, resources of the CP (remote) not available.
	ue	=	Negative acknowledgment, FDL application/FDL interface error (remote)
	rs	=	Service or rem_add not activated on SAP (remote)
	ls	=	Service not activated on SAP (local)
	na	=	No or no plausible reaction from station (remote)
	ds	=	CP (local) not in logical token ring or disconnected from the bus.
	iv	=	Invalid parameters in the request.

Indication

The SDA indication shows that an SDA request has been received from a remote station.

The receive data are entered in the receive buffer.

Request Block Header

length	80
user	Unchanged from "await_indication"
rb_type	2
priority	Priority of the receive frame low/high
subsystem	22H
opcode	Indication
response	No significance
fill_length_1	Length of the data ($\leq$ 258)
seg_length_1	No significance
offset_1	80
fill_length_2	No significance
seg_length_2	No significance
offset_2	No significance

Application Block

opcode	Indication
subsystem	No significance
id	No significance
service.code	sda
loc_add.station	No significance
loc_add.segment	No significance
ssap	SAP of the station (local) 0 to 63 or DEFAULT_SAP
dsap	SAP of the station (remote) 0 to 62 or DEFAULT_SAP
rem_add.station	address of the source station 0 to 126
rem_add.segment	No significance
serv_class	Priority of the receive frame low/high
receive_l_sdu.length	Length of the received user data 1 to 246
send_l_sdu.length	No significance
l_status	No significance

The following diagram shows the structure of the data received with the SDA indication.

This data is contained in the user_data_1 structure element of the request block.

The offset and the user data are entered in the receive buffer by the CP.

**Structure of the
Receive Buffers**

```
        ┌────────┬──────────┬──────────────────┬──────┐
        │ Offset │ Reserved │    User data     │      │
        └────────┴──────────┴──────────────────┴──────┘
                            │◄─────────────────►│
                              receive_l_sdu.length

        │◄──────────────────────────────────────►│
                        fill_length_1
```
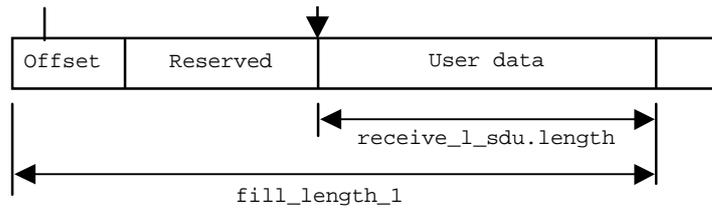
The offset (first byte in the receive buffer) indicates the number of
bytes from the start of the receive buffer to the first byte of the user
data.

3.2.2 SDN (send data with no acknowledge)

Request

The station (local) sends data to a station, a group of stations (MULTICAST) or all stations (BROADCAST). The FDL application only receives a local confirmation, but no confirmation of reception from the remote station(s).

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the send frame low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	Length of the data 13 to 258
seg_length_1	Length of the buffer used 15..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	sdn
loc_add.station	No significance
loc_add.segment	NO_SEGMENT
ssap	0 to 62 or DEFAULT_SAP
dsap	0 to 63 or DEFAULT_SAP
rem_add.station	0 to 126 or MULTICAST (= 127) or BROADCAST (= 127)
rem_add.segment	NO_SEGMENT
serv_class	Priority of the send frame low/high
receive_l_sdu.length	No significance
send_l_sdu.length	Number of net bytes to be transferred 1 to 246
l_status	No significance

Meaning of the Parameters

rem_add.station = BROADCAST: dsap = 63
rem_add.station = MULTICAST: dsap = 0 to 62 or DEFAULT_SAP

The MULTICAST groups are determined by the dsap service access points.

The following diagram shows the structure of the data of the SDN frame. This data is contained in the user_data_1 structure element of the request block.

The total length of the structure element is fixed at 260 bytes in the header file "fdl_rb.h".

The offset byte and the user data must be entered in the data buffer by the FDL application.

Recommendation

Select 12 for the offset byte.

Select 260 as the value for seg_length_1.

Select 12 + the length of the user data as the value for fill_length_1.

**Structure of the
Send Buffer**

| Offset | Reserved | User data | |

$$12 \leq \text{Offset} \leq 120$$

fill_length_1

seg_length_1

≤ 2
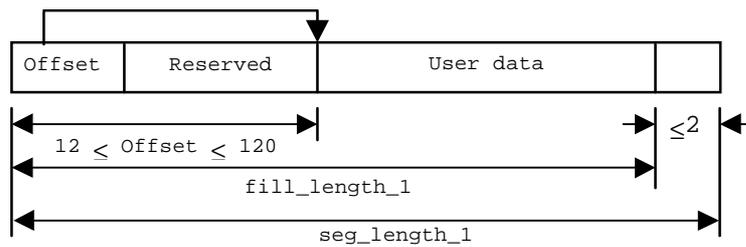
Note

The use of address extensions reduces the maximum number of net bytes that can be transmitted by up to 2 bytes.

Address extensions occur when an SAP other than the default SAP is used for dsap and/or ssap.

Confirmation

The SDN confirmation confirms execution of the SDN request.

The result of the service is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, ls, ds, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	sdn
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	May be different from request
send_l_sdu.length	Unchanged from request
l_status	ok, ls, ds, iv

l_status Values

ok = Transfer of the data from the CP completed
ls = Service not activated on the SAP (local)
ds = CP not in the logical token ring or disconnected from the bus
iv = Invalid parameters in the request

Indication   The SDN indication shows that an SDN request has been received from a remote station.

The received data entered in the receive buffer.

Request Block Header

length	80
user	Unchanged from "await_indication"
rb_type	2
priority	Priority of the receive frame low/high
subsystem	22H
opcode	Indication
response	No significance
fill_length_1	Length of the data ($\leq$ 258)
seg_length_1	No significance
offset_1	80
fill_length_2	No significance
seg_length_2	No significance
offset_2	No significance

Application Block

opcode	Indication
subsystem	No significance
id	No significance
service.code	sdn
loc_add.station	Address of the destination station 0 to 126 (local PROFIBUS address)
loc_add.segment	NO_SEGMENT
ssap	SAP of the station (local) 0 to 62 or DEFAULT_SAP
dsap	SAP of the station (remote) 0 to 62 or DEFAULT_SAP
rem_add.station	Address of the source station 0 to 126
rem_add.segment	NO_SEGMENT
serv_class	Priority of the receive frame low/high
receive_l_sdu.length	Length of the received user data 1 to 246
send_l_sdu.length	No significance
l_status	No significance

The following diagram shows the structure of the data received with the SDN indication.

This data is contained in the user_data_1 structure element of the request block.

The offset and the user data are entered in the receive buffer by the CP.

Structure of the Receive Buffer

The offset (first byte in the receive buffer) indicates the number of bytes from the start of the receive buffer to the first byte of the user data.

Indication (Broadcast, Multicast)

The SDN_BROADCAST indication shows that an SDN request has been received from a remote station that was sent to more than one or to all stations.

The received data are entered in the receive buffer.

Request Block Header

length	80
user	Unchanged from "await_indication"
rb_type	2
priority	Priority of the receive frame low/high
subsystem	22H
opcode	Indication
response	No significance
fill_length_1	Length of the data (≤ 258)
seg_length_1	No significance
offset_1	0
fill_length_2	No significance
seg_length_2	No significance
offset_2	No significance

Application Block

opcode	Indication
subsystem	No significance
id	No significance
service.code	sdn_broadcast
loc_add.station	Address of the destination station BROADCAST
loc_add.segment	NO_SEGMENT
ssap	SAP of the station (local) 0 to 62: MULTICAST or 63: BROADCAST
dsap	SAP of the station (remote) 0 to 62, DEFAULT_SAP
rem_add.station	Address of the source station 0 to 126
rem_add.segment	NO_SEGMENT
serv_class	Priority of the receive frame low/high
receive_l_sdu.length	Length of the received user data 1 to 246
send_l_sdu.length	No significance
l_status	No significance

The following diagram shows the structure of the data received with the SDN indication.

This data is contained in the user_data_1 structure element of the request block.

The offset and the user data are entered in the receive buffer by the CP.

Structure of the Receive Buffer

The offset (first byte in the receive buffer) indicates the number of bytes from the start of the receive buffer to the first byte of the user data.

3.2.3 SRD (send and request data)

Request   The station (local) sends data to a remote station and at the same time
requests data back from this station. As a confirmation of the reception
of the data by the remote station, the local station receives the
response data.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the send frame low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	Length of the data 12 to 258
seg_length_1	Length of the send buffer used 14..260
offset_1	80
fill_length_2	0
seg_length_2	Length of the receive buffer 260
offset_2	Offset from the start of the request block to the data buffer user_data_2

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	srd
loc_add.station	No significance
loc_add.segment	No significance
ssap	0 to 62 or DEFAULT_SAP
dsap	0 to 62 or DEFAULT_SAP
rem_add.station	0 to 126
rem_add.segment	No significance
serv_class	Priority of the send frame low/high
receive_l_sdu.length	Receive buffer length $\geq$ max. (LEN_MIN_RECEIVE_BUFFER, expected frame length) Recommendation: 255
send_l_sdu.length	Number of net bytes to be transferred 0 to 246
l_status	No significance

The following diagram shows the structure of the data of the SRD
frame. This data is contained in the user_data_1 structure element of
the request block.

The total length of the structure element is fixed at 260 bytes in the
header file "fdl_rb.h".

The offset and the user data must be entered in the data buffer by the
FDL application.

Recommendation   Select 12 for the offset byte in the send buffer.

Select 260 as the value for seg_length_1.

Select 12 + the length of the user data as the value for fill_length_1.

Select 340 as the value for offset_2 (offset_1 + seg_length_1). This makes sure that the received data are entered in the user_data_2 structure element.

**Structure of the
Send Buffer**

Offset	Reserved	User data	

$12 \leq$ Offset ≤ 120

≤ 2

fill_length_1

seg_length_1
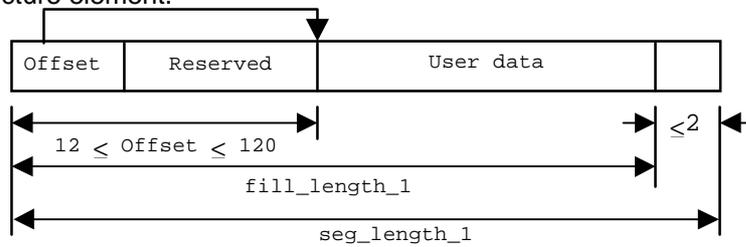
**Structure of the
Receive Buffer**   See SRD confirmation.

The use of address extensions reduces the maximum number of net bytes that can be transferred by up to 2 bytes.

Confirmation   The SRD confirmation confirms execution of the SRD request.

The result of the service is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ue, rr, rs, dl, nr, dh, rdl, rdh, ls, na, ds, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Number of received data ($\leq$ 258)
seg_length_2	Unchanged from request
offset_2	Offset from the start of the request block to data buffer user_data_2

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	srd
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Number of received net bytes 0 to 246, if l_status appropriate
send_l_sdu.length	Unchanged from request
l_status	ue, rr, rs, dl, nr, dh, rdl, rdh, ls, na, ds, iv

The following diagram shows the structure of the data received with the SRD confirmation.

This data is in the user_data_2 structure element of the request block.

The offset and the user data are entered in the receive buffer by the CP.

**Structure of the
Receive Buffer**
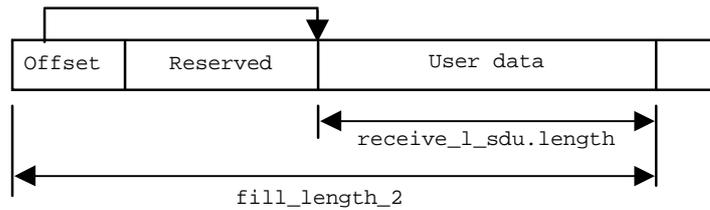
The offset (first byte in the receive buffer) indicates the number of bytes from the start of the receive buffer to the first byte of the user data.

☞ **Note that here fill_length_2 is used since fill_length_1 is already being used for sending the data.**

l_status Values

ue	=	Negative acknowledgment, FDL application/FDL interface error (remote).
rs	=	Service or rem_add not activated on SAP (remote) .
ls	=	Service not activated on SAP (local).
na	=	No or no plausible reaction from station (remote).
ds	=	CP not in logical token ring or disconnected from the bus.
iv	=	Invalid parameters in the request.
dl	=	Response data low exist. Positive acknowledgment for transmitted data.
dh	=	Response data high exist. Positive acknowledgment for transmitted data.
nr	=	Negative acknowledgment. Response data not available on CP (remote). Positive acknowledgment for transmitted data.
rdl	=	Response data low exist. Negative acknowledgment for transmitted data since CP resources (remote) are not available.
rdh	=	Response data high exist. Negative acknowledgment for transmitted data since CP resources (remote) are not available.
rr	=	Negative acknowledgment. CP resources (remote) and response data (remote) are not available.

