SIEMENS

SIMATIC NET

FDL-Programmierschnittstelle

Band 1 von 1

C79000-G8900-C072 Ausgabe 5

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.

C79000-G8900-C072

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

Technical data subject to change.

C79000-G8900-C072
Copyright © Siemens AG 1995 bis 2001
All Rights Reserved

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

Nous nous réservons le droit de modifier les caractéristiques techniques.

C79000-G8900-C072
Copyright © Siemens AG 1995 bis 2001
All Rights Reserved

Siemens Aktiengesellschaft

Elektronikwerk Karlsruhe
Printed in the Federal Republic of Germany

SIEMENS

SIMATIC NET

FDL-Programmierschnittstelle

Beschreibung C79000-B8900-C072/05

Hinweis

Wir weisen darauf hin, daß der Inhalt dieser Betriebsanleitung nicht Teil einer früheren oder bestehenden Vereinbarung, Zusage oder eines Rechtsverhältnisses ist oder diese abändern soll. Sämtliche Verpflichtungen von Siemens ergeben sich aus dem jeweiligen Kaufvertrag, der auch die vollständige und allein gültige Gewährleistungsregel enthält. Diese vertraglichen Gewährleistungsbestimmungen werden durch die Ausführungen dieser Betriebsanleitung weder erweitert noch beschränkt.

Wir weisen außerdem darauf hin, daß aus Gründen der Übersichtlichkeit in dieser Betriebsanleitung nicht jede nur erdenkliche Problemstellung im Zusammenhang mit dem Einsatz dieses Gerätes beschrieben werden kann. Sollten Sie weitere Informationen benötigen oder sollten besondere Probleme auftreten, die in der Betriebsanleitung nicht ausführlich genug behandelt werden, können Sie die erforderliche Auskunft über die örtliche Siemens-Niederlassung anfordern.

Allgemeines

Dieses Gerät wird mit Elektrizität betrieben. Beim Betrieb elektrischer Geräte stehen zwangsläufig bestimmte Teile dieser Geräte unter gefährlicher Spannung.

WARNUNG !

Bei Nichtbeachtung der Warnhinweise können deshalb schwere Körperverletzungen und/oder Sachschäden auftreten.

Nur entsprechend qualifiziertes Personal sollte an diesem Gerät oder in dessen Nähe arbeiten. Dieses Personal muß gründlich mit allen Warnungen und Instandhaltungsmaßnahmen gemäß dieser Betriebsanleitung vertraut sein.

Der einwandfreie und sichere Betrieb dieses Gerätes setzt sachgemäßen Transport, fachgerechte Lagerung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Anforderung an die Qualifikation des Personals

Qualifiziertes Personal im Sinne dieser Betriebsanleitung bzw. der Warnhinweise sind Personen, die mit Aufstellung, Montage, Inbetriebsetzung und Betrieb dieses Produktes vertraut sind und über die ihrer Tätigkeit entsprechenden Qualifikation verfügen, wie z. B.:

- Ausbildung oder Unterweisung bzw. Berechtigung, Stromkreise und Geräte bzw. Systeme gemäß den aktuellen Standards der Sicherheitstechnik ein- und auszuschalten, zu erden und zu kennzeichnen;
- Ausbildung oder Unterweisung gemäß an den aktuellen Standards der Sicherheitstechnik in Pflege und Gebrauch angemessener Sicherheitsausrüstungen;
- Schulung in Erster Hilfe.

FDL-Programmierschnittstelle

Diese Benutzeranleitung beschreibt die FDL-Programmierschnittstelle und die verfügbaren Dienste der PROFIBUS-Schicht 2.

Das FDL-Protokoll (**F**ield **D**ata **L**ink-Protokoll) ist für PROFIBUS im offenen, herstellerunabhängigen Kommunikationssystem SIMATIC NET für den Zellen- und Feldbereich mit dem Einsatzschwerpunkt in industrieller Umgebung geeignet.

PROFIBUS (**Pro**cess **Fi**eld **Bus**), basiert auf dem in der Norm EN 50 170 Vol. 2 festgelegten Standard und orientiert sich am ISO/OSI-Referenzmodell.

Mit der Erfüllung der Anforderung nach EN 50 170 Vol. 2 gewährleistet PROFIBUS Offenheit für die Anbindung normgerechter Komponenten anderer Hersteller.

PROFIBUS ist das Netzwerk für den mittleren Leistungsbereich. Mit maximal 127 anschließbaren Teilnehmern ist ein breites Spektrum an Automatisierungsaufgaben erfüllbar. Die Datenrate ist per Software stufenweise einstellbar.

1 Einführung in die FDL-Programmierschnittstelle

Dieses Kapitel stellt das Konzept der FDL-Programmierschnittstelle dar. Es erläutert die grundsätzlichen Mechanismen, deren Verständnis für eine Programmierung einer Applikation erforderlich sind.

Anschließend verfügen Sie über Grundkenntnisse, die Ihnen das Verständnis der weiteren Kapitel erleichtern.

Einführung

Das Programm, das die Dienste der Schicht 2 nutzt, wird FDL-Applikation genannt.

Eine FDL-Applikation kann in der Programmiersprache C oder C++ erstellt werden. Um den Zugriff auf den CP zu ermöglichen, werden Includedateien und Libraries auf der Lieferdiskette mitgeliefert.

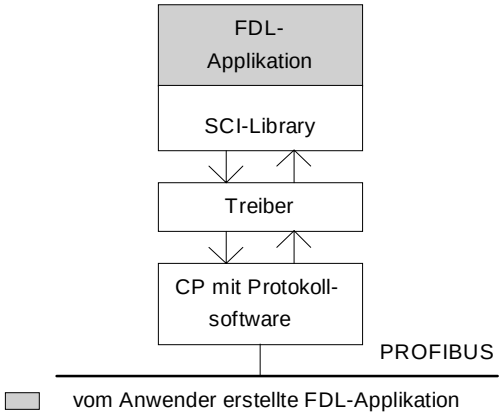


Bild 1-1: Kommunikationsarchitektur der FDL-Programmierschnittstelle

Struktur der Schicht 2

Die Schicht 2 Protokollsoftware von PROFIBUS gliedert sich in die 3 Instanzen FLC (Fieldbus Link Control), FMA (Fieldbus Management), MAC (Media Access Control).

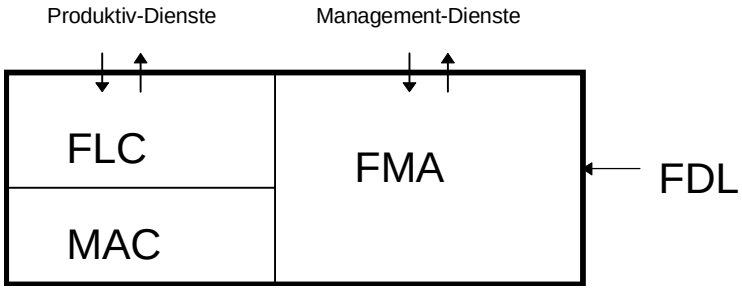


Bild 1-2: Struktur der Schicht-2-Protokoll-Software (FDL)

Über die Instanzen FLC und FMA kann die FDL-Applikation Aufträge an die Schicht 2 übergeben, die ggf. von der MAC-Instanz auf die physikalische Leitung gelangen. Auf umgekehrtem Weg empfängt die MAC-Instanz Bustelegramme, die dann über die Instanzen FLC oder FMA der FDL-Applikation übergeben werden.

FLC Die FLC-Instanz ist für die Entgegennahme der im PROFIBUS beschriebenen Dienste an der FDL-Programmierschnittstelle (Datenübertragungsdienste, Senden und Empfangen von Nachrichten) zuständig. Die Aufträge der FDL-Applikation werden entgegengenommen, bearbeitet (Telegrammaufbereitung etc.) und gegebenenfalls über eine interne Schnittstelle an die MAC-Instanz weitergeleitet.

FMA Die FMA-Instanz ist für die Entgegennahme der im PROFIBUS beschriebenen Managementdienste (administrative Dienste, Parametrierungen, Änderung von Betriebsparametern etc.) zuständig.

MAC Die MAC-Instanz realisiert die vollständige Bus-Zugriffsverwaltung gemäß DIN 19245 Teil 1.

Mechanismus der Schnittstelle

Die FDL-Programmierschnittstelle verwendet für die Auftragsbearbeitung mit dem CP Requestblöcke (Auftragsblöcke, RB). Ein Requestblock beschreibt vollständig einen Auftrag an die FDL-Programmierschnittstelle. Der Requestblock wird mit einer Funktion der SCI-Library an den CP übergeben und später von einer anderen Funktion wieder abgeholt.

Requestblockkennung

Generell übergibt die FDL-Applikation der Schicht 2 einen Requestblock mit der Kennung **request** und erhält je nach Dienst einen Requestblock mit der Kennung **confirm** oder mit der Kennung **indication** zurück.

Requestblock-kennung	Aufgabe des Requestblocks
request	Aufträge der FDL-Applikation an den CP.
confirm	Rückmeldung eines Requests vom CP an die FDL-Applikation.
indication	Meldung eines Ereignisses vom CP an die FDL-Applikation.

Notizen

2 Die FDL-Dienste

In diesem Kapitel erfahren Sie, welche Dienste Ihnen das FDL-Protokoll zur Kommunikation mit anderen Busteilnehmern zur Verfügung stellt.

Beschreibung der FDL-Dienste

Die Dienste der Schicht 2 lassen sich in Produktiv-Dienste und in Management-Dienste unterteilen. Produktiv-Dienste werden in der Produktivphase zum Senden von Datentelegrammen verwendet. Management-Dienste dienen zur Aktivierung/Deaktivierung von lokalen Dienstzugangspunkten (SAP = Service Access Point), zur Bereitstellung von Ressourcen für den Empfang von Datentelegrammen anderer Teilnehmer und sonstigen administrativen Diensten. Die folgenden Tabellen enthalten einen Überblick der verfügbaren FDL-Dienste.

Produktiv-Dienste

SDA (send and request data with acknowledge)	Der CP sendet ein Datentelegramm an einen remote Teilnehmer. Im Positivfall sendet der adressierte remote Teilnehmer eine Quittung zurück. Im Fehlerfall generiert die Schicht 2 eine lokale Fehlermeldung.
SDN (send data with no acknowledge)	Der CP sendet ein Datentelegramm an einen oder mehrere remote Teilnehmer. Im Unterschied zum SDA-Dienst senden die adressierten remote Teilnehmer keine Quittung zurück. Nach dem Senden generiert die Schicht 2 eine lokale Quittung.
SRD (send and request data)	Der CP sendet ein Datentelegramm an einen remote Teilnehmer. Im Positivfall sendet der adressierte remote Teilnehmer eine Quittung zurück. Im Gegensatz zum SDA-Dienst kann der remote Teilnehmer im Antworttelegramm Nutzdaten zurücksenden. Im Fehlerfall generiert die Schicht 2 eine lokale Fehlermeldung.
REPLY_UPDATE_SINGLE	Mit diesem Aufruf werden an die Schicht 2 Daten übergeben, die von einer remote Station ausgelesen werden können. Die Daten werden im Quittungstelegramm eines SRD-Telegramms an einen remote Teilnehmer zurückgesendet. Die Daten können von einer remote Station nur **einmal** ausgelesen werden.
REPLY_UPDATE_MULTIPLE	Ablauf wie REPLY_UPDATE_SINGLE. Unterschied: Die an die Schicht 2 übergebenen Daten können von remote Teilnehmern **mehrfach** ausgelesen werden.

Management-Dienste

SAP_ACTIVATE	Mit diesem Dienst kann in der Schicht 2 ein Dienstzugangspunkt SAP (Service Access Point) aktiviert werden. Die Aktivierung ist Voraussetzung für das Senden und für den Empfang von Datentelegrammen.
RSAP_ACTIVATE	Entspricht dem Aufruf SAP_ACTIVATE. Unterschied: Mit diesem RSAP_ACTIVATE-Dienst kann ein SAP nicht für das aktive Senden von Datentelegrammen eingerichtet werden.
SAP_DEACTIVATE	Mit diesem Dienst kann ein SAP, der durch (R)SAP_ACTIVATE aktiviert wurde, wieder deaktiviert werden. Danach kann über diesen SAP kein Datentransfer mehr erfolgen.
AWAIT_INDICATION	Mit diesem Dienst kann an einen SAP ein Empfangspuffer übergeben werden. Erst danach kann über diesen SAP ein Aufruftelegramm (SDA, SDN, SRD) eines remote Teilnehmers empfangen werden. Nach Empfang eines remote Aufruftelegramms muß ein neuer Empfangspuffer an den SAP übergeben werden.
WITHDRAW_INDICATION	Mit diesem Dienst können Empfangspuffer, die durch AWAIT_INDICATION an einen SAP übergeben worden sind, wieder zurückgeholt werden.
LSAP_STATUS	Dieser Dienst ermittelt die Konfiguration eines SAP der eigenen Station.
FDL_IDENT	Dieser Dienst ermittelt die Identifikation der eigenen oder auch einer remote Station
FDL_LIFE_LIST_CREATE_LOCAL	Der Dienst liefert eine Auflistung der am Bus befindlichen **aktiven** und teilweise der **passiven** Teilnehmer. Die Liste wird nur aus lokalen Informationen innerhalb der Schicht 2 generiert. Es werden keine zusätzlichen Bustelegramme gesendet.
FDL_LIFE_LIST_CREATE_REMOTE	Der Dienst liefert ein Auflistung der am Bus befindlichen aktiven und passiven Teilnehmer. Im Unterschied zu FDL_LIFE_LIST_CREATE_LOCAL wird von jedem möglichen Teilnehmer ein Statustelegramm angefordert (Busbelastung).
FDL_READ_STATISTIC_CTR	Der Dienst dient zum Auslesen von busspezifischen Statistikwerten (ungültige Telegramme etc.).
FDL_EVENT	Mit diesem Dienst werden der FDL-Applikation Ereignisse der Schicht 2 mitgeteilt.
FDL_READ_VALUE	Mit diesem Dienst können die aktuellen Parametrierdaten der Schicht 2 ausgelesen werden.

2.1 Übergabemechanismen

Übergabe an die FDL-Applikation

Die Kommunikation einer FDL-Applikation mit der Schicht 2 erfolgt über drei unterschiedliche Übergabemechanismen:

1) Request der FDL-Applikation an die Schicht 2
2) Confirmation der Schicht 2 an die FDL-Applikation
3) Indication der Schicht 2 an die FDL-Applikation

Request

Ein Request ist eine Dienstanforderung der FDL-Applikation an die Schicht 2. Der Request wird durch einen SCP_send-Aufruf (siehe Kapitel 7.2) oder ihi_write-Aufruf (siehe Kapitel 6.2) an den CP übergeben.

Aufrufparameter der Funktion sind ein 'Handle' und die Adresse des Pointers, der auf eine Requestblock-Struktur *) zeigt. Der Requestblock muß vor dem Aufruf entsprechend der Dienstbeschreibung ausgefüllt werden.