Indication   The SRD indication confirms the reception of an SRD request from a
remote station.

The received data are entered in the receive buffer.

The update status of the service is entered in the l_status structure
element.

Request Block Header

length	80
user	Unchanged from "await_indication"
rb_type	2
priority	Priority of the receive frame low/high
subsystem	22H
opcode	Indication
response	Update_status
fill_length_1	Length of the data
seg_length_1	No significance
offset_1	80
fill_length_2	No significance
seg_length_2	No significance
offset_2	No significance

Application Block

opcode	Indication
subsystem	No significance
id	No significance
service.code	srd
loc_add.station	No significance
loc_add.segment	No significance
ssap	SAP of the station (local) 0 to 62 of the DEFAULT_SAP
dsap	SAP of the station (remote) 0 to 62 of the DEFAULT_SAP
rem_add.station	Address of the source station 0 to 126
rem_add.segment	No significance
serv_class	Priority of the receive frame low/high
receive_l_sdu.length	Number of received net bytes 0 to 246, if l_status appropriate
send_l_sdu.length	No significance
l_status	update_status

The following diagram shows the structure of the data received with the
SRD indication.

This data is contained in the user_data_1 structure element of the
request block.

The offset and the user data are entered in the receive buffer by the
CP.

**Structure of the
Receive Buffer**

```
┌──────────────────────┐
│                      ▼
┌────────┬────────────┬──────────────────────┬──────┐
│ Offset │  Reserved  │      User data       │      │
└────────┴────────────┴──────────────────────┴──────┘
         │            │◄──────────────────────►│
         │                 receive_l_sdu.length
         │◄──────────────────────────────────►│
                    fill_length_1
```
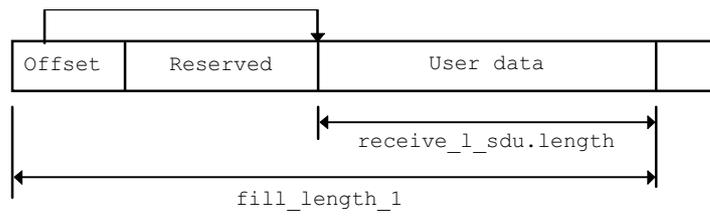
The offset (first byte in the receive buffer) indicates the number of
bytes from the start of the receive buffer to the first byte of the user
data.

**update_status
Values**

lo	=	The response to this SRD was low priority data.
hi	=	The response to this SRD was high priority data.
no_data	=	No data were transmitted in response to this SRD.

3.2.4 REPLY_UPDATE_SINGLE

Request   With this service, the FDL application prepares data for a particular
                           service access point (ssap). This data can be fetched by a different
                           station that has access to this SAP using an SRD. The data is only
                           transferred **once**.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the send frame
subsystem	22H
opcode	Request
response	No significance
fill_length_1	Length of the data 12 to 258
seg_length_1	Length of the buffer used 14..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	Reply_update_single
loc_add.station	No significance
loc_add.segment	No significance
ssap	0 to 62 or DEFAULT_SAP
	Data is made ready for this SAP
dsap	No significance
rem_add.station	No significance
rem_add.segment	No significance
serv_class	Priority of the send frame low/high
receive_l_sdu.length	No significance
send_l_sdu.length	Number of net bytes to be transferred 0 to 246
l_status	No significance

The following diagram shows the structure of the data of the
REPLY_UPDATE_SINGLE. This data is contained in the user_data_1
structure element of the request block.

The total length of the structure element is fixed at 260 bytes in the
header file "fdl_rb.h".

The offset and the user data must be entered in the data buffer by the
FDL application.

Recommendation   Select 12 for the offset byte in the send buffer.

                          Select 260 as the value for seg_length_1.

                          Select 12 + the length of the user data as the value for fill_length_1.

**Structure of the
Send Buffer**
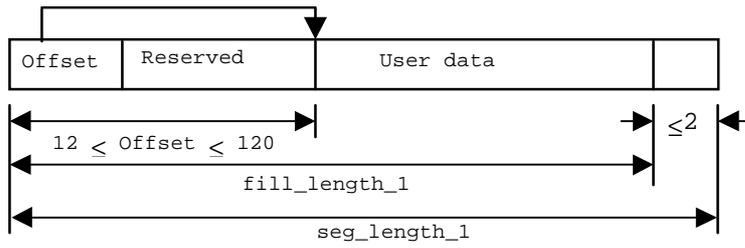
The FDL can only provide a low **or** high priority data buffer per SAP.

Confirmation   The REPLY_UPDATE_SINGLE confirmation confirms execution of the
                                    REPLY_UPDATE_SINGLE request.

                                    The result of the service is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, ls, lr, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	reply_update_single
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Unchanged from request
l_status	ok, ls, lr, iv

l_status Values   ok = data area loaded.
                             ls = service not activated on the SAP (local)
                             lr = response resource currently being used by the CP (temporary
                                    error).
                             iv = invalid parameters in the request.

                             To transfer new data to an SAP, the FDL application can use the
                             REPLY_UPDATE_SINGLE service at any time.

                             Please note that the job is acknowledged negatively if such a buffer
                             has   already   been   transferred   to   this   SAP   with
                             REPLY_UPDATE_SINGLE or REPLY_UPDATE_MULTIPLE **and** this
                             buffer is currently being sent. The REPLY_UPDATE_SINGLE must
                             then be started again.

3.2.5 REPLY_UPDATE_MULTIPLE

Request

With this service, the FDL application prepares data for a certain service access point (ssap). This data can be fetched by any other station with access to this SAP using an SRD.

In contrast to the REPLY_UPDATE_SINGLE request, the data can be transferred **more than once**.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the send frame low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	Length of the data 12 to 258
seg_length_1	Length of the buffer used 14..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	Reply_update_multiple
loc_add.station	No significance
loc_add.segment	No significance
ssap	0 to 62 or DEFAULT_SAP
	Data are prepared for this SAP
dsap	No significance
rem_add.station	No significance
rem_add.segment	No significance
serv_class	Priority of the send frame low/high
receive_l_sdu.length	No significance
send_l_sdu.length	Number of net bytes to be transferred 0 to 246
l_status	No significance

The following diagram shows the structure of the data of the REPLY_UPDATE_MULTIPLE. This data is contained in the user_data_1 structure element of the request block.

The total length of the structure element is fixed at 260 bytes in the header file "fdl_rb.h".

The offset and the user data must be entered in the data buffer by the FDL application.

Recommendation   Select 12 for the offset byte in the send buffer.

Select 260 as the value for seg_length_1.

Select 12 + the length of the user data as the value for fill_length_1

**Structure of the
Send Buffer**
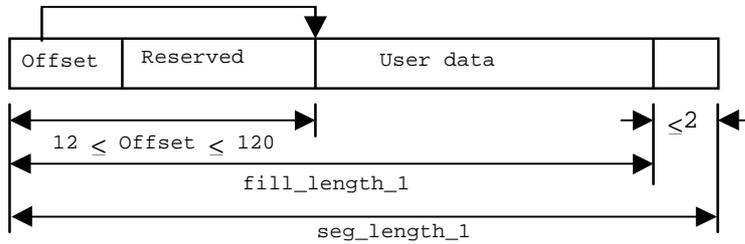
The FDL can only prepare either a low **or** high priority data buffer per SAP.

Confirmation   The REPLY_UPDATE_MULTIPLE confirmation confirms execution of the REPLY_UPDATE_MULTIPLE request.

The result of the service is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, ls, lr, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	Reply_update_multiple
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Unchanged from request
l_status	ok, ls, lr, iv

I_status Values   ok = data area loaded.
 ls = service not activated on the SAP (local).
 lr = response source currently being used by the CP (temporary error).
 iv = invalid parameters in the request.

To transfer new data to an SAP, the FDL application can use the REPLY_UPDATE_MULTIPLE service at any time.

Please note that the job is acknowledged negatively if such a buffer has already been transferred to this SAP with REPLY_UPDATE_SINGLE or REPLY_UPDATE_MULTIPLE **and** this buffer is currently being sent. The REPLY_UPDATE_MULTIPLE must then be started again.

Notes

4   Management Services

This chapter explains the basic principles of the management services.

The chapter explains the following:

➢   The data structures of the management services

➢   The request blocks of the management services

The following management services are described in detail:

➢   SAP_ACTIVATE

➢   RSAP_ACTIVATE

➢   SAP_DEACTIVATE

➢   AWAIT_INDICATION

➢   WITHDRAW_INDICATION

➢   LSAP_STATUS

➢   FDL_LIFE_LIST_CREATE_LOCAL

➢   FDL_LIFE_LIST_CREATE_REMOTE

➢   FDL_READ_STATISTIC_COUNTER

➢   FDL_EVENT

➢   FDL_READ_VALUE

4.1   Data Structures of the Management Services

Request Block Structure

The same request block structure is used for the management services as for the productive services. Owing to the different functions, there are new service codes and the contents of the send and receive buffers are different. Up to now, the FDL application has structured the user data of the frames when it saved them, the data required by the management service are, however, now saved in the appropriate structure.

```
typedef struct
{
  rb2_header_type rb2_header;
  struct application_block    application_block;
  UBYTE     reserved  [12];
  UBYTE     reference [2];
  UBYTE     user_data_1 [260];
  UBYTE     user_data_2 [260];
} fdl_rb;
```

Description of the Parameters

rb2_header	Request block header. General, non service-dependent parameters
application_block	Argument area. FDL parameters.
reference	ID of the FDL application.
user_data_1	User data depending on the particular job.
user_data_2	User data depending on the particular job.

Substructure of the Request Block Header

```
typedef  struct
{
  UWORD      reserved [2];
  UBYTE      length;
  UWORD      user;
  UBYTE      rb_type;
  UBYTE      priority;
  UBYTE      reserved_1;
  UWORD      reserved_2;
  UBYTE      subsystem;
  UBYTE      opcode;
  UWORD      response;
  UWORD      fill_length_1;
  UBYTE      reserved_3;
  UWORD      seg_length_1;
  UWORD      offset_1;
  UWORD      reserved_4;
  UWORD      fill_length_2;
  UBYTE      reserved_5;
  UWORD      seg_length_2;
  UWORD      offset_2;
  UWORD      reserved_6;
} rb2_header_type;
```

Description of the Parameters

length	Length of the request block without "user_data_1" and "user_data_2" (= 80 bytes).
user	User ID, available for the FDL application.
rb_type	Type of request block used (= 2).
priority	Priority of the job.
subsystem	Communication layer selection (FDL = 22h).
opcode	Request, confirm, indication (same as the parameter "opcode" in the application block).
response	Return parameter (same as the parameter "l_status" in the application block).
fill_length_1	Number of relevant bytes in data buffer 1.
seg_length_1	Actual length of data buffer 1.
offset_1	Offset of data buffer 1 relative to the start of the request block.
fill_length_2	Number of relevant bytes in data buffer 2.
seg_length_2	Actual length of data buffer 2.
offset_2	Offset of data buffer 2 relative to the start of the request block.

Substructure Argument Area

```
struct application_block
{
  UBYTE                                 opcode;
  UBYTE                                 subsystem;
  UWORD                                 id;
  struct    service                     service;
  struct    remote_address              loc_add;
  UBYTE                                 ssap;
  UBYTE                                 dsap;
  struct    remote_address              rem_add;
  enum      service_class               serv_class;
  struct    link_service_data_unit receive_l_sdu;
  UBYTE                                 reserved_1;
  UBYTE                                 reserved;
  struct    link_service_data_unit send_l_sdu;
  enum      link_status                 l_status;
  UWORD                                 reserved_2 [2];
};
struct service
{
  enum      service_code                code;
};
struct remote_address
{
  UBYTE                                 station;
  UBYTE                                 segment;
};
struct link_service_data_unit
{
  void      far *                       buffer_ptr;
  UBYTE                                 length;
};
```

Description of the Parameters

opcode	Request, confirm, indication
subsystem	Reserved for the CP.
id	Reserved for the CP.
service.code	fdl_read_value, sap_activate, rsap_activate, sap_deactivate, fdl_life_list_create_local, fdl_ident, fdl_event, await_indication, withdraw_indication, lsap_status, fdl_life_list_create_remote
loc_add.station	Irrelevant for management services
loc_add.segment	Irrelevant for management services
ssap	Source service access point, 0 to 62
dsap	Destination service access point for LSAP_STATUS; Number of the SAP for (R)SAP_ACTIVATE, SAP_DEACTIVATE (0 to 63)
rem_add.station	Remote address, 0 to 126, for FDL_IDENT
rem_add.segment	Reserved
serv_class	Priority of the service (low or high)
receive_l_sdu.length	Service dependent
send_l_sdu.length	No significance
l_status	Return parameter, link_status

Send Buffers   The send buffers have the following significance for the various jobs:

Service	Structure used
sap_activate	fdl_sap
rsap_activate	fdl_sap
Others	No significance

Return Values   The FDL application receives the following completed structures:

Service	Structure used
fdl_read_value	Bus_parameter_block
fdl_event	Event_indication
lsap_status	Byte buffer
fdl_life_list_create_local	Byte buffer
fdl_life_list_create_remote	Byte buffer
fdl_ident	Byte buffer
fdl_read_statistic_ctr	Statistic_ctr_list
await_indication	Byte buffer
others	No significance

Substructure Bus Parameter Block

```
struct        bus_parameter_block
{
  UBYTE                            hsa;
  UBYTE                            ts;
  enum     station_type       station_type;
  enum     baud_rate          baud_rate;
  enum     redundancy         medium_red;
  UWORD                            retry_ctr;
  UBYTE                            default_sap;
  UBYTE                            network_connection_sap;
  UWORD                            tsl;
  UWORD                            tqui;
  UWORD                            tset;
  UWORD                            min_tsdr;
  UWORD                            max_tsdr;
  d_word                           ttr;
  UBYTE                            g;
  boolean                          in_ring_desired;
  enum     physical_layer     physical_layer;
  struct   ident              ident;
};

struct ident
{
  UBYTE                  reserved_header[8];
  UBYTE                  ident[202];
  UBYTE                  response_frame_length;
};
```

Meaning of the Parameters

hsa	Highest PROFIBUS address of an active station on the bus
ts	PROFIBUS address of the station (local)
station_type	Type of the station (local) (active, passive);
baud_rate	Kbps_9_6 , Kbps_19_2, Kbps_93_75, Kbps_187_5, Kbps_500, Mbps_1_5, Mbps_3, Mbps_6, Mbps_12
medium_red	Redundancy
retry_ctr	Number of repeated calls to a responder that is not answering, 0 to 7
network_connection_sap	No significance
default_sap	Number of the default SAP (local)
tsl	SLOT time
tqui	Transmitter fall time/repeater changeover time
tset	Setup time
min_tsdr	Minimum station delay time
max_tsdr	Maximum station delay time
ttr	Target rotation time
g	GAP update factor
in_ring_desired	Request to enter ring
physical_layer	Selectable physical bus characteristics
ident	Vendor name, controller type, hardware and software versions

Structure fdl_sap

```
struct      fdl_sap
{
  UWORD     user_id;
  UBYTE     max_l_sdu_length;
  UBYTE     access_sap;
  UBYTE     access_station;
  UBYTE     access_segment;
  UBYTE     max_l_sdu_length;
  UBYTE     sda;
  UBYTE     sdn;
  UBYTE     srd;
  UBYTE     csrd;
  void      far   *rup_l_sdu_ptr_low;
  void      far   *rup_l_sdu_ptr_high;
};
```

Meaning of the   see Section 4.2.2
Parameters

**Structure
event_indication**

```
struct       event_indication
{
  struct     event_ctr    time_out;
  struct     event_ctr    not_syn;
  struct     event_ctr    uart-error;
  struct     event_ctr    out_of_ring;
  struct     event_ctr    sdn_not_indicated;
  struct     event_ctr    duplicate_address;
  struct     event_ctr    hardware_error;
  struct     event_ctr    mac_error;
};
```

**Meaning of the
Parameters**   see Section 4.2.11

Structure event_ctr

```
struct       event_ctr
{
  UWORD      threshold
  UWORD      counter
};
```

**Meaning of the
Parameters**   see Section 4.2.11

**Structure
statistic_ctr_list**

```
struct       statistic_ctr_list
{
  UWORD      invalid_start_delimiter_ctr;
  UWORD      invalid_fcb_fcv_ctr;
  UWORD      invalid_token_ctr;
  UWORD      collision_ctr;
  UWORD      wrong_fcs_or_ed_ctr;
  UWORD      frame_error_ctr;
  UWORD      char_error_ctr;
  UWORD      retry_ctr;
  d_word     start_delimiter_ctr;
  d_word     stop_receive_ctr;
  d_word     send_confirmed_ctr;
  d_word     send_sdn_ctr;
};
```

**Meaning of the
Parameters**   see Section **Fehler! Verweisquelle konnte nicht gefunden werden.**

Constants   Constants used in this chapter and available to the FDL application are as follows:

Values for the Application Block:

DEFAULT_SAP	FFH	Default SAP identifier
NO_SEGMENT	FFH	Segment invalid
BROADCAST	127	Global address
MULTICAST	127	Global address
EVENT_SAP	64	SAP number for events
LEN_MAX_RECEIVE_BUFFER	255	Max. receive buffer
LEN_MIN_RECEIVE_BUFFER	32	Min. receive buffer
LEN_DATA_OVERHEAD	14	Length of the maximum frame header plus trailer

Structure Values for Management Services:

LEN_BUS_PARAMETER	Length of structure "bus_parameter_block"
LEN_SAP_ACTIVATE	Length of structure "fdl_sap"
LEN_POLL_ELEMENT	Length of structure "user_poll_element"
LEN_APPLICATION_BLOCK	Length of structure "application_block"
LEN_IDENT	Length of structure "ident"
LEN_EVENT_INDICATION	Length of structure "event_indication"
LEN_STATISTIC_CTR_LIST	Length of structure "statistic_ctr_list"

Constants for SAP Configurations:

ALL	7FH
SEGMENT_VALID	80H
SEGMENT_INVALID	00H
SEGMENT_TYP	40H
INITIATOR	00H
RESPONDER	10H
BOTH_ROLES	20H
SERVICE_NOT_ACTIVATED	30H

Constants for Life List:

STATION_PASSIVE	00H
STATION_NOT_EXISTENT	10H
STATION_ACTIVE_READY	20H
STATION_ACTIVE	30H

4.2   Request Blocks of the Management Services

4.2.1 FDL_READ_VALUE

Request   The current bus parameters of the CP can be read with this service.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	Length of the buffer used ($\geq$ LEN_BUS_PARAMETER)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	fdl_read_value
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	No significance
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

Confirmation   The FDL_READ_VALUE confirmation confirms execution of the FDL_READ_VALUE request.

The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, iv
fill_length_1	LEN_BUS_PARAMETER
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	fdl_read_value
loc_add.station	No significance
loc_add.segment	No significance
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Unchanged from request
l_status	ok, iv

l_status Values   ok = Positive acknowledgment, the bus parameters were read.

                      iv = Negative acknowledgment:
                            - CP is currently being reset
                            - No receive buffer

Meaning of the Parameters

The parameters described in the following structure are entered in the user_data_1 structure element by the CP.

struct bus_parameter_block

hsa	Highest PROFIBUS address on the bus, 2 to 126.
ts	PROFIBUS address local station, 0 to hsa or 126.
station_type	Type of station (local)
baud_rate	Data rate: Kbps_9_6, Kbps_19_2, Kbps_93_75, Kbps_187_5, Kbps_500, Mbps_1_5, Mbps_3, Mbps_6, Mbps_12.
medium_red	Redundancy
retry_ctr	Number of repeated calls to a non-responding station (remote), 0 to 7.
default_sap	Number of the default SAP of the station (local), 0 to 63.
network_connection_sap	Reserved
tsl	SLOT time
tqui	Transmitter fall time/repeater changeover time
tset	Setup time
min_tsdr	Minimum station delay time.
max_tsdr	Maximum station delay time.
ttr	Target rotation time
g	GAP update factor
in_ring_desired	Request to enter the ring
physical_layer	Selectable physical bus characteristics
ident	Vendor name, controller type, hardware and software versions

4.2.2 SAP_ACTIVATE

Request   With this service you can assign parameters to service access points
                               (SAPs) in the FDL and activate them.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	LEN_SAP_ACTIVATE
seg_length_1	Length of the buffer used ($\geq$ LEN_SAP_ACTIVATE)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	Sap_activate
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	Number of the SAP to be activated, 0 to 63
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

Note   The number of the SAP to be activated locally must be specified in the
                               application block in the DSAP parameter (instead of SSAP).

Structure Element
user_data_1

The parameters described in the following structure are entered in the user_data_1 structure element by the FDL application.

struct fdl_sap

user_id	Identification for the FDL application; no significance for the CP.
max_l_sdu_length	Maximum user data length processed on this SAP. Recommendation: 246
access_sap	Optional access right for a particular SAP (remote) on this SAP. Other remote SAPs ($\neq$ access_sap) are not allowed access (0 to 63, ALL). ALL = no access restrictions.
access_station	Optional access right for a particular station (remote) on this SAP. Stations with the PROFIBUS address $\neq$ access_station are not allowed access (0 to hsa, ALL). ALL = no access restrictions.
access_segment	Reserved
sda	Specifies the role
sdn	Specifies the role
srd	Specifies the role
csrd	Reserved
*rup_l_sdu_ptr_low	No significance
*rup_l_sdu_ptr_high	No significance

Role:

INITIATOR	Station (local) can only be initiator of the service.
RESPONDER	Station (local) can only be responder in the service.
BOTH_ROLES	Station (local) can be both initiator and responder in the service.
SERVICE_NOT_ACTIVATED	Service is not activated.

Note:

An SAP can be activated for several services. If, however BOTH_ROLES and/or RESPONDER are entered more than once, all entries (SDA, SDN and SRD) become BOTH_ROLES.

CSRD is no longer supported.

Note on
LSAP_STATUS

The service LSAP_STATUS allows the roles set with SAP_ACTIVATE to be read.

Confirmation   The SAP_ACTIVATE confirmation confirms execution of the SAP_ACTIVATE request.

The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, no, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	Sap_activate
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Unchanged from request
l_status	ok, no, iv

l_status Values   ok = Positive acknowledgment, SAP was activated.
                          no = Negative acknowledgment, SAP exists already.
                          iv = Negative acknowledgment:
                                   - CP currently being reset
                                   - SAP parameter invalid
                                   - SAP number invalid

4.2.3 RSAP_ACTIVATE

Request   With this service, service access points (SAPs) with a pure responder
                         role can be assigned parameters and activated for SRD in the FDL.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	LEN_SAP_ACTIVATE
seg_length_1	Length of the buffer used (≥ LEN_SAP_ACTIVATE)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	Rsap_activate
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	Number of the SAP to be activated, 0 to 63
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

Note   The number of the SAP to be activated locally must be specified in the
                         application block in the DSAP parameter (instead of SSAP).

**Structure Element
user_data_1**

The parameters described in the following structure are entered in the
user_data_1 structure element by the FDL application.

struct fdl_sap

user_id	Identification for the FDL application, no significance for the CP.
max_l_sdu_length	Maximum user data length processed on this SAP (recommendation 246).
access_sap	Optional access rights for a particular SAP (remote) on this SAP. Other remote SAPs (≠ access_sap) are not permitted access (0 to 63, ALL). ALL = no access restriction.
access_station	Optional access right for a particular station (remote) on this SAP. Stations with an PROFIBUS address ≠ access_station are not permitted access (0..hsa, ALL). ALL = no access restrictions
access_segment	Reserved
sda	No significance
sdn	No significance
srd	Specifies the role = RESPONDER
csrd	Reserved
*rup_l_sdu_ptr_low	No significance
*rup_l_sdu_ptr_high	No significance

Confirmation The RSAP_ACTIVATE confirmation confirms execution of the RSAP_ACTIVATE request.

The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, no, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	rsap_activate
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Unchanged from request
l_status	ok, no, iv

l_status Values

ok = Positive acknowledgment, SAP was activated.

no = Negative acknowledgment, SAP exists already

iv = Negative acknowledgment:
- CP currently being reset
- SAP parameter invalid
- SAP number invalid

4.2.4 SAP_DEACTIVATE

Request

With this service, service access points (SAPs) can be deactivated.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	Length of the buffer used ≥ LEN_SAP_ACTIVATE
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	sap_deactivate
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	Number of the SAP to be deactivated, 0 to 63
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

Note

The number of the SAP to be activated locally must be specified in the application block in the DSAP parameter (instead of SSAP).