Der Rückgabewert des SCP_send- bzw. des ihi_write-Aufrufes bezieht sich nur auf die korrekte Übergabe des Requests an die Schicht 2 durch den Treiber.

Ob ein Request durch die Schicht 2 fehlerfrei bearbeitet worden ist, kann nur durch die Auswertung der zugehörigen Schicht-2-Confirmation ermittelt werden.

Confirmation

Über eine Confirmation teilt die Schicht 2 der FDL-Applikation das Ergebnis einer Requestbearbeitung mit. Die Confirmation muß durch einen SCP_receive-Aufruf (siehe Kapitel 7.3) oder einen ihi_read-Aufruf (siehe Kapitel 6.3) an den CP ausgelesen werden.

Der Rückgabewert des SCP_receive- bzw. ihi_read-Aufrufes bezieht sich nur auf die korrekte Übergabe der Daten an den Treiber. Das Ergebnis der Requestbearbeitung muß aus dem Requestblock ermittelt werden, der durch SCP_receive oder ihi_read zurückgegeben wird.

Indication

Über eine Indication teilt die Schicht 2 der FDL-Applikation mit, daß von einem remote Teilnehmer ein Aufruftelegramm (SDA, SRD oder SDN) empfangen worden ist. Die Indication muß durch einen SCP_receive-Aufruf oder einen ihi_read-Aufruf an den CP ausgelesen werden *).

Der Rückgabewert des SCP_receive- bzw. ihi_read-Aufrufes bezieht sich nur auf die korrekte Übergabe der Daten an den Treiber. Art und Inhalt der Indication muß aus dem Requestblock ermittelt werden, der durch SCP_receive oder ihi_read zurückgegeben wird.

*) siehe 3.1 Datenstrukturen der Produktiv-Dienste

Requester und Responder

Requester — Station, die eine Auftragsbearbeitung anstößt und auf den Erhalt der Confirmation wartet.

Responder — Station, die von einem remote Teilnehmer ein Datentelegramm empfängt und ein Quittungstelegramm zurücksendet.

Die folgende Tabelle zeigt die möglichen Übergabemechanismen bei den verfügbaren Produktiv- und Management-Diensten beim Requester und Responder.

	Requester		Responder
Dienst	Request	Confirmation	Indication
SDA (Send data with acknowledge)	ja	ja	ja
SDN (Send data with no acknowledge)	ja	ja	ja
SRD (Send and request data)	ja	ja	ja
REPLY UPDATE SINGLE	ja	ja	nein
REPLY UPDATE MULTIPLE	ja	ja	nein
SAP ACTIVATE	ja	ja	nein
RSAP ACTIVATE	ja	ja	nein
SAP DEACTIVATE	ja	ja	nein
AWAIT_INDICATION	ja	Positivfall: nein Fehlerfall: ja	nein
WITHDRAW_INDICATION	ja	ja	nein
LSAP_STATUS	ja	ja	nein
FDL_IDENT	ja	ja	nein
FDL_LIFE_LIST_CREATE_REMOTE	ja	ja	nein
FDL_LIFE_LIST_CREATE_LOCAL	ja	ja	nein
FDL_READ_STATISTIC_COUNTER	ja	ja	nein
FDL_EVENT	nein	nein	ja
FDL_READ_VALUE	ja	ja	nein

Notizen

3 Produktiv-Dienste

In diesem Kapitel werden die grundlegenden Prinzipien der Produktiv-Dienste erläutert.

Das Kapitel zeigt Ihnen,

- ➢ die Datenstrukturen der Produktiv-Dienste
- ➢ die Requestblöcke der Produktiv-Dienst.

Insbesondere werden folgende Produktiv-Dienste detailliert beschrieben:

- ➢ SDA (send data with acknowlegde)
- ➢ SDN (send data with no acknowlegde)
- ➢ SRD (send and request data)
- ➢ REPLY_UPDATE_SINGLE
- ➢ REPLY_UPDATE_MULTIPLE

3.1 Datenstrukturen der Produktiv-Dienste

Requestblock-Aufbau

Die Datenstrukturen sind in dem **Includefile "fdl_rb.h"** definiert.

Die nachfolgend beschriebene Struktur "fdl_rb" ist der Requestblock, der den ihi-Funktionen als Parameter mitgegeben wird.

```
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;
```

Beschreibung der Parameter

rb2_header	Requestblock-Header. Allgemeine, dienstunabhängige Parameter.
application_block	Argumentbereich. FDL-Parameter.
reference	Kennung der FDL-Applikation.
user_data_1	Nettodaten, abhängig vom jeweiligen Auftrag.
user_data_2	Nettodaten, abhängig vom jeweiligen Auftrag.

Unterstruktur Requestblock-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;
```

Beschreibung der Parameter

length	Länge des Requestblocks ohne "user_data_1" und "user_data_2" (= 80 Byte).
user	Benutzerkennung, steht der FDL-Applikation zur Verfügung.
rb_type	Typ des verwendeten Requestblocks (= 2).
priority	Priorität des Auftrags. (identisch mit dem Parameter "serv_class" im Application-Block).
subsystem	Auswahl der Kommunikationsschicht. (FDL = 22h).
opcode	request, confirm, indication (gleich dem Parameter "opcode" im Application-Block).
response	Return-Parameter (gleich dem Parameter "l_status" im Application-Block).
fill_length_1	Anzahl relevanter Bytes im Datenpuffer 1.
seg_length_1	Tatsächliche Länge des Datenpuffers 1.
offset_1	Offset des Datenpuffers 1 bezogen auf den Requestblockanfang.
fill_length_2	Anzahl relevanter Bytes im Datenpuffer 2.
seg_length_2	Tatsächliche Länge des Datenpuffers 2.
offset_2	Offset des Datenpuffers 2 bezogen auf den Requestblockanfang.

Unterstruktur Argumentbereich