An SAP can only be deactivated when there are no more resources attached to the SAP. Buffers transferred with previous AWAIT_INDICATION requests that still exist in layer 2 must first be fetched back with WITHDRAW_INDICATION before an SAP_DEACTIVATE can be performed.

Confirmation   The SAP_DEACTIVATE confirmation confirms execution of the SAP_DEACTIVATE request.

The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, no, lr, iv
fill_length_1	LEN_SAP_ACTIVATE
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	sap_deactivate
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Unchanged from request
l_status	ok, no, lr, iv

l_status Values   ok = Positive acknowledgment, SAP was deactivated
no = Negative acknowledgment, SAP does not exist
lr = Negative acknowledgment: CP access to SAP (temporary),
       there are still indication resources on the SAP.
iv = Negative acknowledgment:
          - CP currently being reset
          - SAP number invalid

Data, transferred to the CP for this SAP using "REPLY_UPDATE_..." are discarded.

4.2.5 LSAP_STATUS

Request

This service allows configuration parameters for a particular SAP to be read. Only the SAPs of the local station (local) can be read.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	Length of the receive buffer
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	lsap_status
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	Number of the SAP (local) 0 to 63 or DEFAULT_SAP
rem_add.station	ts (local PROFIBUS address)
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	Receive buffer length
	Recommendation: 255
Send_l_sdu.length	No significance
l_status	No significance

Note

The number of the SAP to be activated locally must be specified in the application block in the DSAP parameter (instead of SSAP).

Confirmation   The LSAP_STATUS confirmation confirms execution of the LSAP_STATUS request.

The result of the request is entered in the l_status structure element.

Request Block Header

|	
length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, iv, rs
fill_length_1	Length of the returned data + offset (see "Structure of the receive buffer")
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	lsap_status
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Number of returned net bytes 0 to 6, with appropriate l_status
send_l_sdu.length	Unchanged from request
l_status	ok, iv, rs

The following diagram shows the structure of the data received with the LSAP_STATUS confirmation.

This data is contained in the user_data_1 structure element of the request block.

The offset and the user data are entered in the receive buffer by the CP.

Structure of the Receive Buffer
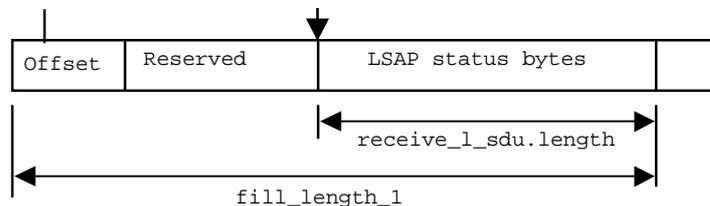
The offset (first byte of the receive buffer) indicates the number of bytes from the start of the receive buffer to the first byte of the user data.

l_status Values   ok = Positive acknowledgment, status was read.
                          rs = No acknowledge of CP 5412(A2)/CP 5613/CP 5614:
                                    - The SAP is not activated
                          iv = Negative acknowledgment:
                                    - CP currently being reset
                                    - Invalid parameters in the application block
                                    - Other management service currently active

Meaning of the   If l_status = ok, 6 status bytes are read. The bytes have the following
Parameters: meaning:

BYTE 1:	Station access restriction (access_station)
BYTE 2:	Reserved
BYTE 3 to 6:	Status of the individual services (SDA, SDN, SRD)

Structure of BYTE 1:
b8:   Bit 8 is always 1.
b7 to b1:   Only the station with the PROFIBUS address b7 to
                      b1 can access this SAP.
                      b7 to b1 = 7FH means there is no access
                      restriction (ALL).

Structure of BYTE 2:
b8 .. b1:   Reserved

Structure of BYTE 3 to 5:
b8 to b5:   Specifies the role in the service:
                      0000   INITIATOR
                      0001   RESPONDER
                      0010   BOTH_ROLES
                      0011   SERVICE_NOT_ACTIVATED

b4 to b1:   Specifies the service ID:
                      0000   SDA_RESERVED
                      0001   SDN_RESERVED
                      0011   SRD_RESERVED
                      0101   Reserved

Structure of BYTE 6:
Reserved

Note   The CSRD is no longer supported.

4.2.6 FDL_LIFE_LIST_CREATE_REMOTE

Request   This service supplies the FDL application with a current list of functional stations on the bus. A status frame is sent to all possible active or passive stations on the bus (bus load).

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	Length of the buffer used 127..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	fdl_life_liste_create_remote
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	No significance
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

In contrast to FDL_LIFE_LIST_CREATE_LOCAL, the function also provides the PROFIBUS addresses of passive stations (slaves) with which the local CP does **not** exchange data.

Confirmation   The FDL_LIFE_LIST_CREATE_REMOTE confirmation shows the execution of the FDL_LIFE_LIST_CREATE_REMOTE request.

The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, ds, lr, iv
fill_length_1	127 if l_status = ok
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	fdl_life_list_create_remote
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	127 if l_status = ok
send_l_sdu.length	Unchanged from request
l_status	ok, ds, lr, iv

Structure of the Life List

user_data_1   Status station x

(byte) e.g. 00h	x = 0 e.g. STATION_PASSIVE
(byte) e.g. 10h	x = 1 e.g. STATION_NON_EXISTENT
(byte) e.g. 30h	x = 2 e.g. STATION_ACTIVE
(byte) e.g. 20h	x = 126 e.g. STATION_ACTIVE_READY

Values for Status

Status: STATION_NON_EXISTENT =   10H
          STATION_ACTIVE_READY =   20H (ready for entry in the logical ring)
          STATION_ACTIVE   =   30H (already in the logical ring)
          STATION_PASSIVE   =   00H

l_status Values

ok = Positive acknowledgment, life list was created.
ds = CP not in logical token ring or disconnected from the bus.
lr = Resources of the CP not available or inadequate.
iv = Negative acknowledgment:
         - CP currently being reset
         - passive station
         - other management service currently active

4.2.7 FDL_LIFE_LIST_CREATE_LOCAL

Request   The service provides the FDL application with a current list of functional stations on the bus. The list is generated from the information on the local station (no bus load).

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	Length of the buffer used 127..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	fdl_life_list_create_local
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	No significance
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

In contrast to FDL_LIFE_LIST_CREATE_REMOTE, the function returns only the PROFIBUS addresses of active and passive nodes (slaves) with which the local CP exchanges data.

If the FDL_LIFE_LIST_CREATE_REMOTE service has already been executed, an image of all nodes is provided, this means that passive nodes that have already been entered are not removed.

Confirmation   The FDL_LIFE_LIST_CREATE_LOCAL confirmation shows the
                          execution of the FDL_LIFE_LIST_CREATE_LOCAL request.

                          The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, lr, iv
fill_length_1	127 if l_status = ok
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	fdl_life_list_create_local
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	127 if l_status = ok
send_l_sdu.length	Unchanged from request
l_status	ok, lr, iv

Structure of the
Life List
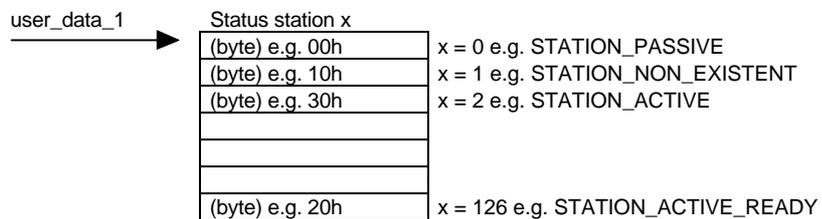
user_data_1   →   Status station x

(byte) e.g. 00h	x = 0 e.g. STATION_PASSIVE
(byte) e.g. 10h	x = 1 e.g. STATION_NON_EXISTENT
(byte) e.g. 30h	x = 2 e.g. STATION_ACTIVE
(byte) e.g. 20h	x = 126 e.g. STATION_ACTIVE_READY

Status: STATION_NON_EXISTENT   =   10H
          STATION_ACTIVE_READY   =   20H (ready for entry in the logical ring)

          STATION_ACTIVE   =   30H (already in the logical ring)
          STATION_PASSIVE   =   00H

l_status Values   ok =   Positive acknowledgment, life list was created.
                         lr =   Resources of the CP not available or inadequate.
                         iv =   Negative acknowledgment:
                                 - CP currently being reset
                                 - no life list buffer exists
                                 - passive station
                                 - other management service currently active

4.2.8 FDL_IDENT

Request   With this service, a station (local) can be identified. The identification
                 includes the vendor name, the module type and the hardware and
                 software versions.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	Length of the receive buffer used ($\geq$ LEN_IDENT $\leq$ 260)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	fdl_ident
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	No significance
rem_add.station	0 to 126; if local PROFIBUS address, then check local ident
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	receive buffer length
	Recommendation: 255
send_l_sdu.length	No significance
l_status	No significance

Confirmation   The FDL_IDENT confirmation confirms execution of the FDL_IDENT
                          request.

                          The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, na, ds, nr, lr, iv
fill_length_1	Length of the Ident (0..200) + offset (if l_status = ok)
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	fdl_ident
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Length of the Ident 0 to 200, if l_status = ok
send_l_sdu.length	Unchanged from request
l_status	ok, na, ds, nr, lr, iv

The following diagram shows the structure of the data received with the
FDL_IDENT confirmation.

This data is contained in the user_data_1 structure element of the
request block.

The offset and die Ident-data are entered in the receive buffer by the
CP.
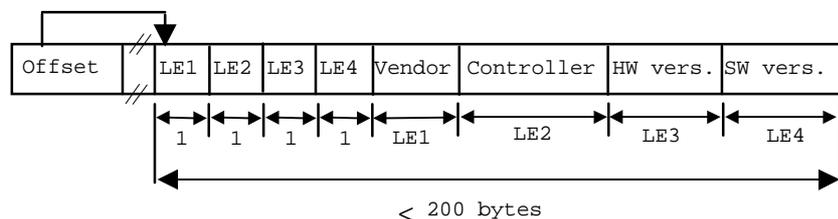
**Structure of the
Receive Buffer**

The offset (first byte of the receive buffer) indicates the number of
bytes from the start of the receive buffer to the first byte of the user
data.
The last four elements contain character strings.

I_status Values   ok   =   Positive acknowledgment, Ident was read.
                          na   =   No or no plausible reaction from the addressed station
                                   (remote).
                          ds   =   CP not in the logical token ring or disconnected from the bus.
                          nr   =   Negative acknowledgment for Ident-data, since these are not
                                   available on the addressed station (remote).
                          lr   =   Resources of the CP not available or inadequate.
                          iv   =   Negative acknowledgment:
                                   - CP currently being reset
                                   - invalid parameters in the application block
                                   - other management service currently active.

I_status Values   ok   =   Positive acknowledgment, Ident was read.

4.2.9 FDL_READ_STATISTIC_COUNTER

Request   This service is used to read the statistical data of the local station. Each time the data is read, the counters are reset.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	Length of the buffer used
	Recommendation: 260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	fdl_read_statistic_ctr
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	No significance
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

Confirmation   The FDL_READ_STATISTIC_COUNTER confirmation shows the execution of the FDL_READ_STATISTIC_COUNTER request.

The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, iv
fill_length_1	LEN_STATISTIC_CTR_LIST
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	fdl_read_statistic_ctr
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	No significance
send_l_sdu.length	Unchanged from request
l_status	ok, iv

Structure Element   The parameters described in the following structure are entered in the
user_data_1   user_data_1 structure element by the CP.

struct statistic_ctr_list:

invalid_start_delimiter_ctr	Receive frame with invalid start delimiter
invalid_fcb_fcv_ctr	Reserved
invalid_token_ctr	Invalid token received
collision_ctr	Unexpected response frame, possibly bus collisions or bus short-circuit.
wrong_fcs_or_ed_ctr	Reserved
frame_error_ctr	Gap in the received frame.
char_error_ctr	Reserved
retry_ctr	Frame repetitions
start_delimiter_ctr	Receive frame with valid start delimiter.
stop_receive_ctr	Reception aborted, because:
	- incorrect start delimiter
	- bus short-circuit or bus collisions
	- station exists twice
	- Invalid entry in the frame
send_confirmed_ctr	Number of sent "confirmed requests" (SDA, SRD).
send_sdn_ctr	Number of sent SDN requests.

l_status Values   ok = Positive acknowledgment, statistics were read.
                       iv = Negative acknowledgment:
                              - CP currently being reset
                              - no statistic buffer exists

4.2.10 AWAIT_INDICATION

Request

With this service, a resource for an indication is made available to the CP. The management of individual resources is SAP-related.

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	If dsap = EVENT_SAP: LEN_EVENT_INDICATION otherwise: 0
seg_length_1	Recommendation: 260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	await_indication
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	0 to 63 or EVENT_SAP; SAP for which resources are made available
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	receive buffer length
	Recommendation: 255
send_l_sdu.buffer_ptr	0
send_l_sdu.length	1
l_status	No significance

☞ **If the DSAP structure element has the value EVENT_SAP, an FDL_EVENT indication is received with the resource. In all other cases, the resource is made available for receiving an SDA, SDN or SRD indication.**

☞ **In contrast to all other SAPs, the EVENT-SAP does not need to be created with the SAP_ACTIVATE service.**

Lengths Dependent on the SAP Used

	SAP 0..63	EVENT_SAP
fill_length_1	Recommendation: 0	Recommendation: LEN_EVENT_INDICATION
seg_length_1	Recommendation: 260	Recommendation: 260
receive_l_sdu.length	Recommendation: 255	No significance

☞ **Please remember that there is only a direct confirmation with the "l_status = ls", "lr" or "iv" for the AWAIT_INDICATION service if the service was unsuccessful. If the request was correct, the request block remains on the CP.**

☞ **If you fetch back the resources using the WITHDRAW_INDICATION service, the opcode structure element is unchanged, in other words the entry continues to be "request".**

| **Meaning of the Parameters** | dsap = EVENT_SAP: | A resource is made available for the FDL_EVENT indication. A resource consists of an application block and a receive buffer (=LEN_EVENT_INDICATION, struct event_indication). |
| | dsap = 0 to 63: | A resource is made available for an SDA, SDN or SRD indication. A resource consists of an application block and a receive buffer. |

Structure of the Receive Buffer with FDL_EVENT

The parameters described in the following structure are entered in the user_data_1 structure element by the FDL application. /1/

struct event_indication	
time_out.counter	Initialize with 0.
time_out.threshold	1 to 65535; threshold, can be set individually for every event. As soon as the "time_out.counter" reaches the "time_out.threshold" an FDL_EVENT indication with the complete receive buffer is triggered.
not_syn.counter	Initialize with 0.
not_syn.threshold	1 to 65535; threshold, can be set individually for every event. As soon as the "not_syn.counter" reaches the "not_syn.threshold" an FDL_EVENT indication with the complete receive buffer is triggered.
uart_error.counter	Initialize with 0.
uart_error.threshold	Not supported.
out_of_ring.counter	Initialize with 0.
out_of_ring. threshold	1 to 65535; threshold, can be set individually for every event. As soon as the "out_of_ring.counter" reaches the "out_of_ring.threshold" an FDL_EVENT indication with the complete receive buffer is triggered.
sdn_not_indicated.counter	Initialize with 0.
sdn_not_indicated.threshold	Not supported.
duplicate_address.counter	Initialize with 0.
duplicate_address.threshold	1 to 65535; threshold, can be set individually for every event. As soon as the "duplicate_address.counter" reaches the "duplicate_address.threshold" an FDL_EVENT indication with the complete receive buffer is triggered.
hardware_error.counter	Initialize with 0.
hardware_error.threshold	1 to 65535; threshold, can be set individually for every event. As soon as the "hardware_error.counter" reaches the "hardware_error.threshold" an FDL_EVENT indication with the complete receive buffer is triggered.
mac_error.counter	Initialize with 0.
mac_error.threshold	1 to 65535; threshold, can be set individually for every event. As soon as the "mac_error.counter" reaches the "mac_error.threshold" an FDL_EVENT indication with the complete receive buffer is triggered.

Confirmation   The AWAIT_INDICATION confirmation is only returned if an error
                              occurs.

Note   Note that the listed request block header and application block is only
                              valid if an error occurs. If no error occurs, the relevant indications are
                              valid.

                              If the service is successful, there is no confirmation. Instead of that a
                              SDA-, SDN-, SRD- or FDL_EVENT-Indication is send back to the
                              FDL application

                              The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ls, lr, iv
fill_length_1	No significance
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	await_indication
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Unchanged from request
l_status	ls, lr, iv

l_status Values   ls = Negative acknowledgment, SAP does not exist.
                         lr = Negative acknowledgment, resource overflow on the CP
                                (more than 255 for one SAP).
                         iv = Negative acknowledgment:
                                - CP currently being reset
                                - invalid parameters in the request

4.2.11 FDL_EVENT

Indication

With this service, the FDL application is informed of events on the CP. An application block and an event buffer (more than one also possible) must be made available to the CP using the AWAIT_INDICATION service. The CP returns the counter readings providing information about how often the corresponding events occur. The indication is triggered when one of the counters reaches the sensitivity threshold that can be set individually by the FDL application. The FDL application receives the application block and the complete event buffer.

Request Block Header

length	80
user	Unchanged from "await_indication"
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Indication
response	No significance
fill_length_1	LEN_EVENT_INDICATION
seg_length_1	No significance
offset_1	80
fill_length_2	No significance
seg_length_2	No significance
offset_2	No significance

Application Block

opcode	Indication
subsystem	No significance
id	No significance
service.code	fdl_event
loc_add.station	No significance
loc_add.segment	No significance
ssap	EVENT_SAP
dsap	No significance
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

The Structure of the event buffer is described along with the AWAIT_INDICATION service. The buffer is in the user_data_1 structure component.

4.2.12 WITHDRAW_INDICATION

Request   With this service, receive resources transferred previously to the CP by the FDL application with the AWAIT_INDICATION service can be fetched back. These resources normally remain on the CP until data are received from a station (remote). With the WITHDRAW_INDICATION service, the resources can be fetched back prematurely (for example to deactivate the SAP).

Request Block Header

length	80
user	Free for use by FDL application
rb_type	2
priority	Priority of the job low/high
subsystem	22H
opcode	Request
response	No significance
fill_length_1	0
seg_length_1	0
offset_1	No significance
fill_length_2	0
seg_length_2	0
offset_2	No significance

Application Block

opcode	Request
subsystem	Reserved for the CP
id	Reserved for the CP
service.code	withdraw_indication
loc_add.station	No significance
loc_add.segment	No significance
ssap	No significance
dsap	0 to 63 or EVENT_SAP; SAP from which the resources are fetched back
rem_add.station	No significance
rem_add.segment	No significance
serv_class	No significance
receive_l_sdu.length	No significance
send_l_sdu.length	No significance
l_status	No significance

Before deactivating an SAP, the resources must be fetched back. The number of returned resources is transferred in the confirmation in the send_l_sdu.length structure element. The resources must then be fetched with individual "ihi_read" or "SCP_receive" calls.

Confirmation   The WITHDRAW_INDICATION confirmation shows the execution of the WITHDRAW_INDICATION request.

The result of the request is entered in the l_status structure element.

Request Block Header

length	Unchanged from request
user	Unchanged from request
rb_type	Unchanged from request
priority	Unchanged from request
subsystem	22H
opcode	Confirm
response	ok, ls, iv
fill_length_1	Unchanged from request
seg_length_1	Unchanged from request
offset_1	Unchanged from request
fill_length_2	Unchanged from request
seg_length_2	Unchanged from request
offset_2	Unchanged from request

Application Block

opcode	Confirm
subsystem	No significance
id	No significance
service.code	withdraw_indication
loc_add.station	Unchanged from request
loc_add.segment	Unchanged from request
ssap	Unchanged from request
dsap	Unchanged from request
rem_add.station	Unchanged from request
rem_add.segment	Unchanged from request
serv_class	Unchanged from request
receive_l_sdu.length	Unchanged from request
send_l_sdu.length	Number of returned resources (if l_status = ok)
l_status	ok, ls, iv

l_status Values   ok = Positive acknowledgment, service was executed.
                       ls = Negative acknowledgment, SAP does not exist
                       iv = Negative acknowledgment:
                                - CP currently being reset
                                - invalid parameters in the request

Fetching the Resources   The WITHDRAW_INDICATION request is followed by the WITHDRAW_INDICATION confirmation. If this service is successful, (l_status = ok), the structure element send_l_sdu.length contains the number of returned resources. After the confirmation, these must be fetched individually by the FDL application using ihi_read or SCP_receive calls. The request or application block of the returned resource is **unchanged** from the AWAIT_INDICATION request. ✝

5   Access to Layer 2

This chapter illustrates the relationship between interface functions and the FDL services. The chapter also explains how communication is implemented between the local and remote station using productive services.

Basic Structure of the FDL Application

An FDL application has the following basic structure:

```
        ┌─────────────────────┐
        │   ihi_open_dev ()   │
        └─────────────────────┘
                   │
     ┌────────────>│
     │      ┌─────────────────────────────┐
     │      │        Layer 2 jobs         │
     │      │ ( ihi_write() and ihi_read() ) │
     │      └─────────────────────────────┘
     │                 │
     │  no   ╱─────────────────────────────╲
     └──────┤  Finish layer 2 communication ? │
            ╲─────────────────────────────╱
                       │
                      yes
                       │
            ┌─────────────────────┐
            │      ihi_close()    │
            └─────────────────────┘
```
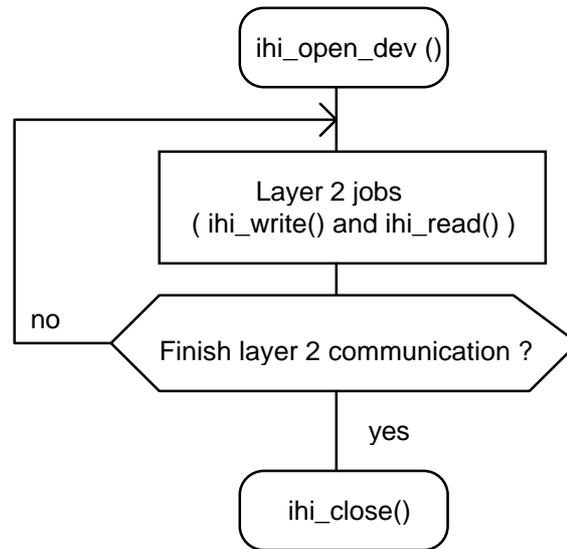
Fig. 5.1: Sequence of an FDL Application

Sequence of Communication

The communication between the FDL application and the CP consists of three essential steps:

1)   The FDL application logs on at the CP with the SCP_open or ihi_open_dev() call.

2)   Layer 2 jobs executed with SCP_send and SCP_receive or ihi_write() and ihi_read().

3)   The FDL application logs off after terminating layer 2 communication with SCP_close or ihi_close().

The interface functions SCP_open(), SCP_send(), SCP_receive(), and SCP_close() are described in Chapter 7 Function Calls of the SCP Interface

The interface functions ihi_open_dev(), ihi_write(), ihi_read() and ihi_close() are described in Chapter 6 Function Calls of the IHI Interface.

5.1   Activating SAPs

Conditions for Data Transfer
Before data can be transmitted or received via the layer 2 interface, one or more SAPs must be activated by one of the management services SAP_ACTIVATE or RSAP_ACTIVATE. SAPs are data interfaces within a PROFIBUS station. The source and destination of a data frame are specified by the PROFIBUS address and the SAP number.

☞ **No data exchange with other PROFIBUS stations is possible without activating SAPs.**

Parameters
To activate an SAP, several parameters must be specified, such as the maximum data length, access rights (remote access, remote SAP), permitted productive services and permitted access type (as initiator or responder).

See also Section 4.2.2 SAP_ACTIVATE.

Default SAP
The default SAP is a special case. If the source and/or destination of a data frame is only specified using the PROFIBUS address, the PROFIBUS station automatically uses the default SAP as the local data interface. As with all other SAPs used for sending or receiving, the default SAP must be activated by the FDL application using the (R)SAP_ACTIVATE service. The number of the default SAP can be read using the FDL_READ_VALUE service. With management services affecting the default SAP, the SAP number must always be specified. On the other hand, with productive services, the constant DEFAULT_SAP as defined in the "fdl_rb.h" include file can be used.

5.2 Data Transfer

Sequence of the
Data Transfer

The FDL application, the FDL protocol software and the remote PROFIBUS stations are involved in data transfer.

To make the situation clearer in the examples, following each request, the application waits for the corresponding result (confirmation/indication). As explained in Chapter 2 and 7, several requests can be sent to layer 2 one after the other and the application only waits for the result after the requests have been sent.