```
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;
};
```

Beschreibung der Parameter

opcode	request, confirm, indication
subsystem	Reserviert für den CP.
id	Reserviert für den CP.
service.code	sda, sdn, sdn_broadcast , srd, reply_update_single, reply_update_multiple
loc_add.station	local address 0...126; bei SDN: 127 = MULTICAST/BROADCAST
loc_add.segment	Reserved
ssap	source service access point, 0...62
dsap	destination service access point, 0...63
rem_add.station	remote address, 0...126; bei SDN : 127 = MULTICAST/BROADCAST
rem_add.segment	Reserved
serv_class	Priorität des Services (low oder high)
receive_l_sdu.buffer_ptr	Reserviert für den CP.
receive_l_sdu.length	Pufferlänge, 32...255 (bei request); Nettodatenlänge bei confirm, indication
send_l_sdu.buffer_ptr	Reserviert für den CP.
send_l_sdu.length	Nettodatenlänge des Sendetelegramms.
l_status	return parameter, link_status

Konstanten für den Application-Block

Konstanten, die in diesem Kapitel verwendet werden und der FDL-Applikation zur Verfügung stehen:

DEFAULT_SAP	FFH	Default-SAP-Kennung
NO_SEGMENT	FFH	Segment ungültig
BROADCAST	127	Globaladresse
MULTICAST	127	Globaladresse
LEN_MAX_RECEIVE_BUFFER	255	max. Empfangspuffer
LEN_MIN_RECEIVE_BUFFER	32	min. Empfangspuffer
LEN_DATA_OVERHEAD	14	Länge von maximalem Telegrammheader + -trailer

Vereinbarung

Die folgenden Tabellen geben an, welche Parameter bei den Requestblöcken der Produktiv-Dienste mandatory (m), optional (o), don`t care (x), returned (r) sind:

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-block-header
subsystem	m	m	m	m	
opcode	m	m	m	m	
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-block
service.code	m	m	m	m	
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	Anwenderdaten 1
user_data_2	x	x	m	x	Anwenderdaten 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-block-header
subsystem	r	r	r	r	
opcode	r	r	r	r	
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-block
service.code	r	r	r	r	
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	Anwenderdaten 1
user_data_2	x	x	r	x	Anwenderdaten 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-block-header
subsystem	r	r	r	r	
opcode	r	r	r	r	
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-block
service.code	r	r	r	r	
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	Anwenderdaten 1
user_data_2	x	x	x	x	Anwenderdaten 2

3.2 Requestblöcke der Produktiv-Dienste

3.2.1 SDA (send data with acknowledge)

Request

Die local Station schickt Daten an eine remote Station und erhält eine Bestätigung über den ordnungsgemäßen oder fehlerhaften Datentransfer.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Sendetelegramms low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	Länge der Daten 13...258
seg_length_1	Länge des verwendeten Puffers 15..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	sda
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	0...62 oder DEFAULT_SAP
dsap	0...63 oder DEFAULT_SAP
rem_add.station	0...126
rem_add.segment	Ohne Bedeutung
serv_class	Priorität des Sendetelegramms low/high
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Anzahl der zu übertragenden Netto-Bytes 1...246
l_status	Ohne Bedeutung

Den Aufbau der Daten des SDA-Telegramms können Sie dem folgenden Diagramm entnehmen. Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Die Gesamtlänge des Strukturelements ist im Headerfile "fdl_rb.h" auf 260 Byte festgelegt.

Das Offsetbyte und die Nettodaten müssen von der FDL-Applikation in den Datenpuffer eingetragen werden.

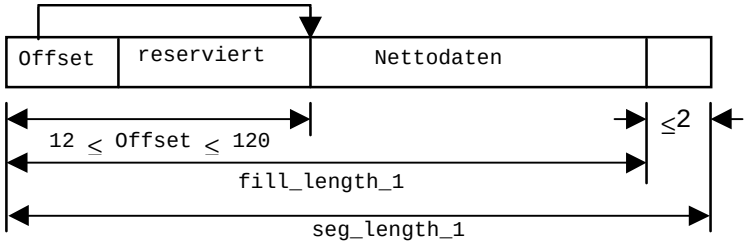
Empfehlung

Wählen Sie 12 für das Offsetbyte.

Wählen Sie 260 als Wert für seg_length_1.

Wählen Sie 12 + Länge der Nettodaten als Wert für fill_length_1.

Aufbau des Sendepuffers



Adreßerweiterung

Das Auftreten von Adreßerweiterungen vermindert die Anzahl der maximal übertragbaren Nettobytes um bis zu 2 Bytes.

Adreßerweiterungen treten dann auf, wenn beim dsap und/oder ssap ein SAP ungleich dem Default-SAP verwendet wird.

Confirm

Die SDA-Confirmation bestätigt die Ausführung des SDA-Requests.

Im Strukturelement l_status wird das Ergebnis des Dienstes hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, rr, ue, rs, ls, na, ds, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Unverändert gegenüber Request
service.code	sda
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Kann gegenüber Request verändert sein
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, rr, ue, rs, ls, na, ds, iv

l_status-Werte

ok = Positive Quittung, Dienst ausgeführt.
rr = Negative Quittung, Betriebsmittel des CPs (remote) nicht verfügbar.
ue = Negative Quittung, FDL-Applikation/FDL-Schnittstellenfehler (remote)
rs = Dienst oder Rem_add beim SAP (remote) nicht aktiviert
ls = Dienst beim SAP (local) nicht aktiviert
na = Keine oder keine plausible Reaktion vom Teilnehmer (remote)
ds = CP (local) nicht im logischen Token-Ring oder von der Busleitung abgetrennt
iv = Ungültige Parameter im Request

Indication

Die SDA-Indication zeigt den Erhalt eines SDA-Requests einer remote Station an.

Im Empfangspuffer sind die empfangenen Daten hinterlegt.

Requestblock-Header

length	80
user	Unverändert gegenüber "await_indication"
rb_type	2
priority	Priorität des Empfangstelegramms low/high
subsystem	22H
opcode	indication
response	Ohne Bedeutung
fill_length_1	Länge der Daten (≤ 258)
seg_length_1	Ohne Bedeutung
offset_1	80
fill_length_2	Ohne Bedeutung
seg_length_2	Ohne Bedeutung
offset_2	Ohne Bedeutung

Application-Block

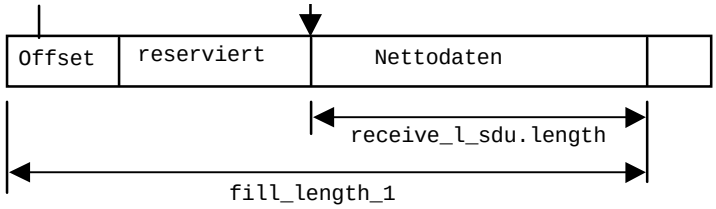
opcode	indication
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	sda
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	SAP der Station (local) 0...63 oder DEFAULT_SAP
dsap	SAP der Station (remote) 0...62 oder DEFAULT_SAP
rem_add.station	Adresse der Quellstation 0...126
rem_add.segment	Ohne Bedeutung
serv_class	Priorität des Empfangstelegramms low/high
receive_l_sdu.length	Länge der empfangenen Netto-Daten 1...246
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Den Aufbau der Daten, die mit der SDA-Indication empfangen wurden, können Sie dem folgenden Diagramm entnehmen.

Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Der Offset und die Nettodaten werden vom CP in den Empfangspuffer eingetragen.

Aufbau des Empfangspuffers



Der Offset (erstes Byte im Empfangspuffer) gibt die Anzahl von Bytes vom Beginn des Empfangspuffers bis zum 1. Byte der Nettodaten an.

3.2.2 SDN (send data with no acknowledge)

Request

Der Teilnehmer (local) schickt Daten an einen Teilnehmer, eine Teilnehmergruppe (MULTICAST) oder an alle Teilnehmer (BROADCAST). Die FDL-Applikation erhält nur eine lokale Confirmation, jedoch keine Empfangsbestätigung der remote Station(en).

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Sendetelegramms low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	Länge der Daten 13...258
seg_length_1	Länge des verwendeten Puffers 15..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	sdn
loc_add.station	Ohne Bedeutung
loc_add.segment	NO_SEGMENT
ssap	0...62 oder DEFAULT_SAP
dsap	0...63 oder DEFAULT_SAP
rem_add.station	0...126 oder MULTICAST (= 127) oder BROADCAST (= 127)
rem_add.segment	NO_SEGMENT
serv_class	Priorität des Sendetelegramms low/high
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Anzahl der zu übertragenden Netto-Bytes 1...246
l_status	Ohne Bedeutung

Bedeutung der Parameter

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

Die Teilnehmergruppen bei MUTICAST werden durch die Dienstzugangspunkte dsap bestimmt.

Den Aufbau der Daten des SDN-Telegramms können Sie dem folgenden Diagramm entnehmen. Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Die Gesamtlänge des Strukturelements ist im Headerfile "fdl_rb.h" auf 260 Byte festgelegt.

Das Offsetbyte und die Nettodaten müssen von der FDL-Applikation in den Datenpuffer eingetragen werden.

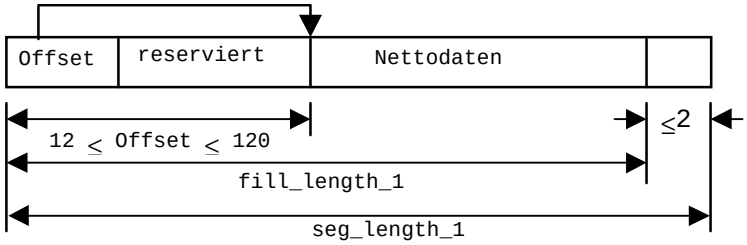
Empfehlung

Wählen Sie 12 für das Offsetbyte.

Wählen Sie 260 als Wert für seg_length_1.

Wählen Sie 12 + Länge der Nettodaten als Wert für fill_length_1.

Aufbau des Sendepuffers



Bemerkung

Das Auftreten von Adreßerweiterungen vermindert die Anzahl der maximal übertragbaren Nettobytes um bis zu 2 Bytes.

Adreßerweiterungen treten dann auf, wenn beim dsap und/oder ssap ein SAP ungleich dem Default-SAP verwendet wird.

Confirmation

Die SDN-Confirmation bestätigt die Ausführung des SDN-Requests.

Im Strukturelement l_status wird das Ergebnis des Dienstes hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, ls, ds, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	sdn
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Kann gegenüber Request verändert sein
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, ls, ds, iv

l_status-Werte

ok = Übertragung der Daten vom CP abgeschlossen
ls = Dienst beim SAP (local) nicht aktiviert
ds = CP nicht im logischen Token-Ring oder von der Busleitung abgetrennt
iv = Ungültige Parameter im Request

Indication

Die SDN-Indication zeigt den Erhalt eines SDN-Requests einer remote Station an.

Im Empfangspuffer sind die empfangenen Daten hinterlegt.

Requestblock-Header

length	80
user	Unverändert gegenüber "await_indication"
rb_type	2
priority	Priorität des Empfangstelegramms low/high
subsystem	22H
opcode	indication
response	Ohne Bedeutung
fill_length_1	Länge der Daten (≤ 258)
seg_length_1	Ohne Bedeutung
offset_1	80
fill_length_2	Ohne Bedeutung
seg_length_2	Ohne Bedeutung
offset_2	Ohne Bedeutung

Application-Block

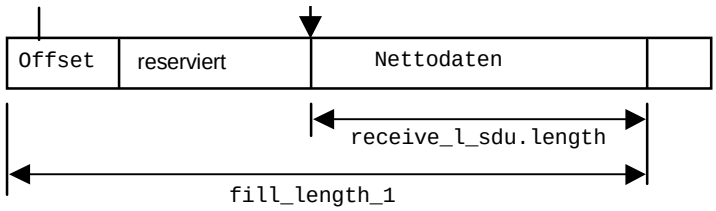
opcode	indication
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	sdn
loc_add.station	Adresse der Zielstation 0...126 (eigene PROFIBUS-Adresse)
loc_add.segment	NO_SEGMENT
ssap	SAP der Station (local) 0...62 oder DEFAULT_SAP
dsap	SAP der Station (remote) 0...62 oder DEFAULT_SAP
rem_add.station	Adresse der Quellstation 0...126
rem_add.segment	NO_SEGMENT
serv_class	Priorität des Empfangstelegramms low/high
receive_l_sdu.length	Länge der empfangenen Netto-Daten 1...246
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Den Aufbau der Daten, die mit der SDN-Indication empfangen wurden, können Sie dem folgenden Diagramm entnehmen.

Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Der Offset und die Nettodaten werden vom CP in den Empfangspuffer eingetragen.

Aufbau des Empfangspuffers



Der Offset (erstes Byte im Empfangspuffer) gibt die Anzahl von Bytes vom Beginn des Empfangspuffers bis zum 1. Byte der Nettodaten an.

Indication (Broadcast, Multicast)

Die SDN_BROADCAST-Indication zeigt den Erhalt eines SDN-Requests einer remote Station an, der an mehrere oder an alle Teilnehmer gesendet wurde.

Im Empfangspuffer sind die empfangenen Daten hinterlegt.

Requestblock-Header

length	80
user	Unverändert gegenüber "await_indication"
rb_type	2
priority	Priorität des Empfangstelegramms low/high
subsystem	22H
opcode	indication
response	Ohne Bedeutung
fill_length_1	Länge der Daten (≤ 258)
seg_length_1	Ohne Bedeutung
offset_1	0
fill_length_2	Ohne Bedeutung
seg_length_2	Ohne Bedeutung
offset_2	Ohne Bedeutung

Application-Block

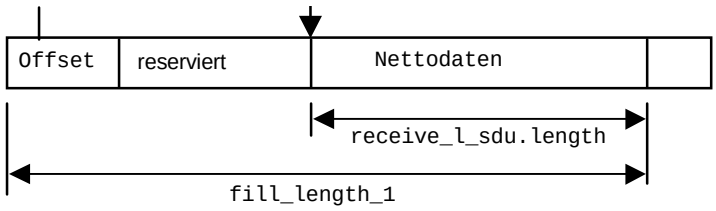
opcode	indication
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	sdn_broadcast
loc_add.station	Adresse der Zielstation BROADCAST
loc_add.segment	NO_SEGMENT
ssap	SAP der Station (local) 0...62: MULTICAST oder 63: BROADCAST
dsap	SAP der Station (remote) 0...62, DEFAULT_SAP
rem_add.station	Adresse der Quellstation 0...126
rem_add.segment	NO_SEGMENT
serv_class	Priorität des Empfangstelegramms low/high
receive_l_sdu.length	Länge der empfangenen Netto-Daten 1...246
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Den Aufbau der Daten, die mit der SDN-Indication empfangen wurden, können Sie dem folgenden Diagramm entnehmen.

Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Der Offset und die Nettodaten werden vom CP in den Empfangspuffer eingetragen.

Aufbau des Empfangspuffers



Der Offset (erstes Byte im Empfangspuffer) gibt die Anzahl von Bytes vom Beginn des Empfangspuffers bis zum 1. Byte der Nettodaten an.

3.2.3 SRD (send and request data)

Request

Die Station (local) schickt Daten an eine remote Station und fordert gleichzeitig Daten von dieser zurück. Sie erhält eine Bestätigung über den Empfang der Daten bei der remote Station und den Erhalt der Antwortdaten.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Sendetelegramms low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	Länge der Daten 12...258
seg_length_1	Länge des verwendeten Sendepuffers 14..260
offset_1	80
fill_length_2	0
seg_length_2	Länge des Empfangspuffers 260
offset_2	Offset vom Beginn des Requestblocks zum Datenpuffer user_data_2

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	srd
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	0...62 oder DEFAULT_SAP
dsap	0...62 oder DEFAULT_SAP
rem_add.station	0...126
rem_add.segment	Ohne Bedeutung
serv_class	Priorität des Sendetelegramms low/high
receive_l_sdu.length	Empfangspufferlänge ≥ max. (LEN_MIN_RECEIVE_BUFFER, erwartete Telegrammlänge) Empfehlung: 255
send_l_sdu.length	Anzahl der zu übertragenden Netto-Bytes 0...246
l_status	Ohne Bedeutung

Den Aufbau der Daten des SRD-Telegramms können Sie dem folgenden Diagramm entnehmen. Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Die Gesamtlänge des Strukturelements ist im Headerfile "fdl_rb.h" auf 260 Byte festgelegt.

Der Offset und die Nettodaten müssen von der FDL-Applikation in den Datenpuffer eingetragen werden.

Empfehlung

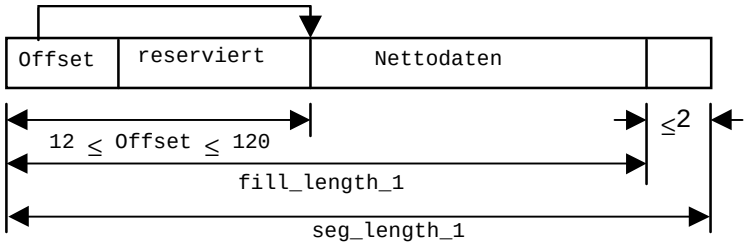
Wählen Sie 12 für das Offsetbyte im Sendepuffer.

Wählen Sie 260 als Wert für seg_length_1.

Wählen Sie 12 + Länge der Nettodaten als Wert für fill_length_1.

Wählen Sie 340 als Wert für offset_2 (offset_1 + seg_length_1). Dadurch wird sichergestellt, daß die empfangenen Daten im Strukturelement user_data_2 eingetragen werden.

Aufbau des Sendepuffers



Aufbau des Empfangspuffers

siehe SRD-Confirmation

Das Auftreten von Adreßerweiterungen vermindert die Anzahl der maximal übertragbaren Netto-Bytes um bis zu 2 Bytes.

Confirmation

Die SRD-Confirmation bestätigt die Ausführung des SRD-Requests.

Im Strukturelement l_status wird das Ergebnis des Dienstes hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ue, rr, rs, dl, nr, dh, rdl, rdh, ls, na, ds, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Anzahl der empfangenen Daten (≤ 258)
seg_length_2	Unverändert gegenüber Request
offset_2	Offset vom Beginn des Requestblocks zum Datenpuffer user_data_2

Application-Block

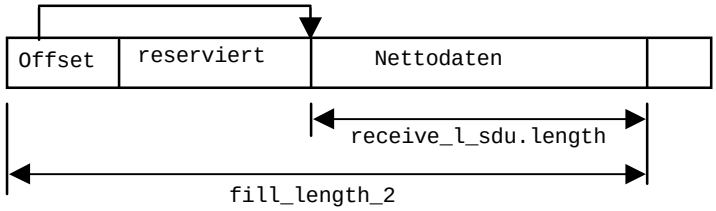
opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	srd
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Anzahl der empfangenen Netto-Bytes 0...246, bei entsprechendem l_status
send_l_sdu.length	Unverändert gegenüber Request
l_status	ue, rr, rs, dl, nr, dh, rdl, rdh, ls, na, ds, iv

Den Aufbau der Daten, die mit der SRD-Confirmation empfangen wurden, können Sie dem folgenden Diagramm entnehmen.

Diese Daten sind im Strukturelement user_data_2 des Requestblocks enthalten.

Der Offset und die Nettodaten werden vom CP in den Empfangspuffer eingetragen.

Aufbau des Empfangspuffers



Der Offset (erstes Byte im Empfangspuffer) gibt die Anzahl von Bytes vom Beginn des Empfangspuffers bis zum 1. Byte der Nettodaten an.

☞ **Beachten Sie, daß hier die fill_length_2 verwendet wird, da die fill_length_1 schon für das Senden der Daten verwendet wird.**

l_status-Werte

- ue = Negativ-Quittung, FDL-Applikation/FDL-Schnittstellenfehler (remote).
- rs = Dienst oder Rem_add beim SAP (remote) nicht aktiviert.
- ls = Dienst beim SAP (local) nicht aktiviert.
- na = Keine oder keine plausible Reaktion vom Teilnehmer (remote).
- ds = CP nicht im logischen Token-Ring oder von der Busleitung abgetrennt.
- iv = Ungültige Parameter im Request.
- dl = Antwortdaten low vorhanden. Positive Quittung für gesendete Daten.
- dh = Antwortdaten high vorhanden. Positive Quittung für gesendete Daten.
- nr = Negative Quittung. Antwortdaten beim CP (remote) nicht verfügbar. Positive Quittung für gesendete Daten.
- rdl = Antwortdaten low vorhanden. Negative Quittung für gesendete Daten, da Betriebsmittel des CPs (remote) nicht verfügbar.
- rdh = Antwortdaten high vorhanden. Negative Quittung für gesendete Daten, da Betriebsmittel des CPs (remote) nicht verfügbar.
- rr = Negative Quittung. Betriebsmittel des CPs (remote) und Antwortdaten (remote) nicht verfügbar.

Indication

Die SRD-Indication bestätigt den Erhalt eines SRD-Requests einer remote Station.

Im Empfangspuffer sind die empfangenen Daten hinterlegt.

Im Strukturelement l_status wird der Update-Status des Dienstes hinterlegt.

Requestblock-Header

length	80
user	Unverändert gegenüber "await_indication"
rb_type	2
priority	Priorität des Empfangstelegramms low/high
subsystem	22H
opcode	indication
response	update_status
fill_length_1	Länge der Daten
seg_length_1	Ohne Bedeutung
offset_1	80
fill_length_2	Ohne Bedeutung
seg_length_2	Ohne Bedeutung
offset_2	Ohne Bedeutung

Application-Block

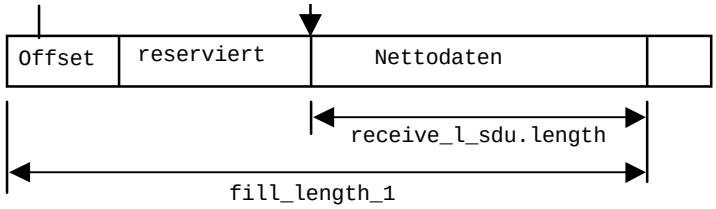
opcode	indication
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	srd
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	SAP der Station (local) 0...62 der DEFAULT_SAP
dsap	SAP der Station (remote) 0...62 der DEFAULT_SAP
rem_add.station	Adresse der Quellstation 0...126
rem_add.segment	Ohne Bedeutung
serv_class	Priorität des Empfangstelegramms low/high
receive_l_sdu.length	Anzahl der empfangenen Netto-Bytes 0...246, bei entsprechendem l_status
send_l_sdu.length	Ohne Bedeutung
l_status	update_status

Den Aufbau der Daten, die mit der SRD-Indication empfangen wurden, können Sie dem folgenden Diagramm entnehmen.

Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Der Offset und die Nettodaten werden vom CP in den Empfangspuffer eingetragen.

Aufbau des Empfangspuffers



Der Offset (erstes Byte im Empfangspuffer) gibt die Anzahl von Bytes vom Beginn des Empfangspuffers bis zum 1. Byte der Nettodaten an.

update_status-Werte

lo	=	Bei dieser SRD-Abwicklung wurde mit low-prioren Daten geantwortet.
hi	=	Bei dieser SRD-Abwicklung wurde mit high-prioren Daten geantwortet.
no_data	=	Bei dieser SRD-Abwicklung wurden keine Antwortdaten übertragen.

3.2.4 REPLY_UPDATE_SINGLE

Request

Mit diesem Dienst stellt die FDL-Applikation Daten für einen bestimmten Dienstzugangspunkt (ssap) bereit. Diese können von jedem anderen Teilnehmer, der Zugriff auf diesen SAP hat, durch einen SRD abgeholt werden. Die Daten werden **nur einmal** übertragen.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Sendetelegramms
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	Länge der Daten 12...258
seg_length_1	Länge des verwendeten Puffers 14..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	reply_update_single
loc_add.station	Ohne Bedeutung
loc_add.segement	Ohne Bedeutung
ssap	0...62 oder DEFAULT_SAP Für diesen SAP werden Daten bereitgestellt
dsap	Ohne Bedeutung
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Priorität des Sendetelegramms low/high
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Anzahl der zu übertragenden Netto-Bytes 0...246
l_status	Ohne Bedeutung

Den Aufbau der Daten des REPLY_UPDATE_SINGLE können Sie dem folgenden Diagramm entnehmen. Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Die Gesamtlänge des Strukturelements ist im Headerfile "fdl_rb.h" auf 260 Byte festgelegt.

Der Offset und die Nettodaten müssen von der FDL-Applikation in den Datenpuffer eingetragen werden.

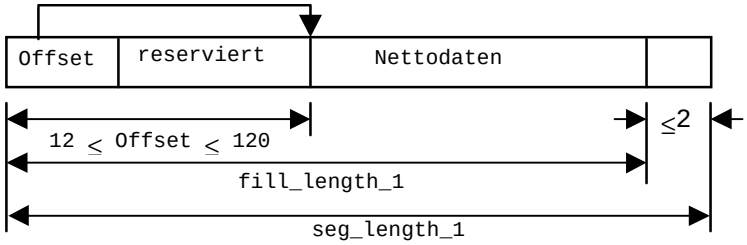
Empfehlung

Wählen Sie 12 für das Offsetbyte im Sendepuffer.

Wählen Sie 260 als Wert für seg_length_1.

Wählen Sie 12 + Länge der Nettodaten als Wert für fill_length_1.

Aufbau des Sende-puffers



Die FDL kann nur entweder einen low- **oder** highprioren Datenpuffer pro SAP bereithalten.

Confirmation

Die REPLY_UPDATE_SINGLE-Confirmation bestätigt die Ausführung des REPLY_UPDATE_SINGLE -Requests.

Im Strukturelement l_status wird das Ergebnis des Dienstes hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, ls, lr, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	reply_update_single
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, ls, lr, iv

l_status-Werte

ok = Datenbereich geladen.
ls = Dienst beim SAP (local) nicht aktiviert.
lr = Response-Ressource wird im Moment vom CP verwendet (temporärer Fehler).
iv = Ungültige Parameter im Request.

Um neue Daten an einen SAP zu übergeben, kann die FDL-Applikation jederzeit den Dienst REPLY_UPDATE_SINGLE verwenden.

Bitte beachten Sie, daß der Auftrag negativ quittiert wird, wenn bereits ein solcher Puffer mit REPLY_UPDATE_SINGLE oder REPLY_UPDATE_MULTIPLE an diesen SAP übergeben wurde **und** dieser Puffer gerade gesendet wird. Der Dienst REPLY_UPDATE_SINGLE ist dann erneut abzusetzen.

3.2.5 REPLY_UPDATE_MULTIPLE

Request

Mit diesem Dienst stellt die FDL-Applikation Daten für einen bestimmten Dienstzugangspunkt (ssap) bereit. Diese können von jedem anderen Teilnehmer, der Zugriff auf diesen SAP hat, durch einen SRD abgeholt werden.

Im Gegensatz zum REPLY_UPDATE_SINGLE-Request können die Daten bei entsprechenden Anforderungen **mehrfach** übertragen werden.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Sendetelegramms low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	Länge der Daten 12...258
seg_length_1	Länge des verwendeten Puffers 14..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	reply_update_multiple
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	0...62 oder DEFAULT_SAP Für diesen SAP werden Daten bereitgestellt
dsap	Ohne Bedeutung
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Priorität des Sendetelegramms low/high
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Anzahl der zu übertragenden Netto-Bytes 0...246
l_status	Ohne Bedeutung

Den Aufbau der Daten des REPLY_UPDATE_MULTIPLE können Sie dem folgenden Diagramm entnehmen. Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Die Gesamtlänge des Strukturelements ist im Headerfile "fdl_rb.h" auf 260 Byte festgelegt.

Der Offset und die Nettodaten müssen von der FDL-Applikation in den Datenpuffer eingetragen werden.

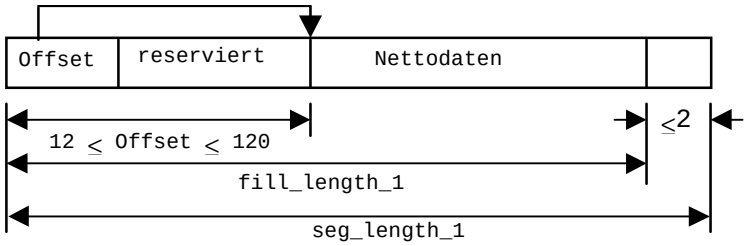
Empfehlung

Wählen Sie 12 für das Offsetbyte im Sendepuffer.

Wählen Sie 260 als Wert für seg_length_1.

Wählen Sie 12 + Länge der Nettodaten als Wert für fill_length_1

Aufbau des Sendepuffers



Die FDL kann nur entweder einen low- **oder** highprioren Datenpuffer pro SAP bereithalten.

Confirmation

Die REPLY_UPDATE_MULTIPLE-Confirmation bestätigt die Ausführung des REPLY_UPDATE_MULTIPLE-Requests.

Im Strukturelement l_status wird das Ergebnis des Dienstes hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, ls, lr, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	reply_update_multiple
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, ls, lr, iv

l_status-Werte

ok = Datenbereich geladen.
ls = Dienst beim SAP (local) nicht aktiviert.
lr = Response-Ressource wird im Moment vom CP verwendet (temporärer Fehler).
iv = Ungültige Parameter im Request.

Um neue Daten an einen SAP zu übergeben, kann die FDL-Applikation jederzeit den Dienst REPLY_UPDATE_MULTIPLE verwenden.

Bitte beachten Sie, daß der Auftrag negativ quittiert wird, wenn bereits ein solcher Puffer mit REPLY_UPDATE_SINGLE oder REPLY_UPDATE_MULTIPLE an diesen SAP übergeben wurde und dieser Puffer gerade gesendet wird. Der Dienst REPLY_UPDATE_MULTIPLE ist dann erneut abzusetzen.

Notizen

4 Management-Dienste

In diesem Kapitel werden die grundlegenden Prinzipien der Management-Dienste erläutert.

Das Kapitel zeigt Ihnen,

- ➢ die Datenstrukturen der Management-Dienste
- ➢ die Requestblöcke der Management-Dienste.

Insbesondere werden folgende Management-Dienste detailliert beschrieben:

- ➢ 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 Datenstrukturen der Management-Dienste

Requestblock-Aufbau

Für die Management-Dienste wird dieselbe Requestblock-Struktur wie für die Produktiv-Dienste verwendet. Aufgrund der anderen Funktionalität existieren neue Servicecodes und die Inhalte der Sende- bzw. Empfangspuffer sind verändert. Wo bisher die von der FDL-Applikation strukturierten Nettodaten der Telegramme abgelegt wurden, werden jetzt die für den jeweiligen Management-Dienst notwendigen Daten in der entsprechenden Struktur hinterlegt.