5.2.1 Sending Data Frames

SDA and SRD to a Remote Partner   The CP sends an **acknowledged** data frame to **one** other station.
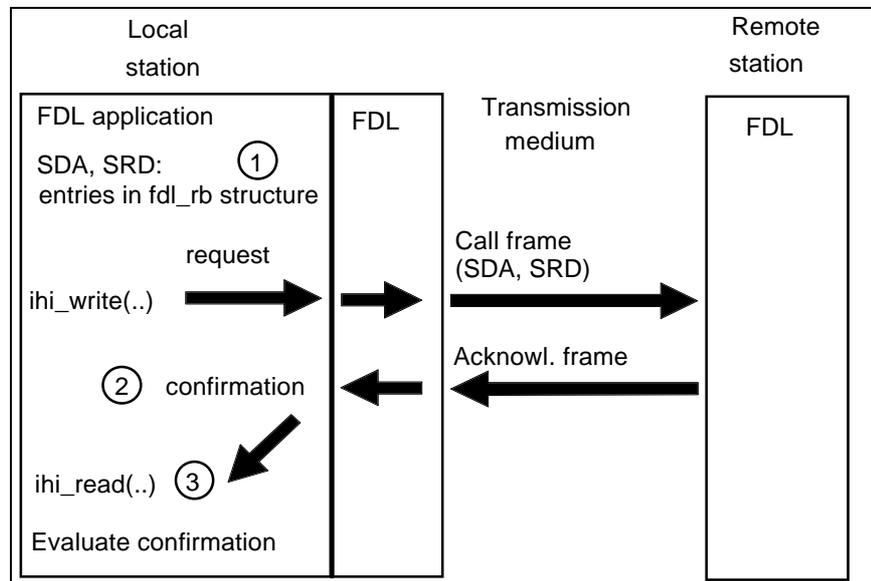

Fig. 5.2: Sending Data Frames SDA, SRD

Notes

① Entries made in the structure according to the service description (Chapter 3 Productive Services).

☞ **Make sure that an offset of at least 12 bytes to the user data is maintained in the send buffer. The size of the offset must be entered in the first byte of the send buffer.**

② After receiving the acknowledgment frame, layer 2 returns the confirmation. If an error occurs (syntax error, remote station does not respond, ...), layer 2 generates a local confirmation.

③ The confirmation must be read out with ihi_read() or SCP_receive(). In the polling mode, ihi_read() or SCP_receive() may need to be called several times.

☞ **If several jobs are processed simultaneously by layer 2, the FDL application should specify the type of structure returned (confirmation/indication) based on the 'opcode' structure element of the 'rb2_header_type' structure. With a confirmation, the assignment to the corresponding request should also be checked.**

SDN to Remote Partner(s)

The CP sends an **unacknowledged** data frame to **one or more** other stations.
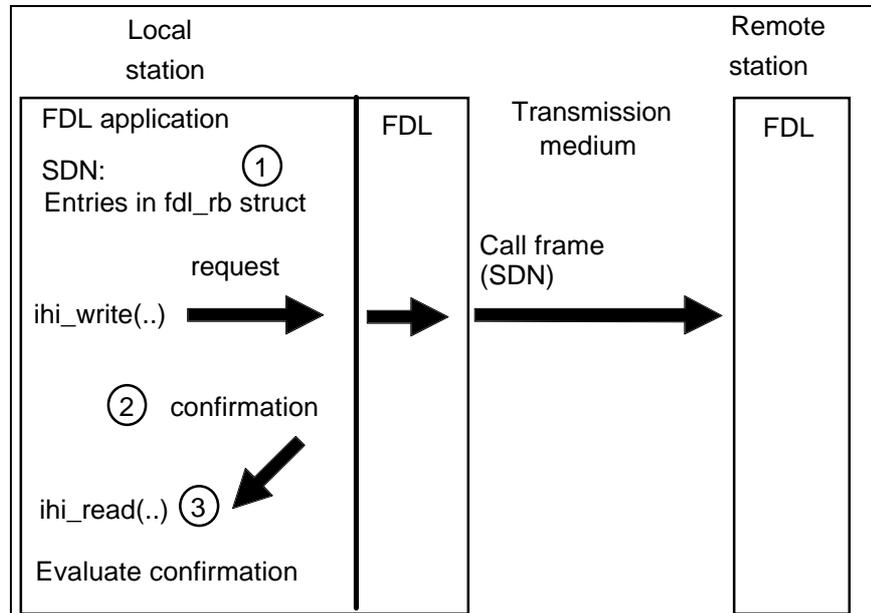
Fig. 5.3: Sending Data Frames SDN

Notes

① Make entries in the structure according to the service description (Chapter 3 Productive Services).

☞ **Make sure that an offset of at least 12 bytes to the user data is maintained in the send buffer. The size of the offset must be entered in the first byte of the send buffer.**

② With unacknowledged services (SDN, SDN_BROADCAST), layer 2 generates a local confirmation after sending the call frame.

③ The confirmation must be read out with ihi_read() or SCP_receive(). In the polling mode, ihi_read() or SCP_receive() may need to be called several times.

☞ **If several jobs are processed simultaneously by layer 2, the FDL application should specify the type of structure returned (confirmation/indication) based on the 'opcode' structure element of the 'rb2_header_type' structure. With a confirmation, the assignment to the corresponding request should also be checked.**

5.2.2 Receiving Data Frames

**SDA, SDN from
Remote Partner**

The CP receives call frames from a remote station.
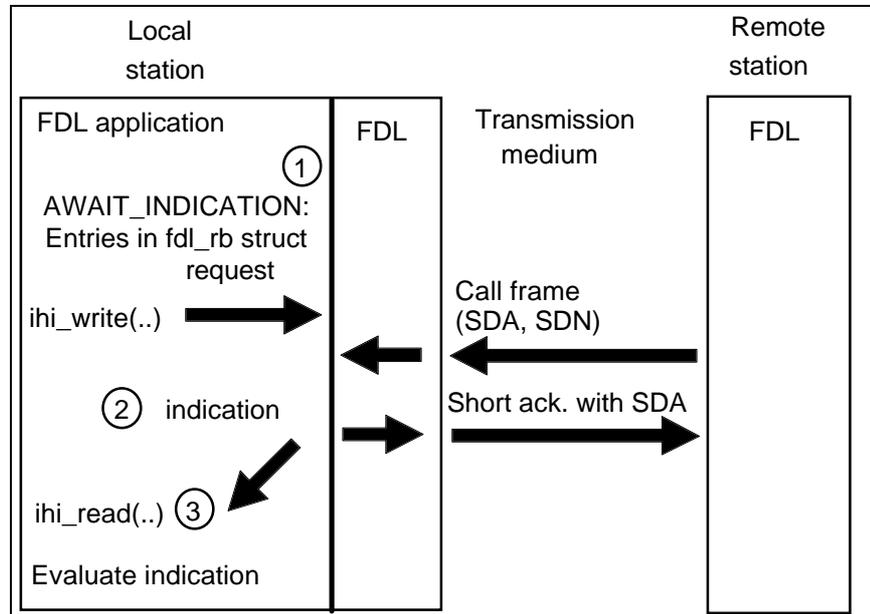
Fig. 5.4: Receiving Data Frames SDA, SDN

Notes

① Entries in the structure according to the service description
(Chapter 3 Productive Services). To be able to receive a data
frame from a remote station, one or more receive resources
must be transferred to the SAP using AWAIT_INDICATION.
Several receive resources can be transferred to the SAP by
repeatedly calling AWAIT_INDICATION. After a call frame has
been received, the resource is used up and must be replaced by
a new AWAIT_INDICATION.

② After receiving a call frame, layer 2 generates an indication
containing the received data and sends it to the FDL application.
The first byte of the receive buffer contains the offset to the
received data.

③ The indication must be read out using ihi_read() or
SCP_receive(). In the polling mode, ihi_read() or SCP_receive()
may need to be called several times.

☞ **If several jobs are processed simultaneously by layer 2, the FDL application should specify the type of structure returned (confirmation/indication) based on the 'opcode' structure element of the 'rb2_header_type' structure.**

☞ **With a confirmation, the assignment to the corresponding request should also be checked. The FDL application must transfer the AWAIT_INDICATION to the CP to continue receiving.**

SRD from Remote Partner   The CP receives a call frame and sends an acknowledgment frame with data back to the remote station.
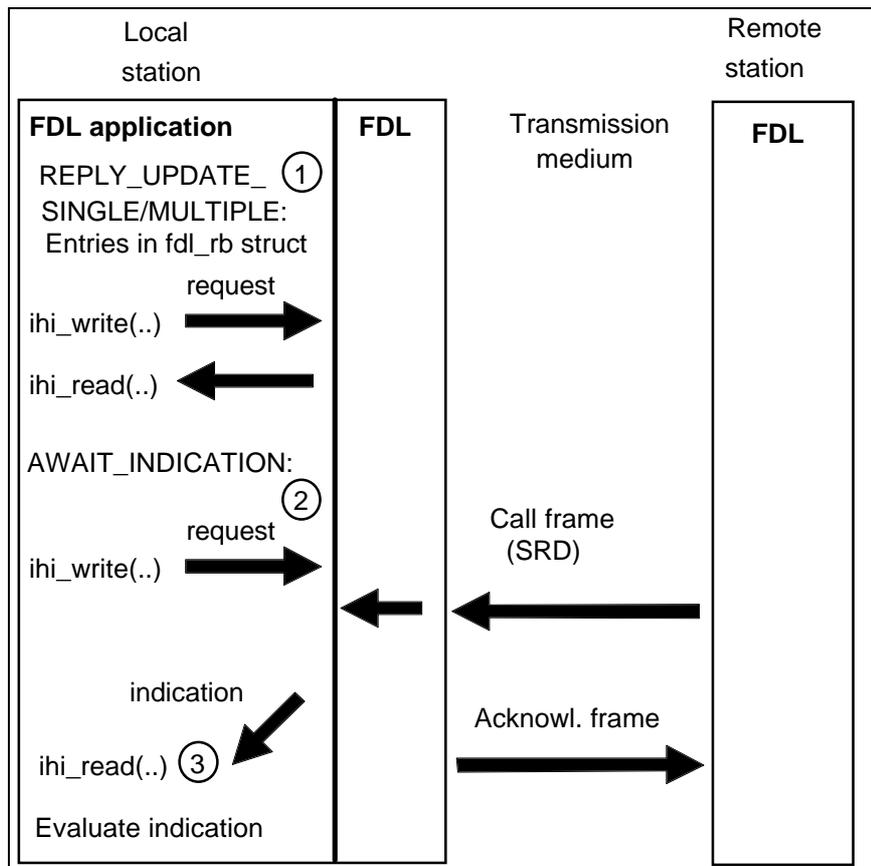
Fig. 5.5: Receiving Data Frames SRD

Notes

① With the REPLY_UPDATE_SINGLE or REPLY_UPDATE_ MULTIPLE service, data is transferred to layer 2 and can be fetched by a remote station using an SRD service. The data is sent to the remote station in the acknowledgment frame. With the REPLY_UPDATE_SINGLE service, the data can only be read out once whereas with the REPLY_UPDATE_ MULTIPLE service it can be read out several times.

☞ **Make sure that an offset of at least 12 bytes to the user data is maintained in the send buffer. The size of the offset must be entered in the first byte of the send buffer.**

If the acknowledgment frame to the remote station does not contain data, ① can be omitted.

② To be able to receive a data frame from a remote station, one or more receive resources must be transferred to the SAP with AWAIT_INDICATION. Several receive resources can be transferred to the SAP by repeatedly calling AWAIT_INDICATION. After receiving a call frame, the resource is used up and must be replaced by a new AWAIT_INDICATION.

③ After receiving a call frame, layer 2 generates an indication containing the received data for the FDL application. The first byte of the receive buffer contains the offset to the received data. The indication can be read out with an ihi_read () or SCP_receive() call.

Notes

6   Function Calls of the IHI Interface

This chapter describes the IHI interface functions with which you transfer FDL jobs and fetch the results.

Under MS-DOS and Windows 3.x, you should only use these calls of the IHI interface.

Under Windows 95/98 and Windows NT, the IHI calls are only intended for porting old applications.

FDL Programming Interface

The FDL programming interface is made available to the FDL application in the form of a library. The library functions of the FDL programming interface handle the transfer of FDL jobs to the CP for the FDL application.

The FDL programming interface involves the following functions:

Logon function for the FDL application	ihi_open_dev
Sending jobs, data	ihi_write
Receiving data (jobs, acknowledgments)	ihi_read
Logoff function for the FDL application	ihi_close

How the Interface Works

The calls for the FDL programming interface must be made by the FDL application. The following order must be maintained: The first interface call is **ihi_open_dev**. Using **ihi_write**, the FDL application can then send jobs to the CP. Each job must then be fetched again with **ihi_read**. Until a request block has been fetched, neither it nor the data buffer appended to it can be used. Finally, the connection to the CP is terminated with **ihi_close**.

6.1 ihi_open_dev

Description of the Function

Using the ihi_open_dev function, the FDL application logs on at the driver. The driver transfers the job to the CP.

The "dev" parameter selects the CP in the PG/PC. The function returns a handle that must be specified for all further calls.

Declaration of the Function

#include "fdl_rb.h"

int   ihi_open _dev (ord16 mode, char * dev);

Description of the Parameters

mode:	mode = 0:	No device name is specified, instead, communication is via the first CP that supports the ihi function calls.
	mode = 1:	A connection is established between the FDL application and the CP selected with the "dev" parameter.
		Recommendation: When using the FDL programming interface, select mode = 1.
dev:		This parameter selects the CP. Syntax: "/name:/FLC"
		name = identical to the name selected in the configuration program.

Return Value

= 0:	**Success:**		Return value = handle
< 0:	**Error:**		
	-1:		Bus driver not installed.
	-2:		Error opening driver.
	-3:		Driver already opened.

6.2   ihi_write

Description of the Function	Using the ihi_write function, request blocks are transferred to the CP for processing.

Declaration of the Function	#include "fdl_rb.h"

int   ihi_write (int handle, RB * rb); |

Description of the Parameters	handle:	Reference (see ihi_open)
	rb:	Address for the request block to be transferred.

Return Value

= 0:	**Success:**	Job transferred correctly to the CP.
< 0:	**Error:**	
	-1:	No ihi_open_dev executed for this handle.
	-2:	Job can no longer be transferred. Maximum number of simultaneous jobs exceeded.
	-3:	No longer occurs.
	-4:	Meaning as for return value -2.
	-5:	Incorrect job, the job was not passed on to the CP.

6.3   ihi_read

Description of the Function

With this call, the FDL application receives back the request blocks processed by the CP. They are returned using a pointer to the request block.

The FDL application has the choice between a synchronous mode in which the call is only completed when a request block is received, and an asynchronous mode, that allows the results to be polled.

Declaration of the Function

#include "fdl_rb.h"

int   ihi_read (int handle, ord16 mode,RB ** rb);

Description of the Parameters

handle:		Reference (see ihi_open)
mode:	mode=0	Asynchronous mode, polling. The function enters the address of an RB in the rb parameter if the return value is 1. Otherwise, the function is terminated with *rb = 0.
	mode=1	Synchronous mode, wait for result. The call is only completed when a request block is returned by the CP.
rb:		Address of a request block pointer returned by the CP.

Return Value

= 0:	**Success:**	Job executed correctly.
= 1:	**Success:**	Job executed correctly. RB transferred.
< 0:	**Error:**	
	-1:	No ihi_open_dev executed for this handle.
	-2:	No jobs exist.
	-3:	Illegal receive mode.
	-4:	No longer occurs.
	-5:	No longer occurs.

☞   **The synchronous mode and asynchronous mode must not be used simultaneously in a program.**

☞   **Under Windows, only the asynchronous mode can be used.**

6.4   ihi_close

Description of the Function   Using the ihi_close function, an FDL application logs off at the driver. Following this call, productive communication is no longer possible with this handle.

Declaration of the Function

#include "fdl_rb.h"

int   ihi_close (int handle);

Description of the Parameters   handle: Reference (see ihi_open).

Return Value   = 0:   **Success:**   Job executed correctly.

Under Windows, ihi_close also returns the value 0 if jobs were discarded.

< 0:   **Error:**

-1:   No ihi_open_dev executed for this handle.

-2:   ihi_close was executed correctly, jobs not yet processed were discarded.

6.5 Examples

Example 1:   Jobs are sent to the CP one after the other.

```
#include "fdl_rb.h"

ex_1 ()
{     int handle;
      RB   rb;     /* request block/
      int ret;
      RB * rb_ptr;

      handle = ihi_open_dev(1,"/CP_L2_1:/FLC");

      if (handle < 0)
      {
          /* error opening the connection to
             the CP 5412 */
      }
      /* make entries in rb */
      ret = ihi_write (handle, &rb);

      if (ret >= 0 )
      {
          /* fetch the request block */
          ret = ihi_read (handle,1,&rb_ptr);
       }

      /* make entries in rb */
      /*(see examples on the diskette) */
      ret = ihi_write (handle, &rb);

      if (ret >= 0 )
      {
          /* fetch the request block */
          ret = ihi_read (handle,1,&rb_ptr);
       }

       /* finish working with PROFIBUS */
       ret = ihi_close (handle);
}
```

Example 2:   Several jobs are processed simultaneously on the CP.

```c
#include "fdl_rb.h"
ex_2 ()
{     int handle;
      RB   rb1;    /* request block*/
      RB   rb2;    /* request block*/
      RB   rb3;    /* request block*/
      int ret;
      RB * rb_ptr;
      int i;

      handle = ihi_open_dev(1,"/CP_L2_1:/FLC");

      if (handle < 0)
      {
          /* error opening the connection to
             the CP5412 */
      }

      /* make entries in rb1 */

      /* senfd rb1 to the CP */
      ret = ihi_write (handle, &rb1);

      /* make entries in rb2 */

      /* send rb2 to the CP */
      ret = ihi_write (handle, &rb2);

      /* make entries in rb3 */

      /* send rb3 to the CP */
      ret = ihi_write (handle, &rb3);
      /* fetch the request blocks */
      for (i = 0; i < 2; i++)
      {
          ret = ihi_read (handle,1,&rb_ptr);
      }

      /* make entries in rb */

      ret = ihi_write (handle, &rb1);

      if (ret >= 0 )
      {
          /* fetch the request block */
          ret = ihi_read (handle,1,&rb_ptr);
      }

      /* finish communication */
      ret = ihi_close (handle);
}
```

7 Function Calls of the SCP Interface

This chapter describes the SCP interface functions with which you transfer FDL jobs and fetch the results.

Under MS-DOS and Windows 3.x, these calls are not available for FDL jobs.

Under Windows 95/98 and Windows NT, the SCP calls are intended for new FDL applications.

FDL Programming Interface

The FDL programming interface is made available to the FDL application in the form of a library. The library functions of the FDL programming interface handle the transfer of FDL jobs to the CP for the FDL application.

The FDL programming interface involves the following functions:

Logon function for the FDL application	SCP_open
Sending jobs, data	SCP_send
Receiving data (jobs, acknowledgments)	SCP_receive
Logoff function for the FDL application	SCP_close
Fetching error IDs	SCP_get_errno

How the Interface Works

The calls for the FDL programming interface must be made by the FDL application. The following order must be maintained: The first interface call is **SCP_open**. Using **SCP_send**, the FDL application can then send jobs to the CP. Each job must then be fetched again with **SCP_receive**. Until a request block has been fetched, neither it nor the data buffer appended to it can be used. Finally, the connection to the CP is terminated with **SCP_close**.

After every function that returns the value -1, **SCP_get_errno** can be called to obtain an error ID that identifies the cause of the error in more detail.

☞ **Please note the points made about the specific operating systems in the Appendix.**

7.1   SCP_open

Description of the Function   Using the SCP_open function, the FDL application logs on at the driver. The driver transfers the job to the CP.

The "dev" parameter selects the CP in the PG/PC. The function returns a handle that must be specified for all further calls.

Declaration of the Function   #include "fdl_rb.h"

int   SCP_open (char * dev);

Description of the Parameters   dev: This parameter selects the CP. Syntax: "/name/FLC"

name = identical to the name selected in the configuration program.

Return Value   = 0:   **Success:**   Return value = handle

= -1:   **Error:**   The exact cause of the error can be obtained with SCP_get_errno().

7.2   SCP_send

Description of the Function

Using the SCP_send function, request blocks are transferred to the CP for processing.

Declaration of the Function

#include "fdl_rb.h"

int   SCP_send (int handle, UWORD length, char * rb);

Description of the Parameters

handle: Reference (see SCP_open)

length: Length of the request block to be transferred in bytes.

rb: Address of the request block to be transferred.

Return Value

= 0:   **Success:**   Job transferred correctly to the CP.

= -1:   **Error:**   The exact cause of the error can be obtained with SCP_get_errno().

7.3   SCP_receive

Description of the Function

With this call, the FDL application receives back job acknowledgments and data from the CP. They are returned in a buffer provided by the application.

The FDL application has the choice between a synchronous mode in which the call is only completed when a request block is received, and an asynchronous mode, that allows the results to be polled.

Declaration of the Function

#include "fdl_rb.h"

int   SCP_receive (int handle, UWORD timeout,
                              UWORD *data_len,
                              UWORD length, char *buffer);

Description of the Parameters	handle:		Reference (see SCP_open)
	timeout:		Waiting time for the receive job. The following values are possible.
		0	Asynchronous mode (SCP_NOWAIT): The function is completed immediately. If no data are available for the caller, then *data_len = 0.
		FFFFh	Synchronous mode (SCP_FOREVER): The call is only completed when data have arrived for the caller.
		0 < timeout < FFFFh	The function is completed when data arrive for the caller or at the latest when a timeout specified in seconds expires.
	data_len:		Pointer to the number of bytes received (return parameter)
	length:		Length of the receive buffer in bytes.
	buffer:		Address of the receive buffer.

Return Value	= 0:	**Success:**	Job executed correctly.
	= -1:	**Error:**	The exact cause of the error can be obtained with SCP_get_errno().

☞   **The synchronous and asynchronous modes must not be used at the same time in a program.**

☞   **In Windows applications, you can only work in the asynchronous mode.**

7.4   SCP_close

Description of the Function Using the SCP_close function, an FDL application logs off at the driver. Following this call, productive communication is no longer possible with this handle.

Declaration of the Function #include "fdl_rb.h"

int   SCP_close (int handle);

Description of the Parameters handle: Reference (see SCP_open).

Return Value

= 0:	**Success:**	Job executed correctly. The value 0 is also returned when pending jobs have been discarded.
= -1:	**Error:**	The exact cause of the error can be obtained with SCP_get_errno().

7.5   SCP_get_errno

Description of the Function   Using the SCP_get_errno function, an application can query the cause of an error that occurred in an SCP function.

Declaration of the Function

#include "fdl_rb.h"

int   WINAPI SCP_get_errno (void);

Description of the Parameters   none

Return Value

=	0:	Last job executed correctly
=	202:	Lack of resources in driver or in the library
=	203:	Configuration error
=	205:	Job not currently permitted
=	206:	Parameter error
=	207:	Device already/not yet open.
=	208:	CP not reacting
=	209:	Error in firmware
=	210:	Lack of memory for driver
=	215:	No message
=	216:	Error accessing application buffer
=	219:	Timeout expired
=	225:	Maximum number of logons exceeded
=	226:	Job aborted
=	233:	An auxiliary program could not be started
=	234:	No authorization exists for this function
=	304:	Initialization not yet completed
=	305:	Function not implemented
=	4865:	CP name does not exist
=	4866:	CP name not configured
=	4867:	Channel name does not exist
=	4868:	Channel name not configured

7.6   Examples

Example 1:   Jobs are sent to the CP one after the other.

```
#include "fdl_rb.h"

exa_1 ()
{     int     handle;
      fdl_rb rb;     /* request block, job block */
      int     ret;
      UWORD  data_len;

      handle = SCP_open ("/CP_L2_1:/FLC");

      if (handle == -1)
      {
          /* error opening the connection to
             the CP */
      }
      /* make entries in rb */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      if (ret == 0)
      {
          /* fetch the acknowledgment/data */
          ret = SCP_receive (handle, 0xffff,
                                &data_len,
                                sizeof(fdl_rb), &rb);
      }

      /* make entries in rb */
      /*(see examples on the diskette) */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      if (ret == 0)
      {
          /* fetch the acknowledgment/data */
          ret = SCP_receive (handle, 0xffff,
                                &data_len,
                                sizeof(fdl_rb), &rb);
      }

      /* finish working with FDL */
      ret = SCP_close (handle);
}
```

Example 2:   Several jobs are processed simultaneously on the CP.

```
#include "fdl_rb.h"
exa_2 ()
{     int    handle;
      fdl_rb rb;    /* request block, job block */
      int    ret;
      UWORD  data_len;
      int    i;

      handle = SCP_open ("/CP_L2_1:/FLC");

      if (handle == -1)
      {
          /* error opening connection to CP */
      }

      /* enter first job in rb */

      /* send rb to the CP */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* enter second job in rb */

      /* send rb to the CP */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* enter third job in rb */

      /* send rb to the CP */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* fetch the request blocks */
      for (i = 0; i < 2; i++)
      {
         ret = SCP_receive (handle, 0xffff,
                            &data_len,
                            sizeof(fdl_rb), &rb);
      }

      /* enter fourth job in rb */

      /* send rb to the CP */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* fetch the request blocks */
      if (ret != -1)
      {
         ret = SCP_receive (handle, 0xffff,
                            &data_len,
                            sizeof(fdl_rb), &rb);
      }

      /* finish communication */
      ret = SCP_close (handle);
}
```

8   Appendix

8.1   Compiling and Linking for Windows 95/98

8.1.1   Working with the MSVC Compilers from Microsoft

Note   Under Windows 95/98, the SCP interface is made available by a DLL.
                          The import library for the MSVC Compilers for Windows 95/98 is
                          s7onlinx.lib. This is in the folder (directory) ..\Fdl.w95\Lib or at the
                          location specified in the installation instructions.