```
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;
```

Beschreibung der Parameter

rb2_header	Requestblock-Header. Allgemeine, dienstunabhängige Parameter.
application_block	Argumentbereich. FDL-Parameter.
reference	Kennung der FDL-Applikation.
user_data_1	Nettodaten, abhängig vom jeweiligen Auftrag.
user_data_2	Nettodaten, abhängig vom jeweiligen Auftrag.

Unterstruktur Requestblock-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;
```

Beschreibung der Parameter

length	Länge des Requestblocks ohne "user_data_1" und "user_data_2" (= 80 Byte).
user	Benutzerkennung, steht der FDL-Applikation zur Verfügung.
rb_type	Typ des verwendeten Requestblocks (= 2).
priority	Priorität des Auftrags.
subsystem	Auswahl der Kommunikationsschicht (FDL = 22h).
opcode	request, confirm, indication (gleich dem Parameter "opcode" im Application-Block).
response	Return-Parameter (gleich dem Parameter "l_status" im Application-Block).
fill_length_1	Anzahl relevanter Bytes im Datenpuffer 1.
seg_length_1	Tatsächliche Länge des Datenpuffers 1.
offset_1	Offset des Datenpuffers 1 bezogen auf den Requestblockanfang.
fill_length_2	Anzahl relevanter Bytes im Datenpuffer 2.
seg_length_2	Tatsächliche Länge des Datenpuffers 2.
offset_2	Offset des Datenpuffers 2 bezogen auf den Requestblockanfang.

Unterstruktur Argumentbereich

```
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;
};
```

Beschreibung der Parameter

opcode	request, confirm, indication
subsystem	Reserviert für den CP.
id	Reserviert für den CP.
service.code	fdl_read_value, sap_activate, rsap_activate, sap_deactivate, fdl_life_list_create_local, fdl_ident, fdl_event, await_indication, with-draw_indication, lsap_status, fdl_life_list_create_remote
loc_add.station	für Management-Dienste ohne Bedeutung
loc_add.segment	für Management-Dienste ohne Bedeutung
ssap	source service access point, 0...62
dsap	destination service access point für LSAP_STATUS; Nummer des SAPs für (R)SAP_ACTIVATE, SAP_DEACTIVATE (0...63)
rem_add.station	remote address, 0...126, bei FDL_IDENT (remote)
rem_add.segment	Reserved
serv_class	Priorität des Services (low oder high)
receive_l_sdu.length	dienstabhängig
send_l_sdu.length	Ohne Bedeutung
l_status	return parameter, link_status

Sendepuffer

Die Sendepuffer haben bei den unterschiedlichen Aufträgen folgende Bedeutung:

Dienst	verwendete Struktur
Sap_activate	fdl_sap
Rsap_activate	fdl_sap
Sonst	ohne Bedeutung

Rückgabewerte

Die FDL-Applikation erhält folgende Strukturen ausgefüllt zurück:

Dienst	verwendete Struktur
Fdl_read_value	bus_parameter_block
Fdl_event	event_indication
Lsap_status	Byte-Puffer
Fdl_life_list_create_local	Byte-Puffer
fdl_life_list_create_remote	Byte-Puffer
fdl_ident	Byte-Puffer
fdl_read_statistic_ctr	statistic_ctr_list
await_indication	Byte-Puffer
sonst	Ohne Bedeutung

Unterstruktur Busparameterblock

```
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_telegram_length;
};
```

Bedeutung der Parameter

hsa	höchste PROFIBUS-Adresse eines aktiven Teilnehmers am Bus
ts	PROFIBUS-Adresse des Teilnehmers (local)
station_type	Typ des Teilnehmers (local) (aktiv, passiv);
baud_rate	kBaud_9_6 , kBaud_19_2, kBaud_93_75, kBaud_187_5, kBaud_500, Mbaud_1_5, Mbaud_3, Mbaud_6, Mbaud_12
medium_red	Redundanz
retry_ctr	Anzahl der Aufrufwiederholungen an einen nicht antwortenden Responder, 0...7
network_connection_sap	Ohne Bedeutung
default_sap	Nummer des Default-SAP (local)
tsl	SLOT-Time
tqui	Modulatorausklingzeit / Repeater-umschaltzeit
tset	Setup-Zeit
min_tsdr	Minimum der station delay time
max_tsdr	Maximum der station delay time
ttr	target rotation time
g	GAP-Updatefaktor
in_ring_desired	Ringaufnahmewunsch
physical_layer	einstellbare Busphysik
ident	Vendorname, Controllertype, Hard- und Softwarestände

Struktur 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;
};
```

Bedeutung der Parameter

siehe Kapitel 4.2.2

Struktur 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;
};
```

Bedeutung der Parameter

siehe Kapitel 4.2.11

Struktur event_ctr

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

Bedeutung der Parameter

siehe Kapitel 4.2.11

Struktur 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;
};
```

Bedeutung der Parameter

siehe Kapitel 4.2.11

Konstanten

Konstanten, die in diesem Kapitel verwendet werden und der FDL-Applikation zur Verfügung stehen:

Werte für den Application-Block:

DEFAULT_SAP	FFH	Default-SAP-Kennung
NO_SEGMENT	FFH	Segment ungültig
BROADCAST	127	Globaladresse
MULTICAST	127	Globaladresse
EVENT_SAP	64	SAP-Nummer für Events
LEN_MAX_RECEIVE_BUFFER	255	max. Empfangspuffer
LEN_MIN_RECEIVE_BUFFER	32	min. Empfangspuffer
LEN_DATA_OVERHEAD	14	Länge von maximalem Telegrammheader + -trailer

Strukturgrößen für Management-Dienste:

LEN_BUS_PARAMETER	Länge Struktur "bus_parameter_block"
LEN_SAP_ACTIVATE	Länge Struktur "fdl_sap"
LEN_POLL_ELEMENT	Länge Struktur "user_poll_element"
LEN_APPLICATION_BLOCK	Länge Struktur "application_block"
LEN_IDENT	Länge Struktur "ident"
LEN_EVENT_INDICATION	Länge Struktur "event_indication"
LEN_STATISTIC_CTR_LIST	Länge Struktur "statistic_ctr_list"

Konstanten für SAP-Konfigurationen:

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

Konstanten für Life-Liste:

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

4.2 Requestblöcke der Management-Dienste

4.2.1 FDL_READ_VALUE

Request

Mit diesem Dienst können die aktuellen Busparameter des CPs gelesen werden.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	Länge des verwendeten Puffers (≥ LEN_BUS_PARAMETER)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	fdl_read_value
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Ohne Bedeutung
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Confirmation

Die FDL_READ_VALUE-Confirmation bestätigt die Ausführung des FDL_READ_VALUE -Requests.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, iv
fill_length_1	LEN_BUS_PARAMETER
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	fdl_read_value
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, iv

l_status-Werte

ok = Positive Quittung, d.h. Busparameter wurden gelesen.
iv = Negative Quittung:
- CP wird momentan zurückgesetzt
- kein Empfangspuffer

Bedeutung der Parameter

Im Strukturelement user_data_1 werden die in der folgenden Struktur beschriebenen Parameter vom CP eingetragen.

struct bus_parameter_block

hsa	Höchste PROFIBUS-Adresse am Bus, 2...126.
ts	PROFIBUS-Adresse local Teilnehmer, 0...hsa bzw. 126.
station_type	Typ des Teilnehmers (local)
baud_rate	Baudrate: kBaud_9_6, kBaud_19_2, kBaud_93_75, kBaud_187_5, kBaud_500, Mbaud_1_5, Mbaud_3, Mbaud_6, Mbaud_12.
medium_red	Redundanz
retry_ctr	Anzahl der Aufrufwiederholungen an einen nicht antwortenden Teilnehmer (remote), 0...7.
default_sap	Nummer des Default-SAP der Station (local), 0...63.
network_connection_sap	Reserved
tsl	SLOT-Time
tqui	Modulatorausklingzeit / Repeaterumschaltzeit
tset	setup-time
min_tsdr	Minimum der station delay time.
max_tsdr	Maximum der station delay time.
ttr	target rotation time
g	GAP-Updatefaktor
in_ring_desired	Ringaufnahmewunsch
physical_layer	Einstellbare Busphysik
ident	Vendorname, Controllertype, Hard- und Softwarestände

4.2.2 SAP_ACTIVATE

Request

Mit diesem Dienst können Dienstzugangspunkte (SAPs) in der FDL parametriert und aktiviert werden.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	Request
response	Ohne Bedeutung
fill_length_1	LEN_SAP_ACTIVATE
seg_length_1	Länge des verwendeten Puffers (≥ LEN_SAP_ACTIVATE)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	Request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	sap_activate
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Nummer des zu aktivierenden SAPs, 0...63
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Hinweis:

Die Nummer des lokal zu aktivierenden SAPs muß im Application-Block im Parameter DSAP vorgegeben werden (anstelle SSAP).

Strukturelement user_data_1

Im Strukturelement user_data_1 werden die in der folgenden Struktur beschriebenen Parameter von der FDL-Applikation eingetragen.

struct fdl_sap

user_id	Identifikation für FDL-Applikation; ohne Bedeutung für CP.
max_l_sdu_length	Maximale Nettodatenlänge, die über diesen SAP bearbeitet wird. Empfehlung: 246
access_sap	Optionales Zugriffsrecht für einen bestimmten SAP (remote) auf diesen SAP. Andere remote SAPs (≠ access_sap) dürfen nicht zugreifen (0..63, ALL). ALL=keine Zugriffsbeschränkung
access_station	Optionales Zugriffsrecht für eine bestimmte Station (remote) auf diesen SAP. Stationen mit PROFIBUS-Adresse ≠ access_station dürfen nicht zugreifen (0..hsa, ALL). ALL = keine Zugriffsbeschränkung
access_segment	Reserved
sda	Angabe der Role
sdn	Angabe der Role
srd	Angabe der Role
csrd	Reserved
*rup_l_sdu_ptr_low	Ohne Bedeutung
*rup_l_sdu_ptr_high	Ohne Bedeutung

Zu Role:

INITIATOR	Station (local) kann nur Initiator bzgl. des Dienstes sein.
RESPONDER	Station (local) kann nur Responder bzgl. des Dienstes sein.
BOTH_ROLES	Station (local) kann Initiator und Responder bzgl. des Dienstes sein.
SERVICE_NOT_ACTIVATED	Dienst ist nicht aktiviert.

Hinweis:

Ein SAP kann für mehrere Dienste aktiviert werden. Wird jedoch BOTH_ROLES und/oder RESPONDER mehr als einmal eingetragen, so werden alle Einträge (SDA, SDN und SRD) zu BOTH_ROLES.

CSRD wird nicht mehr unterstützt.

Hinweis zu LSAP_STATUS

Der Dienst LSAP_STATUS ermöglicht es, die mit SAP_ACTIVATE eingestellten Roles zu lesen.

Confirmation

Die SAP_ACTIVATE-Confirmation bestätigt die Ausführung des SAP_ACTIVATE-Requests.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, no, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	sap_activate
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, no, iv

l_status-Werte

ok = Positive Quittung, SAP wurde aktiviert.
no = Negative Quittung, SAP existiert bereits.
iv = Negative Quittung:
- CP wird momentan zurückgesetzt
- SAP-Parameter ungültig
- SAP-Nummer ungültig

4.2.3 RSAP_ACTIVATE

Request

Mit diesem Dienst können Dienstzugangspunkte (SAPs) mit reiner Responderfunktionalität für SRD in der FDL parametriert und aktiviert werden.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	LEN_SAP_ACTIVATE
seg_length_1	Länge des verwendeten Puffers (≥ LEN_SAP_ACTIVATE)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	rsap_activate
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Nummer des zu aktivierenden SAPs, 0...63
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Hinweis:

Die Nummer des lokal zu aktivierenden SAPs muß im Application-Block im Parameter DSAP vorgegeben werden (anstelle SSAP).

Strukturelement user_data_1

Im Strukturelement user_data_1 werden die in der folgenden Struktur beschriebenen Parameter von der FDL-Applikation eingetragen.

struct fdl_sap

user_id	Identifikation für FDL-Applikation, ohne Bedeutung für CP.
max_l_sdu_length	Maximale Nettodatenlänge, die über diesen SAP bearbeitet wird (Empfehlung 246).
access_sap	Optionales Zugriffsrecht für einen bestimmten SAP (remote) auf diesen SAP. Andere remote SAPs (≠ access_sap) dürfen nicht zugreifen (0..63, ALL). ALL=keine Zugriffsbeschränkung
access_station	Optionales Zugriffsrecht für eine bestimmte Station (remote) auf diesen SAP. Stationen mit einer PROFIBUS-Adresse ≠ access_station dürfen nicht zugreifen (0..hsa, ALL). ALL = keine Zugriffsbeschränkung
access_segment	Reserved
sda	Ohne Bedeutung
sdn	Ohne Bedeutung
srd	Angabe der Role = RESPONDER
csrd	Reserved
*rup_l_sdu_ptr_low	Ohne Bedeutung
*rup_l_sdu_ptr_high	Ohne Bedeutung

Confirmation

Die RSAP_ACTIVATE-Confirmation bestätigt die Ausführung des RSAP_ACTIVATE -Requests.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, no, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	rsap_activate
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, no, iv

l_status-Werte

ok = Positive Quittung, SAP wurde aktiviert.
no = Negative Quittung, SAP existiert bereits.
iv = Negative Quittung:
- CP wird momentan zurückgesetzt
- SAP-Parameter ungültig
- SAP-Nummer ungültig

4.2.4 SAP_DEACTIVATE

Request

Mit diesem Dienst können Dienstzugangspunkte (SAPs) deaktiviert werden.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	Länge des verwendeten Puffers ≥ LEN_SAP_ACTIVATE
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	sap_deactivate
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Nummer des zu deaktivierenden SAPs, 0...63
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Hinweis:

Die Nummer des lokal zu aktivierenden SAPs muß im Application-Block im Parameter DSAP vorgegeben werden (anstelle SSAP).

Ein SAP darf nur dann deaktiviert werden, wenn keine Ressourcen mehr an den SAP gebunden sind. Puffer die mit vorangegangenen AWAIT_INDICATION-Requests übergeben wurden und noch in der Schicht 2 vorhanden sind, müssen zunächst mit WITHDRAW_INDICATION zurückgeholt werden. Erst danach darf ein SAP_DEACTIVATE durchgeführt werden.

Confirmation

Die SAP_DEACTIVATE-Confirmation bestätigt die Ausführung des SAP_DEACTIVATE-Requests.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, no, lr, iv
fill_length_1	LEN_SAP_ACTIVATE
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	sap_deactivate
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, no, lr, iv

l_status-Werte

ok = Positive Quittung, SAP wurde deaktiviert.
no = Negative Quittung, SAP existiert nicht.
lr = Negative Quittung: CP-Zugriff auf SAP (temporär) es befinden sich noch Indication-Ressourcen im SAP
iv = Negativ-Quittung:
- CP wird momentan zurückgesetzt
- SAP-Nummer ungültig

Daten, die mit "REPLY_UPDATE_..." dem CP für diesen SAP übergeben wurden, werden verworfen.

4.2.5 LSAP_STATUS

Request

Der Dienst erlaubt das Lesen der Konfigurationsparameter für einen bestimmten SAP. Es können nur die SAPs der eigenen Station (local) ausgelesen werden.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	Länge des Empfangspuffers
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	lsap_status
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Nummer des SAPs (local) 0...63 oder DEFAULT_SAP
rem_add.station	ts (eigene PROFIBUS-Adresse)
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Empfangspufferlänge Empfehlung: 255
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Hinweis:

Die Nummer des lokal zu aktivierenden SAPs muß im Application-Block im Parameter DSAP vorgegeben werden (anstelle SSAP).

Confirmation

Die LSAP_STATUS-Confirmation bestätigt die Ausführung des LSAP_STATUS-Requests.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, iv, rs
fill_length_1	Länge der zurückgegebenen Daten + Offset (siehe "Aufbau des Empfangspuffers")
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

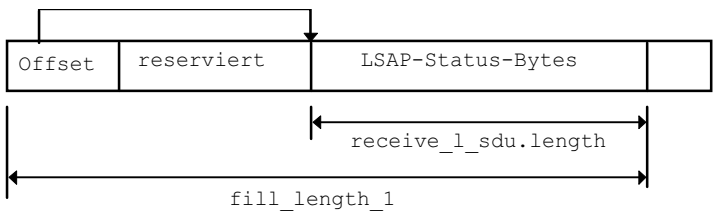
opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	lsap_status
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Anzahl der zurückgegebenen Netto-Bytes 0...6, bei entsprechendem l_status
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, iv, rs

Den Aufbau der Daten, die mit der LSAP_STATUS-Confirmation empfangen wurden, können Sie dem folgenden Diagramm entnehmen.

Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Der Offset und die Nettodaten werden vom CP in den Empfangspuffer eingetragen.

Aufbau des Empfangspuffers



Der Offset (erstes Byte des Empfangspuffers) gibt die Anzahl von Bytes vom Beginn des Empfangspuffers bis zum 1. Byte der Nettodaten an.

l_status-Werte

ok = Positive Quittung, Status wurde gelesen.
rs = Negative Quittung bei CP 5412(A2)/CP 5613/CP 5614:
- Der SAP ist nicht aktiviert

iv = Negative Quittung:
- CP wird momentan zurückgesetzt
- ungültige Parameter im Application-Block
- momentan anderer Management-Dienst aktiv

Bedeutung der Parameter:

Es werden bei l_status = ok 6 Statusbytes gelesen. Die Bytes haben folgende Bedeutung:

BYTE 1:	Teilnehmerzugriffsbeschränkung (access_station)
BYTE 2:	Reserved
BYTE 3...6:	Status der Einzeldienste (SDA, SDN, SRD)

Aufbau von BYTE 1:

b8: Bit 8 ist immer 1.

b7 ... b1: Auf diesen SAP darf nur die Station mit der PROFIBUS-Adresse b7 ... b1 zugreifen. b7 ... b1 = 7FH bedeutet, es besteht keine Zugriffsbeschränkung (ALL).

Aufbau von BYTE 2:

b8 .. b1: Reserved

Aufbau von BYTE 3 .. 5:

b8 ... b5: Geben die Role zum entsprechenden Dienst an:

0000	INITIATOR
0001	RESPONDER
0010	BOTH_ROLES
0011	SERVICE_NOT_ACTIVATED

b4 ... b1: Geben die Dienstkennung an:

0000	SDA_RESERVED
0001	SDN_RESERVED
0011	SRD_RESERVED
0101	Reserved

Aufbau von BYTE 6:

Reserved

Hinweis

Der CSRD wird nicht mehr unterstützt.

4.2.6 FDL_LIFE_LIST_CREATE_REMOTE

Request

Der Dienst liefert der FDL-Applikation eine aktuelle Liste am Bus "funktionsfähiger" Teilnehmer. Es wird an jeden möglichen aktiven oder passiven Teilnehmer ein Statustelegramm gesendet (Busbelastung).

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	Länge des verwendeten Puffers 127..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	fdl_life_liste_create_remote
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Ohne Bedeutung
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Im Gegensatz zu FDL_LIFE_LIST_CREATE_LOCAL liefert die Funktion auch PROFIBUS-Adressen passiver Teilnehmer (Slaves) zurück, mit denen der eigene CP **keinen** Datenverkehr durchführt.

Confirmation

Die FDL_LIFE_LIST_CREATE_REMOTE-Confirmation zeigt die Ausführung des FDL_LIFE_LIST_CREATE_REMOTE-Requests an.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, ds, lr, iv
fill_length_1	127 bei l_status = ok
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	fdl_life_list_create_remote
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	127 bei l_status = ok
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, ds, lr, iv

Aufbau der Life-Liste

user_data_1 → Status Teilnehmer x