                          The source and make files for compiling the sample program
                          FDLdemo.c are located in the \Fdl.w95\Sample\Fdl folder.

8.2   Compiling and Linking for Windows NT

8.2.1   Working with the MSVC Compilers from Microsoft

Note   Under Windows NT, the SCP interface is made available by a DLL.
                          The import library for the MSVC Compilers for Windows NT is
                          s7onlinx.lib. This is in the directory ..\Fdl.nt\Lib or at the location
                          specified in the installation instructions.

                          The source and make files for compiling the sample program
                          FDLdemo.c are located in the \Fdl.nt\Sample\Fdl folder.

8.3 Special Features for Windows

One of the differences between Windows programs and console programs is that they branch to a WndProc. At a central point, Windows programs wait for Windows messages that are then processed in a WndProc procedure. It is possible that during the processing of the WndProc, control is transferred to Windows and the WndProc is called again.

After SCP_open() in a Windows program, you must call the routine SetSinecHWnd with a Windows handle so that the driver knows where to send its messages. If an asynchronous command is issued, a WM_SINEC message is sent to Windows when a message is received. It can then be processed in the corresponding WndProc if you execute an SCP_receive with timeout 0.

Example of a typical Windows application:

```
WndProc (hWnd,... )
{
int handle;
int ret;
RB * rb_ptr;

   switch (msg)
   {
      case ....  /* init -code */ :
             handle = SCP_open ("/CP_L2_1:/FLC");
             SetSinecHWnd (handle,hWnd);
             break;

      case  .... /* trigger the function */:
             ret = SCP_send (handle,....);
             break;

      case WM_SINEC:
             ret = SCP_receive (handle, 0, &rb_ptr);
             if (ret != -1)
             {
                /* a request block was returned==> */
                /* analyze it                      */
             }
             break;
   }
}
```

Note   Call format for SetSinecHWnd:

SetSinecHWnd (int handle, HWND hWnd)

The function used as an alternative to SetSinecHWnd

SetSinecHWndMsg (int handle,HWND hWnd,
unsigned int msg_id)

allows the FDL application to be informed by the driver when data arrive using a self-defined message (msg_id).

9   Index

Notes

Notes

Glossary

Base address	Logical address of a module in S7 systems.
Bus parameters	Bus parameters control the data transmission on the bus. Each -> station on the -> PROFIBUS network must use bus parameters that match those of other stations.
Bus segment	Part of a -> subnet. Subnets can consist of bus segments and connectivity devices such as repeaters and bridges.
CFB	Communication Function Block: A communication technique for program-controlled transmission of data from or to a CPU in an S7-300/400 using special function blocks. These function blocks were defined based on the IEC 1131-5 draft. The communication partners can be other modules with communication capabilities in an S7-300/400, operator stations, PCs or other controllers and computers.
CP	Communications Processor. Module for communication tasks.
Device master data	Device master data (DMD) contain DP slave descriptions complying with EN 50 170 Vol. 2. Using DMD makes configuration of the -> DP master and -> DP slaves easier.
Distributed I/Os	Input and output modules used at a distance (distributed) from the CPU (central processing unit of the controller). The connection between the programmable controller and the distributed I/Os is established on -> PROFIBUS. The programmable logic controllers do not recognize any difference between these I/Os and local process inputs and outputs.
DP I/O module	DP slaves have a modular design. A -> DP slave has at least one DP I/O module.

DP I/O type	The DP I/O type identifies a -> DP I/O module. The following types exist:
	    Input module
	    Output module
	    Input/output module

DP master	A -> station with master functions in -> PROFIBUS DP. The DP master controls the exchange of user data with the -> DP slaves assigned to it.
DP module list	The DP module list contains the modules belonging to a -> DP slave. You make entries in the DP module list when configuring a -> DP master with -> COM PROFIBUS.
DP module name	Name of a -> DP I/O module entered in the ->DP module list.

DP module type	Type identifier of a -> DP I/O module in the -> device master data of a -> DP slave complying with EN 50 170 Vol. 2.
	A -> station with slave functions in -> PROFIBUS-DP.
DP slave catalog	The DP slave catalog contains the device descriptions of -> DP slaves required for configuring -> DP masters according to the -> DP standard. The DP slave catalog is available when configuring with -> COM PROFIBUS.
DP slave name	A DP slave name is entered in the DP slave list to identify a -> DP slave in the DP configuration.
DP subnet	PROFIBUSsubnet in which only -> distributed I/Os are operated.
DP subsystem	A -> DP master and all -> DP slaves with which this DP master exchanges data.
Driver	Software required for the data transfer between applications and the -> CP.
Enhanced mode	Enhanced mode under 3.x for personal computers with an Intel 386 or compatible processor.
FDL	Fieldbus Data Link. Layer 2 in -> PROFIBUS.
Frame	A message from one PROFIBUS station to another.
Frame header	A frame header consists of an identifier for the -> frame and the source and destination address.
Frame trailer	A frame trailer consists of a checksum and the end identifier of the -> frame.
FREEZE mode	The FREEZE mode is a DP mode in which process data are acquired at the same time and fetched from all (or a group of) DP slaves. The time at which the data are acquired is indicated in the FREEZE command (a synchronization control frame).
Gap update factor	A free address area (gap) between two active -> stations is checked cyclically by the station with the lower -> PROFIBUS address to find out whether or not another station is requesting to enter the logical ring. The cycle time for this check is as follows: gap update factor x target rotation time
Gateway	Intelligent connectivity device that connects different types of local area -> networks at OSI layer 7.

GD circle	A GD circle is a group of -> stations that exchange global data with each other. A -> GD packet is sent to the stations belonging to the GD circle.
GD packet	Collection of data that may be distributed within the programmable logic controller (for example flags/memory bits or data blocks) to be transferred using the -> global data technique.
Global data	Global data (GD) is the name of a communication technique for the cyclic exchange of limited amounts of data from STEP 7 data areas between CPUs of the S7-300/400. Transmitted data can be received by several CPUs at the same time.
Global I/Os	Part of the I/O area of SIMATIC S5 PLCs can be used for global data exchange between SIMATIC S5 PLCs on -> PROFIBUS . The main characteristic of this technique is the cyclic transmission of data that have changed since the last cycle.
Group identifier	DP slaves can be assigned to one or more groups using a group identifier. The -> control frames can be addressed to specific groups of DP slaves using the group identifier.
Highest PROFIBUS address	A -> bus parameter for -> PROFIBUS . This specifies the highest -> PROFIBUS address (HSA) of an active -> station on the PROFIBUS. PROFIBUS addresses higher than the highest station address are possible for passive stations (possible values: HSA 1 to 126).
PROFIBUS address	The PROFIBUS address is a unique identifier for a -> station connected to -> PROFIBUS. The PROFIBUS address is transferred in the -> frame to address a -> station.
Master	An active station in -> PROFIBUS that can send -> frames on its own initiative when it is in possession of the token.
Maximum station delay	A -> bus parameter for -> PROFIBUS . The maximum station delay (max. TSDR) specifies the longest interval required by a -> station in the -> subnet between receiving the last bit of an unacknowledged -> frame and sending the first bit of the next frame. After sending an unacknowledged frame, a sender must wait for the max. TSDR to elapse before sending a further frame.
Minimum station delay	A -> bus parameter for -> PROFIBUS. The minimum station delay (min. TSDR) specifies the minimum time that the receiver of a -> frame must wait before sending the acknowledgment or sending a new frame. The min. TSDR takes into account the longest interval required by a station in the subnet for receiving an acknowledgment after sending a frame.
Network	A network consists of one or more interconnected -> subnets with any number of -> stations. Several networks can exist side by side. There is a common -> node table for every -> subnet.

Node table	The node table applies to all -> networks within a -> system. Each entry in the node table describes the interface between a programmable logic controller (or any other station) and a -> subnet. The entries in the subnet are used by the system to locate and establish connections between stations.
Offset	The length of the reserved area at the beginning of a data buffer of the FDL programming interface.
Process image	The process image is a special memory area in the programmable logic controller. At the start of the cyclic program, the signal states of the input modules are transferred to the process image of the inputs. At the end of the cyclic program, the process image of the outputs is transferred to the output modules
PROFIBUS	A fieldbus complying with EN 50 170 Vol. 2.
PROFIBUS DP	DP mode complying with EN 50 170 Vol. 2.
	PROFIBUS PA is a recommendation of the PROFIBUS users' organization extending PROFIBUS EN 50 170 Vol. 2. to include aspects of intrinsic safety.
Protocol	A set of rules governing data transmission. Using these rules, both the formats of the messages and the data flow during transmission can be specified.
Reorganization token ring	All the -> masters on -> PROFIBUS form a logical token ring. Within this token ring, the token is passed on from station to station. If the transmission of the token is incorrect or if a master is removed from the ring, this leads to an error when the token is passed on (the token is not accepted by this station) and the station is excluded from the ring. The number of exclusions is counted in the internal token_error_counter. If this counter reaches an upper limit value, the logical token ring is then reorganized.
SCOPE PROFIBUS	Diagnostic software for -> PROFIBUS with which the traffic on the -> network can be recorded and analyzed.
Segment	Synonym for -> bus segment.
Services	Services provided by a communication protocol.
Setup time	A -> bus parameter for -> PROFIBUS. The setup time specifies the minimum interval on the sender between receiving an acknowledgment and sending a new call frame.
SIMATIC NET	Siemens Network and Communication. Product name for -> Siemens networks and network components.
PROFIBUS	SIMATIC NET bus system for industrial applications based on PROFIBUS.

PROFIBUS DP	PROFIBUS distributed I/Os. Transmission services complying with PROFIBUS DIN E 19245 Part 3.
PROFIBUS DP master	A -> station with master functions in -> PROFIBUS DP.
PROFIBUS FMS	PROFIBUS Fieldbus Message Specification. Upper sublayer of layer 7 of the ISO/OSI reference model for PROFIBUS.
Slot time	A bus parameter for -> PROFIBUS . The slot time (TSL) is the time during which the sender of a -> frame waits for the acknowledgment from the receiver before detecting a timeout.
Station	A station is identified by an -> PROFIBUS address in the -> PROFIBUS network.
Subnet	A subnet is part of a -> network whose -> bus parameters (for example -> PROFIBUS addresses) must be matched. It includes the bus components and all attached stations. Subnets can, for example, be connected together by -> gateways to form a network. A -> system consists of several subnets with unique -> subnet numbers. A subnet consists of several ->stations with unique -> PROFIBUS addresses.
Subnet number	A -> system consists of several -> subnets with unique subnet numbers.
SYNC mode	The SYNC mode is a DP mode in which several or all -> DP slaves transfer data to their process outputs at a certain time. The time at which the data is transferred is indicated in the SYNC command (a control command for synchronization).
System	All the electrical equipment within a system. A system includes, among other things, programmable logic controllers, devices for operation and monitoring, bus systems, field devices, actuators, supply lines.
Target rotation time	A -> bus parameter for -> PROFIBUS . The token represents the right to transmit for a -> station on PROFIBUS . A station compares the actual token rotation time it has measured with the target rotation time and depending on the result can then send high or low priority frames.
Transmission rate	Transmission rate on the bus (unit in bits per second). A -> bus parameter for -> PROFIBUS. The set or selected transmission rate depends on various conditions, for example distance across the network.
Watchdog	A monitoring time that can be set for a -> DP slave so that it detects the failure of the -> DP master to which it is assigned.

Notes

❑