(Byte) z. B. 00h	x = 0 z. B. STATION_PASSIVE
(Byte) z. B. 10h	x = 1 z. B. STATION_NON_EXISTANT
(Byte) z. B. 30h	x = 2 z. B. STATION_ACTIVE
(Byte) z. B. 20h	x = 126 z. B. STATION_ACTIVE_READY

Werte für Status

Status:
- STATION_NON_EXISTANT = 10H
- STATION_ACTIVE_READY = 20H (bereit zur Aufnahme in den logischen Ring)
- STATION_ACTIVE = 30H (bereits im logischen Ring)
- STATION_PASSIVE = 00H

l_status-Werte

- ok = Positive Quittung, Life-Liste wurde erstellt.
- ds = CP nicht im logischen Token-Ring oder von der Busleitung abgetrennt.
- lr = Betriebsmittel des CPs nicht verfügbar oder nicht ausreichend.
- iv = Negative Quittung:
  - CP wird momentan zurückgesetzt
  - passiver Teilnehmer
  - momentan anderer Management-Dienst aktiv

4.2.7 FDL_LIFE_LIST_CREATE_LOCAL

Request

Der Dienst liefert der FDL-Applikation eine aktuelle Liste am Bus "funktionsfähiger" Teilnehmer. Die Liste wird aus in dem lokalen Teilnehmer vorhandener Information generiert (keine Busbelastung).

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	Länge des verwendeten Puffers 127..260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	fdl_life_list_create_local
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Ohne Bedeutung
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Im Gegensatz zu FDL_LIFE_LIST_CREATE_REMOTE liefert die Funktion nur PROFIBUS-Adressen aktiver und passiver Teilnehmer (Slaves) zurück, mit denen der eigene CP Datenverkehr durchführt.

Wurde der Dienst FDL_LIFE_LIST_CREATE_REMOTE bereits durchgeführt, so wird ein Abbild aller Teilnehmer geliefert, d.h. bereits eingetragene passive Teilnehmer werden nicht ausgetragen.

Confirmation

Die FDL_LIFE_LIST_CREATE_LOCAL-Confirmation zeigt die Ausführung des FDL_LIFE_LIST_CREATE_LOCAL-Requests an.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

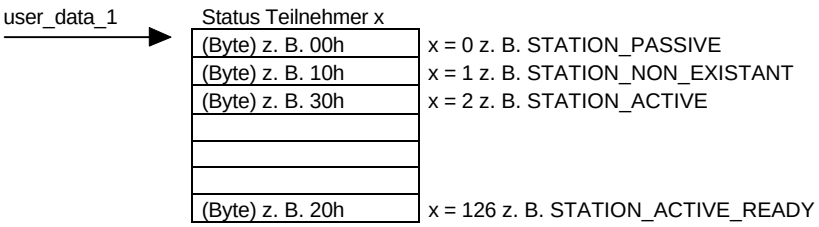
Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, lr, iv
fill_length_1	127 bei l_status = ok
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	fdl_life_list_create_local
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	127 bei l_status = ok
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, lr, iv

Aufbau der Life-Liste:



Status:
STATION_NON_EXISTANT = 10H
STATION_ACTIVE_READY = 20H (bereit zur Aufnahme in den logischen Ring)
STATION_ACTIVE = 30H (bereits im logischen Ring)
STATION_PASSIVE = 00H

l_status-Werte

ok = Positive Quittung, Life-Liste wurde erstellt.
lr = Betriebsmittel des CPs nicht verfügbar oder nicht ausreichend.
iv = Negative Quittung:
- CP wird momentan "FDL_RESET" aktiv
- kein Life-Listen-Puffer vorhanden
- passiver Teilnehmer
- momentan anderer Management-Dienst aktiv

4.2.8 FDL_IDENT

Request

Mit diesem Dienst kann eine Teilnehmeridentifikation (local) Teilnehmers durchgeführt werden. Die Identifikation umfaßt den Herstellernamen, den Anschaltungstyp und die Ausgabestände von Hard- und Software.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	Länge des verwendeten Empfangspuffers (≥ LEN_IDENT ≤ 260)
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	fdl_ident
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Ohne Bedeutung
rem_add.station	0...126; wenn eigene PROFIBUS-Adresse, dann Abfrage des eigenen Idents (local)
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Empfangspufferlänge Empfehlung: 255
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Confirmation

Die FDL_IDENT-Confirmation bestätigt die Ausführung des FDL_IDENT-Requests.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, na, ds, nr, lr, iv
fill_length_1	Länge des Ident (0..200) + Offset (bei l_status = ok)
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

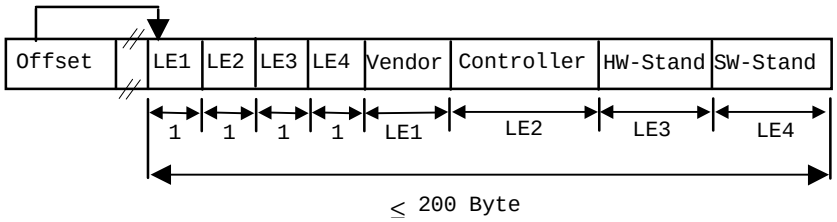
opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	fdl_ident
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Länge des Ident 0...200, bei l_status = ok
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, na, ds, nr, lr, iv

Den Aufbau der Daten, die mit der FDL_IDENT-Confirmation empfangen wurden, können Sie dem folgenden Diagramm entnehmen.

Diese Daten sind im Strukturelement user_data_1 des Requestblocks enthalten.

Der Offset und die Ident-Daten werden vom CP in den Empfangspuffer eingetragen.

Aufbau des Empfangsbuffers



Der Offset (erstes Byte des Empfangspuffers) gibt die Anzahl von Bytes vom Beginn des Empfangspuffers bis zum 1. Byte der Nettodaten an.

Die letzten vier Elemente enthalten Zeichenfolgen.

l_status-Werte

ok	=	Positive Quittung, Ident wurde gelesen.
na	=	Keine oder keine plausible Reaktion vom angesprochenen Teilnehmer (remote).
ds	=	CP nicht im logischen Token-Ring oder von der Busleitung abgetrennt.
nr	=	Negative Quittung für Ident-Daten, da beim angesprochenen Teilnehmer (remote) nicht verfügbar.
lr	=	Betriebsmittel des CPs nicht verfügbar oder nicht ausreichend.
iv	=	Negative Quittung: - CP wird momentan zurückgesetzt - ungültige Parameter im Application-Block - momentan anderer Management-Dienst aktiv

4.2.9 FDL_READ_STATISTIC_COUNTER

Request

Der Dienst dient dem Lesen von Statistik-Daten des lokalen Teilnehmers. Jedes Lesen bewirkt ein Rücksetzen der Zähler.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	Länge des verwendeten Puffers Empfehlung: 260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	fdl_read_statistic_ctr
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	Ohne Bedeutung
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Confirmation

Die FDL_READ_STATISTIC_COUNTER-Confirmation zeigt die Ausführung des FDL_READ_STATISTIC_COUNTER-Requests an.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, iv
fill_length_1	LEN_STATISTIC_CTR_LIST
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	fdl_read_statistic_ctr
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Unverändert gegenüber Request
l_status	ok, iv

Strukturelement user_data_1

Im Strukturelement user_data_1 werden die in der folgenden Struktur beschriebenen Parameter vom CP eingetragen.

struct statistic_ctr_list:

invalid_start_delimiter_ctr	Empfangstelegramm mit ungültigem Startdelimiter.
invalid_fcb_fcv_ctr	Reserved
invalid_token_ctr	Ungültiger Token empfangen.
collision_ctr	Nicht erwartetes Responsetelegramm, möglicherweise Buskollisionen oder Buskurzschluß.
wrong_fcs_or_ed_ctr	Reserved
frame_ error_ctr	Lücke im Empfangstelegramm.
char_error_ctr	Reserved
retry_ctr	Telegrammwiederholungen
start_delimiter_ctr	Empfangstelegramm mit gültigem Startdelimiter.
stop_receive_ctr	Empfang abgebrochen, da: - fehlerhafter Startdelimiter - Buskurzschluß oder Buskollisionen - Teilnehmer doppelt vorhanden - ungültiger Eintrag im Telegramm
send_confirmed_ctr	Anzahl der gesendeten "confirmed requests" (SDA, SRD).
send_sdn_ctr	Anzahl der gesendeten SDN-Requests.

l_status-Werte

ok = Positive Quittung, Statistik wurde gelesen.
iv = Negative Quittung:
- CP wir momentan zurückgesetzt
- kein Statistik-Puffer vorhanden

4.2.10 AWAIT_INDICATION

Request

Mit diesem Dienst wird dem CP eine Ressource für eine Indication zur Verfügung gestellt. Die einzelnen Ressourcen werden SAP-bezogen verwaltet.

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	bei dsap = EVENT_SAP: LEN_EVENT_INDICATION sonst: 0
seg_length_1	Empfehlung: 260
offset_1	80
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	await_indication
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	0...63 oder EVENT_SAP; SAP, für den Ressourcen zur Verfügung gestellt werden
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Empfangspufferlänge Empfehlung: 255
send_l_sdu.buffer_ptr	0
send_l_sdu.length	1
l_status	Ohne Bedeutung

Hat das Strukturelement dsap den Wert EVENT_SAP, wird mit der Ressource eine FDL_EVENT-Indication empfangen. In allen anderen Fällen wird die Ressource für den Empfang einer SDA-, SDN- oder SRD-Indication zur Verfügung gestellt.

Der EVENT-SAP muß, im Gegensatz zu allen anderen SAPs, nicht mit dem Dienst SAP_ACTIVATE eingerichtet werden.

Längen in Abhängigkeit des verwendeten SAPs

	SAP 0..63	EVENT_SAP
fill_length_1	Empfehlung: 0	Empfehlung: LEN_EVENT_INDICATION
seg_length_1	Empfehlung: 260	Empfehlung: 260
receive_l_sdu.length	Empfehlung: 255	Ohne Bedeutung

Bitte beachten Sie, daß auf den Dienst AWAIT_INDICATION nur im Negativfall eine direkte Confirmation mit dem "l_status = ls", "lr", oder "iv" erfolgt. War der gestellte Request korrekt, verbleibt der Requestblock im CP.

Holen Sie Ressourcen mit dem Dienst WITHDRAW_INDICATION zurück, so ist das Strukturelement opcode unverändert, d.h. der Eintrag lautet weiterhin "request".

Bedeutung der Parameter

dsap = EVENT_SAP:	Es wird eine Ressource für die FDL_EVENT-Indication zur Verfügung gestellt. Eine Ressource besteht aus einem Application-Block und einem Empfangspuffer (=LEN_EVENT_INDICATION, struct event_indication).
dsap = 0...63:	Es wird eine Ressource für eine SDA-, SDN- oder SRD-Indication zur Verfügung gestellt. Eine Ressource besteht aus einem Application-Block und einem Empfangspuffer.

Aufbau Empfangspuffer bei FDL_EVENT

Im Strukturelement user_data_1 werden die in der folgenden Struktur beschriebenen Parameter von der FDL-Applikation eingetragen. /1/

struct event_indication

time_out.counter	Mit 0 vorbelegen.
time_out.threshold	1...65535; Empfindlichkeitsschwelle, für jeden Event individuell einstellbar. Sobald der "time_out.counter" den "time_out.threshold" erreicht, wird eine FDL_EVENT-Indication mit dem kompletten Empfangspuffer ausgelöst.
not_syn.counter	Mit 0 vorbelegen.
not_syn.threshold	1...65535; Empfindlichkeitsschwelle, für jeden Event individuell einstellbar. Sobald der "not_syn.counter" den "not_syn.threshold" erreicht, wird eine FDL_EVENT-Indication mit dem kompletten Empfangspuffer ausgelöst.
uart_error.counter	Mit 0 vorbelegen.
uart_error.threshold	Wird nicht unterstützt.
out_of_ring.counter	Mit 0 vorbelegen.
out_of_ring. threshold	1...65535; Empfindlichkeitsschwelle, für jeden Event individuell einstellbar. Sobald der "out_of_ring.counter" den "out_of_ring.threshold" erreicht, wird eine FDL_EVENT-Indication mit dem kompletten Empfangspuffer ausgelöst.
sdn_not_indicated.counter	Mit 0 vorbelegen.
sdn_not_indicated.threshold	Wird nicht unterstützt.
duplicate_address.counter	Mit 0 vorbelegen.
duplicate_address.threshold	1...65535; Empfindlichkeitsschwelle, für jeden Event individuell einstellbar. Sobald der duplicate_address.counter" den "duplicate_address.threshold" erreicht, wird eine FDL_EVENT-Indication mit dem kompletten Empfangspuffer ausgelöst.
hardware_error.counter	Mit 0 vorbelegen.
hardware_error.threshold	1...65535; Empfindlichkeitsschwelle, für jeden Event individuell einstellbar. Sobald der "hardware_error.counter" den "hardware_error.threshold" erreicht, wird eine FDL_EVENT-Indication mit dem kompletten Empfangspuffer ausgelöst.
mac_error.counter	Mit 0 vorbelegen.
mac_error.threshold	1...65535; Empfindlichkeitsschwelle, für jeden Event individuell einstellbar. Sobald der "mac_error.counter" den "mac_error.threshold" erreicht, wird eine FDL_EVENT-Indication mit dem kompletten Empfangspuffer ausgelöst.

Confirmation

Die AWAIT_INDICATION-Confirmation wird nur im Fehlerfall zurückgegeben.

Hinweis:

Beachten Sie, daß der aufgeführte Requestblockheader und Applikationsblock nur im Fehlerfall gütig ist. Im Positivfall sind die entsprechenden Indications zu beachten.
Im Positivfall erfolgt keine Confirmation ! Stattdessen wird eine SDA-, SDN-, SRD-, oder FDL_EVENT-Indication an die FDL-Applikation zurückgegeben

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ls, lr, iv
fill_length_1	Ohne Bedeutung
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	await_indication
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Unverändert gegenüber Request
l_status	ls, lr, iv

l_status-Werte

ls = Negative Quittung, SAP existiert nicht.
lr = Negative Quittung, Ressource-Überlauf im CP (mehr als 255 Ressourcen für einen SAP).
iv = Negative Quittung:
- CP wird momentan zurückgesetzt
- ungültige Parameter im Request

4.2.11 FDL_EVENT

Indication

Mit diesem Dienst werden der FDL-Applikation Ereignisse des CPs mitgeteilt. Dazu müssen dem CP mit dem Dienst AWAIT_INDICATION ein Application-Block und ein Eventpuffer zur Verfügung gestellt werden (auch mehrere möglich). Der CP gibt die Zählerstände, die über die Häufigkeit des Auftretens der entsprechenden Events Auskunft geben, zurück. Die Indication wird dann ausgelöst, wenn einer der Zähler seine, von der FDL-Applikation individuell einstellbare, Empfindlichkeitsschwelle erreicht hat. Die FDL-Applikation erhält den Application-Block mit dem kompletten Eventpuffer zurück.

Requestblock-Header

length	80
user	Unverändert gegenüber "await_indication"
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	indication
response	Ohne Bedeutung
fill_length_1	LEN_EVENT_INDICATION
seg_length_1	Ohne Bedeutung
offset_1	80
fill_length_2	Ohne Bedeutung
seg_length_2	Ohne Bedeutung
offset_2	Ohne Bedeutung

Application-Block

opcode	indication
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	fdl_event
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	EVENT_SAP
dsap	Ohne Bedeutung
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Der Aufbau des Eventpuffers ist beim Dienst AWAIT_INDICATION beschrieben. Der Puffer liegt in der Strukturkomponente user_data_1.

4.2.12 WITHDRAW_INDICATION

Request

Mit diesem Dienst können die Empfangsressourcen von einem SAP zurückgeholt werden, die zuvor von der FDL-Applikation mit dem Dienst AWAIT_INDICATION an den CP übergeben wurden. Diese bleiben normalerweise solange im CP, bis von einem Teilnehmer (remote) Daten empfangen werden. Mit dem Dienst WITHDRAW_INDICATION können die Ressourcen vorzeitig zurückgeholt werden (beispielsweise um den SAP zu deaktivieren).

Requestblock-Header

length	80
user	Frei verwendbar für FDL-Applikation
rb_type	2
priority	Priorität des Auftrags low/high
subsystem	22H
opcode	request
response	Ohne Bedeutung
fill_length_1	0
seg_length_1	0
offset_1	Ohne Bedeutung
fill_length_2	0
seg_length_2	0
offset_2	Ohne Bedeutung

Application-Block

opcode	Request
subsystem	Reserviert für den CP
id	Reserviert für den CP
service.code	withdraw_indication
loc_add.station	Ohne Bedeutung
loc_add.segment	Ohne Bedeutung
ssap	Ohne Bedeutung
dsap	0...63 oder EVENT_SAP; SAP, von dem Ressourcen zurückgeholt werden
rem_add.station	Ohne Bedeutung
rem_add.segment	Ohne Bedeutung
serv_class	Ohne Bedeutung
receive_l_sdu.length	Ohne Bedeutung
send_l_sdu.length	Ohne Bedeutung
l_status	Ohne Bedeutung

Vor der Deaktivierung eines SAPs müssen die Ressourcen zurückgeholt werden. Die Anzahl der zurückgegebenen Ressourcen wird bei der Confirmation in dem Strukturelement send_l_sdu.length übergeben. Die Ressourcen müssen anschließend mit einzelnen “ihi_read"- oder “SCP_receive”-Aufrufen abgeholt werden.

Confirmation

Die WITHDRAW_INDICATION-Confirmation zeigt die Ausführung des WITHDRAW_INDICATION-Requests an.

Im Strukturelement l_status wird das Ergebnis der Ausführung hinterlegt.

Requestblock-Header

length	Unverändert gegenüber Request
user	Unverändert gegenüber Request
rb_type	Unverändert gegenüber Request
priority	Unverändert gegenüber Request
subsystem	22H
opcode	confirm
response	ok, ls, iv
fill_length_1	Unverändert gegenüber Request
seg_length_1	Unverändert gegenüber Request
offset_1	Unverändert gegenüber Request
fill_length_2	Unverändert gegenüber Request
seg_length_2	Unverändert gegenüber Request
offset_2	Unverändert gegenüber Request

Application-Block

opcode	confirm
subsystem	Ohne Bedeutung
id	Ohne Bedeutung
service.code	withdraw_indication
loc_add.station	Unverändert gegenüber Request
loc_add.segment	Unverändert gegenüber Request
ssap	Unverändert gegenüber Request
dsap	Unverändert gegenüber Request
rem_add.station	Unverändert gegenüber Request
rem_add.segment	Unverändert gegenüber Request
serv_class	Unverändert gegenüber Request
receive_l_sdu.length	Unverändert gegenüber Request
send_l_sdu.length	Anzahl der zurückgegebenen Ressourcen (falls l_status = ok)
l_status	ok, ls, iv

l_status-Werte

ok = Positive Quittung, Dienst wurde durchgeführt.
ls = Negative Quittung, SAP existiert nicht.
iv = Negative Quittung:
- CP wird momentan zurückgesetzt
- ungültige Parameter im Request

Abholen der Ressourcen

Nach dem WITHDRAW_INDICATION-Request erfolgt die WITHDRAW_INDICATION-Confirmation. Im Positivfall (l_status = ok) enthält das Strukturelement send_l_sdu.length die Anzahl der zurückgegebenen Ressourcen. Diese müssen im Anschluß an die Confirmation von der FDL-Applikation mit ihi_read- oder SCP_receive-Aufrufen einzeln abgeholt werden. Der Request- bzw. Application-Block der zurückgegebenen Ressource hat sich gegenüber dem AWAIT_INDICATION-Request **nicht** verändert.

5 Zugriff auf die Schicht 2

Dieses Kapitel verdeutlicht den Zusammenhang zwischen den Schnittstellenfunktionen und den FDL-Diensten. Weiterhin erfahren Sie, wie die Kommunikation zwischen dem local und dem remote Teilnehmer bei den Produktiv-Diensten erfolgt.

Prinzipieller Aufbau der FDL-Applikation

Eine FDL-Applikation hat folgenden prinzipiellen Aufbau:

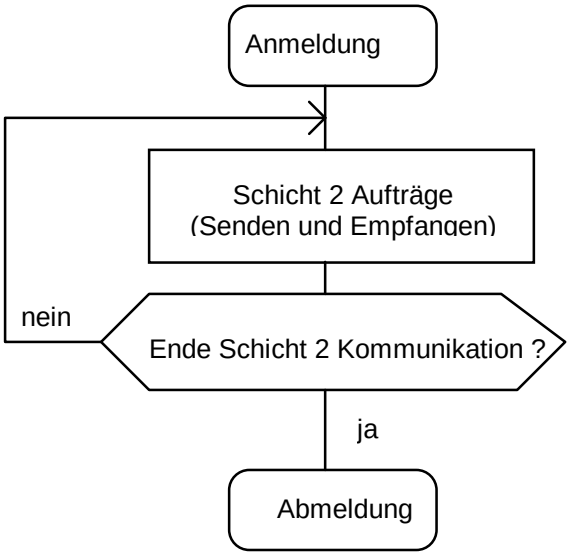


Bild 5-1: Ablaufdiagramm einer FDL-Applikation

Ablauf der Kommunikation

Die Kommunikation der FDL-Applikation und dem CP setzt sich im wesentlichen aus drei Schritten zusammen:

1) Anmeldung der FDL-Applikation beim CP durch den Aufruf von SCP_open() oder ihi_open_dev().

2) Durchführung von Schicht-2-Aufträgen mittels SCP_send() und SCP_receive() oder ihi_write() und ihi_read().

3) Abmeldung der FDL-Applikation nach Beendigung der Schicht 2 -Kommunikation durch SCP_close() oder ihi_close().

Die Schnittstellenfunktionen SCP_open(), SCP_send(), SCP_receive() und SCP_close() sind in Kapitel 7 Funktionsaufrufe der SCP-Schnittstelle beschrieben.

Die Schnittstellenfunktionen ihi_open_dev(), ihi_write(), ihi_read() und ihi_close() sind in Kapitel 6 Funktionsaufrufe der IHI-Schnittstelle beschrieben.

5.1 Aktivieren von SAPs

Vorbedingung für den Datentransfer

Bedingung für das Senden und Empfangen von Daten über die Schicht-2-Schnittstelle ist die Aktivierung eines oder mehrerer SAPs durch einen der Management-Dienste SAP_ACTIVATE oder RSAP_ACTIVATE. SAPs sind lokale Datenschnittstellen innerhalb eines PROFIBUS-Teilnehmers. Die Herkunft und das Ziel eines Datentelegramms werden festgelegt durch PROFIBUS-Adresse und SAP-Nummer.

Ohne Aktivierung von SAPs ist kein Datenverkehr mit anderen PROFIBUS-Teilnehmern möglich.

Parameter

Bei der Aktivierung eines SAPs müssen mehrere Parameter festgelegt werden, wie maximale Datenlänge, Zugriffsrechte (remote Adresse, remote SAP), zulässige Produktiv-Dienste und zulässige Art des Zugriffs (als Initiator oder als Responder).

Siehe hierzu Kapitel 4.2.2 SAP_ACTIVATE.

Default-SAP

Einen Sonderfall stellt der Default-SAP dar. Werden Herkunft und/oder Ziel eines Datentelegramms nur durch die PROFIBUS-Adresse festgelegt, so benutzt der PROFIBUS-Teilnehmer automatisch den Default-SAP als lokale Datenschnittstelle. Der Default-SAP muß, wie alle anderen SAPs, die für das Senden oder Empfangen genutzt werden sollen, von der FDL-Applikation mit dem Dienst (R)SAP_ACTIVATE aktiviert werden. Die Nummer des Default-SAPs kann mit dem Dienst FDL_READ_VALUE gelesen werden. Bei Management-Diensten, die den Default-SAP betreffen, ist immer die SAP-Nummer anzugeben. Bei Produktivdiensten kann dagegen die Konstante DEFAULT_SAP genutzt werden, die im Includefile "fdl_rb.h" definiert ist.

5.2 Datentransfer

Ablauf des Datentransfers

Beim Datentransfer sind die FDL-Applikation, die FDL-Protokollsoftware und remote PROFIBUS-Teilnehmer beteiligt.

Zur besseren Übersicht wird in den Beispielen nach jedem Request sofort auf das zugehörige Ereignis (Confirmation/Indication) gewartet. Wie in Kapitel 6 und 7 erläutert wird, können auch mehrere Requests in Folge an die Schicht 2 übergeben werden und erst danach auf das Ereignis gewartet werden.

5.2.1 Senden von Datentelegrammen

SDA und SRD an remote Partner

Der CP sendet ein **quittiertes** Datentelegramm an **einen** anderen Teilnehmer.

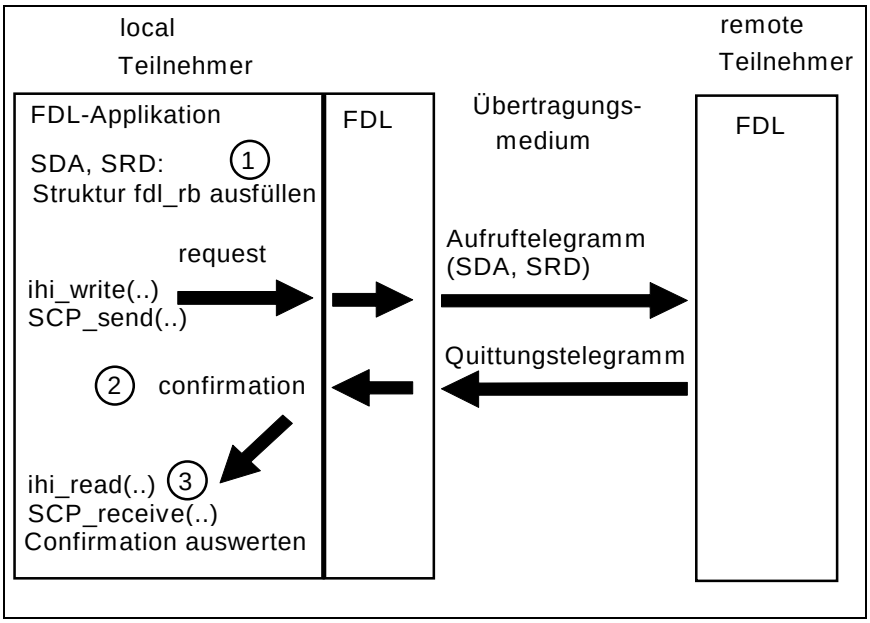


Bild 5-2: Senden von Datentelegrammen SDA, SRD

Anmerkungen

(1) Vorbelegung der Struktur entsprechend der Dienstbeschreibung (Kapitel 3 Produktiv-Dienste).

☞ **Es muß darauf geachtet werden, daß im Sendepuffer ein Offset von mindestens 12 Byte auf die Nettodaten eingehalten wird. Im ersten Byte des Sendepuffers muß die Größe des Offsets eingetragen werden.**

(2) Nach Empfang des Quittungstelegramms gibt die Schicht 2 die Confirmation zurück. Im Fehlerfall (Syntaxfehler, remote Station antwortet nicht, ...) generiert die Schicht 2 eine lokale Confirmation.

(3) Die Confirmation muß durch ihi_read() oder SCP_receive() ausgelesen werden. Im Polling-Mode muß ihi_read() bzw. SCP_receive() u. U. mehrfach aufgerufen werden.

☞ **Werden mehrere Aufträge gleichzeitig durch die Schicht 2 bearbeitet, sollte die FDL-Applikation anhand des Strukturelements 'opcode' der Struktur 'rb2_header_type' die Art (Confirmation/Indication) der zurückgegebenen Struktur bestimmen. Im Falle einer Confirmation ist zusätzlich die Zuordnung zum zugehörigen Request zu prüfen.**

SDN an remote Partner

Der CP sendet ein **unquittiertes** Datentelegramm an **einen oder mehrere** andere Teilnehmer.

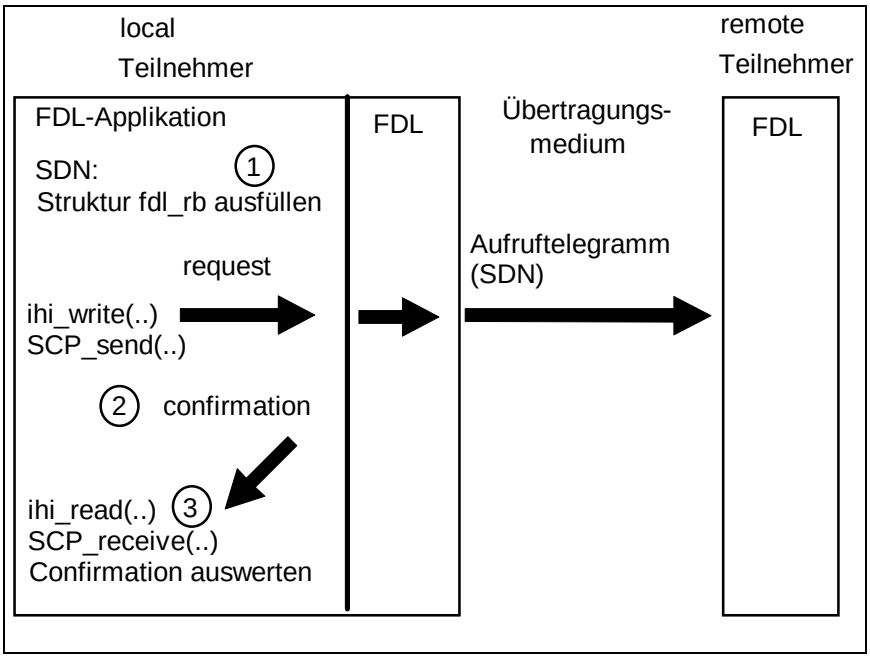


Bild 5-3: Senden von Datentelegrammen SDN

Anmerkungen

(1) Vorbelegung der Struktur entsprechend der Dienstbeschreibung (Kapitel 3 Produktiv-Dienste).

☞ **Es muß darauf geachtet werden, daß im Sendepuffer ein Offset von mindestens 12 Byte auf die Nettodaten eingehalten wird. Im ersten Byte des Sendepuffers muß die Größe des Offsets eingetragen werden.**

(2) Bei unquittierten Diensten (SDN, SDN_BROADCAST) generiert die Schicht 2 nach dem Senden des Aufruftelegramms eine lokale Confirmation.

(3) Die Confirmation muß durch ihi_read() oder SCP_receive() ausgelesen werden. Im Polling-Mode muß ihi_read() bzw. SCP_receive() u. U. mehrfach aufgerufen werden.

☞ **Werden mehrere Aufträge gleichzeitig durch die Schicht 2 bearbeitet, sollte die FDL-Applikation anhand des Strukturelements 'opcode' der Struktur 'rb2_header_type' die Art (Confirmation/Indication) der zurückgegebenen Struktur bestimmen. Im Falle einer Confirmation ist zusätzlich die Zuordnung zum zugehörigen Request zu prüfen.**

5.2.2 Empfang von Datentelegrammen

SDA, SDN von remote Partner

Der CP empfängt Aufruftelegramme von einem remote Teilnehmer.

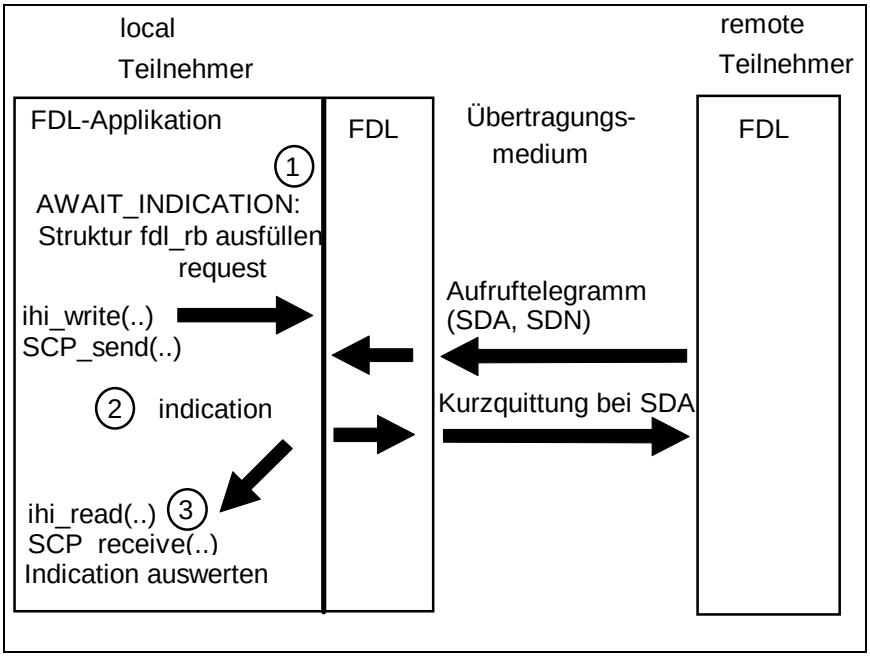


Bild 5-4: Empfang von Datentelegrammen SDA, SDN

Anmerkungen

(1) Vorbelegung der Struktur entsprechend der Dienstbeschreibung (Kapitel 3 Produktiv-Dienste). Um ein Datentelegramm eines remote Teilnehmers empfangen zu können, muß an den entsprechenden SAP durch AWAIT_INDICATION eine oder mehrere Empfangsressourcen übergeben werden. Mehrere Empfangsressourcen können durch wiederholten Aufruf von AWAIT_INDICATION an den SAP übergeben werden. Nach dem Empfang eines Aufruftelegramms ist die entsprechende Ressource verbraucht und muß durch einen neuen AWAIT_INDICATION ersetzt werden.

(2) Nach dem Empfang eines Aufruftelegramms generiert die Schicht 2 eine Indication mit den empfangenen Daten an die FDL-Applikation. Dabei enthält das erste Byte des Empfangspuffers den Offset auf die empfangenen Daten.

(3) Die Indication muß durch ihi_read() oder SCP_receive() ausgelesen werden. Im Polling-Mode muß ihi_read() bzw. SCP_receive() u. U. mehrfach aufgerufen werden.

☞ **Werden mehrere Aufträge gleichzeitig durch die Schicht 2 bearbeitet, sollte die FDL-Applikation anhand des Strukturelements 'opcode' der Struktur 'rb2_header_type' die Art (Confirmation/Indication) der zurückgegebenen Struktur bestimmen.**

☞ **Im Falle einer Confirmation ist zusätzlich die Zuordnung zum zugehörigen Request zu prüfen. Die FDL-Applikation muß für einen weiteren Empfang den Dienst AWAIT_INDICATION an den CP übergeben.**

SRD von remote Partner

Der CP empfängt ein Aufruftelegramm und sendet ein Quittungstelegramm mit Daten an den remote Teilnehmer zurück.

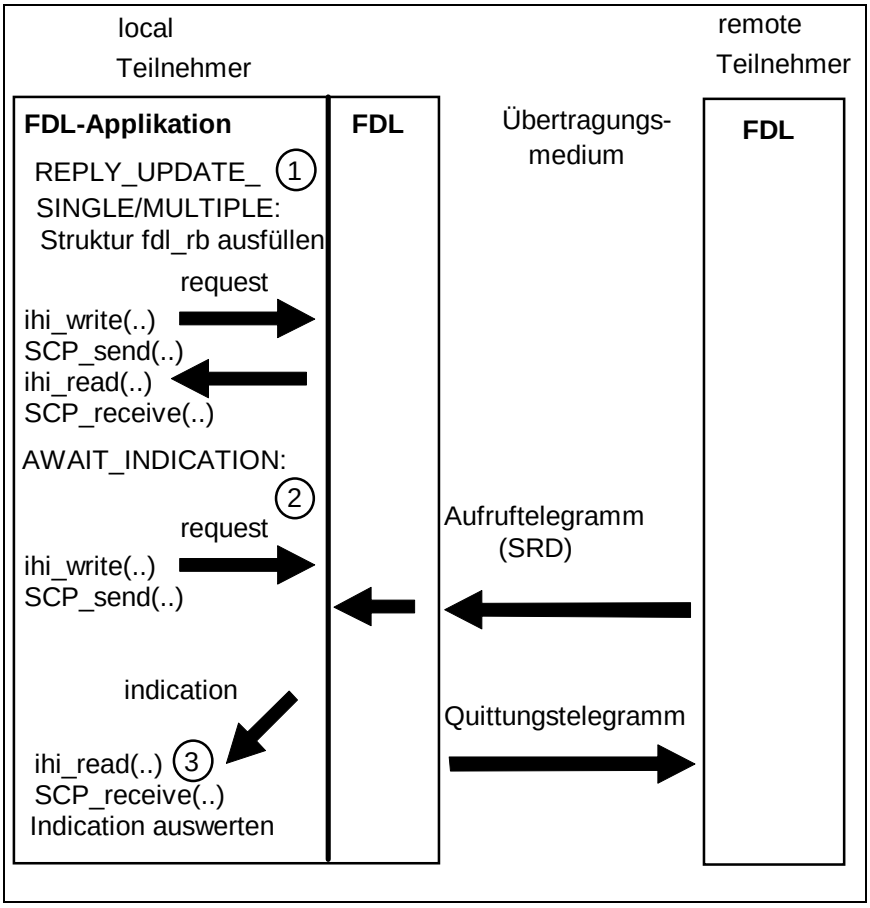


Bild 5-5: Empfang von Datentelegrammen SRD

Anmerkungen

① Mit dem REPLY_UPDATE_SINGLE oder REPLY_UPDATE_ MULTIPLE-Dienst werden Daten an die Schicht 2 übergeben, die ein remote Teilnehmer durch einen SRD-Dienst abholen kann. Die Daten werden im Quittungstelegramm an den remote Teilnehmer gesendet. Beim REPLY_UPDATE_SINGLE-Dienst können die Daten nur einmal, beim REPLY_UPDATE_ MULTIPLE-Dienst können die Daten durch mehrere SRD-Dienste ausgelesen werden.

☞ **Es muß darauf geachtet werden, daß im Sendepuffer ein Offset von mindestens 12 Byte auf die Nettodaten eingehalten wird. Im ersten Byte des Sendepuffers muß die Größe des Offsets eingetragen werden.**

Sollen im Quittungstelegramm keine Daten an den remote Teilnehmer gesendet werden, kann ① entfallen.

② Um ein Datentelegramm eines remote Teilnehmers empfangen zu können, muß an den entsprechenden SAP durch AWAIT_INDICATION eine oder mehrere Empfangsressourcen übergeben werden. Mehrere Empfangsressourcen können durch wiederholten Aufruf von AWAIT_INDICATION an den SAP übergeben werden. Nach dem Empfang eines Aufruftelegramms ist die entsprechende Ressource verbraucht und muß durch einen neuen AWAIT_INDICATION ersetzt werden.

③ Nach Empfang eines Aufruftelegramms generiert die Schicht 2 eine Indication mit den empfangenen Daten an die FDL-Applikation. Dabei enthält das erste Byte des Empfangspuffers den Offset auf die empfangenen Daten. Die Indication kann durch einen ihi_read()- oder SCP_receive()-Aufruf ausgelesen werden.

Notizen

6 Funktionsaufrufe der IHI-Schnittstelle

In diesem Kapitel werden die IHI-Schnittstellenfunktionen beschrieben, mit denen Sie die FDL-Aufträge übergeben und die Ergebnisse abholen können.

Unter MS-DOS und Windows 3.x sind nur diese Aufrufe der IHI-Schnittstelle zu benutzen.

Unter Windows 95/98 und Windows NT sind die IHI-Aufrufe nur für die Portierung von Altapplikationen vorgesehen.

Programmier-schnittstelle

Die FDL-Programmierschnittstelle wird der FDL-Applikation in Form einer Library zur Verfügung gestellt. Die Library-Funktionen der FDL-Programmierschnittstelle übernehmen für die FDL-Applikation die Übertragung der FDL-Aufträge zum CP.

Die FDL-Programmierschnittstelle umfaßt folgende Funktionen:

Anmeldefunktion für die FDL-Applikation	ihi_open_dev
Senden von Aufträgen, Daten	ihi_write
Empfangen von Daten (Aufträgen, Quittungen)	ihi_read
Abmeldefunktion für die FDL-Applikation	ihi_close

Schnittstellen-dynamik

Die Aufrufe der FDL-Programmierschnittstelle müssen durch die FDL-Applikation erfolgen. Dabei ist folgende Reihenfolge einzuhalten: Der erste Aufruf der Schnittstelle ist **ihi_open_dev**. Mit **ihi_write** kann die FDL-Applikation danach Aufträge an den CP abgeben. Jeder Auftrag muß mit **ihi_read** wieder abgeholt werden. Solange ein Requestblock nicht abgeholt wurde, darf er und die angehängten Datenpuffer nicht mehr verwendet werden. Am Ende wird die Verbindung zum CP mit **ihi_close** beendet.

6.1 ihi_open_dev

Beschreibung der Funktion

Mit der Funktion ihi_open_dev meldet sich die FDL-Applikation beim Treiber an. Der Treiber übergibt den Auftrag an den CP.

Über den Parameter "dev" wird der CP im PG/PC selektiert. Die Funktion liefert ein handle zurück, das bei allen weiteren Aufrufen angegeben werden muß.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     ihi_open_dev (ord16 mode, char * dev);
```

Beschreibung der Parameter

mode:	mode = 0:	Es wird kein Devicename vorgegeben, statt dessen wird über den ersten CP kommuniziert, der die ihi-Funktionsaufrufe unterstützt.
	mode = 1:	Es wird eine Verbindung zwischen der FDL-Applikation und dem über den Parameter "dev" gewählten CP hergestellt.
		Empfehlung: Für den Betrieb über die FDL-Programmierschnittstelle ist mode = 1 zu setzen.
dev:		Dieser Parameter selektiert den CP. Syntax: "/name:/FLC"
		name = identisch mit dem im Konfigurationsprogramm gewählten.

Rückgabewert

= 0:	**Erfolg:**	Rückgabewert = handle
< 0:	**Fehler:**	
	-1:	Bustreiber nicht installiert.
	-2:	Fehler bei Treiberöffnung.
	-3:	Treiber bereits geöffnet.

6.2 ihi_write

Beschreibung der Funktion

Mit der Funktion ihi_write werden Requestblöcke an den CP zur Bearbeitung übergeben.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     ihi_write (int handle, RB * rb);
```

Beschreibung der Parameter

handle:	Referenz (siehe ihi_open)
rb:	Adresse auf den zu übergebenden Requestblock.

Rückgabewert

= 0:	**Erfolg:**	Auftrag korrekt an den CP übergeben.
< 0:	**Fehler:**	
	-1:	Kein ihi_open_dev für dieses handle durchgeführt.
	-2:	Auftrag nicht mehr übergebbar. Maximale Zahl gleichzeitiger Aufträge überschritten.
	-3:	Kommt nicht mehr vor.
	-4:	Bedeutung wie bei dem Rückgabewert -2.
	-5:	Fehlerhafter Auftrag, d.h. der Auftrag wurde nicht an den CP weitergegeben.

6.3 ihi_read

Beschreibung der Funktion

Mit diesem Aufruf erhält die FDL-Applikation die vom CP bearbeiteten Requestblöcke zurück. Die Rückgabe erfolgt durch einen Pointer auf den Requestblock.

Die FDL-Applikation hat die Wahl zwischen einem synchronen Modus, bei dem der Aufruf erst durch den Erhalt eines Requestblocks beendet wird, und einem asynchronen Modus, der die Ergebnisabfrage durch Polling ermöglicht.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     ihi_read (int handle, ord16 mode,RB ** rb);
```

Beschreibung der Parameter

Handle:		Referenz (siehe ihi_open)
Mode:	mode=0	Asynchroner Modus, Polling. Die Funktion speichert die Adresse eines RB im Parameter rb, sofern der Rückgabewert 1 ist. Andernfalls wird die Funktion mit *rb = 0 beendet.
	mode=1	Synchroner Modus, warten auf Ergebnis. Der Aufruf wird erst beendet, wenn ein Requestblock durch den CP zurückgegeben wird.
rb:		Adresse eines Requestblockpointers, der vom CP zurückgegeben wurde.

Rückgabewert

= 0:	**Erfolg:**	Auftrag korrekt bearbeitet.
= 1:	**Erfolg:**	Auftrag korrekt ausgeführt. RB übergeben.
< 0:	**Fehler:**	
	-1:	Kein ihi_open_dev für dieses handle durchgeführt.
	-2:	Es sind keine Aufträge vorhanden.
	-3:	Unzulässige Empfangsart (falscher mode).
	-4:	Kommt nicht mehr vor.
	-5:	Kommt nicht mehr vor.

Synchroner Modus und Asynchroner Modus dürfen nicht gleichzeitig in einem Programm verwendet werden.

Unter Windows darf nur im asynchronen Modus gearbeitet werden.

6.4 ihi_close

Beschreibung der Funktion

Mit der Funktion ihi_close meldet sich eine FDL-Applikation beim Treiber ab. Nach diesem Aufruf ist keine Produktivkommunikation mit diesem handle mehr möglich.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     ihi_close (int handle);
```

Beschreibung der Parameter

handle: Referenz (siehe ihi_open).

Rückgabewert

= 0: **Erfolg:** Auftrag korrekt ausgeführt.
Unter Windows wird von ihi_close() der Wert 0 auch dann zurückgegeben, wenn Aufträge verworfen wurden.

< 0: **Fehler:**

-1: Kein ihi_open_dev für dieses handle durchgeführt.

-2: ihi_close wurde erfolgreich abgeschlossen, nicht bearbeitete Aufträge wurden verworfen.

6.5 Beispiele

Beispiel 1: Aufträge werden hintereinander an den CP gesendet.

```
#include "fdl_rb.h"

bsp_1 ()
{     int handle;
      RB   rb;    /* Requestblock, Auftragsblock */
      int ret;
      RB * rb_ptr;

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

      if (handle < 0)
      {
          /* Fehler beim Öffnen der Verbindung
             zum CP 5412 */
      }
      /* rb ausfüllen */
      ret = ihi_write (handle, &rb);

      if (ret >= 0 )
      {
           /* Abholen des Requestblockes */
           ret = ihi_read (handle, 1, &rb_ptr);
       }

      /* rb ausfüllen */
      /*(siehe Beispiele auf der Diskette) */
      ret = ihi_write (handle, &rb);

      if (ret >= 0 )
      {
           /* Abholen des Requestblockes */
           ret = ihi_read (handle, 1, &rb_ptr);
       }

       /* Beenden der Arbeit mit PROFIBUS */
       ret = ihi_close (handle);
}
```

Beispiel 2: Mehrere Aufträge werden gleichzeitig auf dem CP bearbeitet.

```
#include "fdl_rb.h"
bsp_2 ()
{     int handle;
      RB   rb1;    /* Requestblock, Auftragsblock */
      RB   rb2;    /* Requestblock, Auftragsblock */
      RB   rb3;    /* Requestblock, Auftragsblock */
      int ret;
      RB * rb_ptr;
      int i;

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

      if (handle < 0)
      {
          /* Fehler beim Öffnen der Verbindung
             zur CP5412 */
      }

      /* rb1 ausfüllen */

      /* rb1 senden an den CP */
      ret = ihi_write (handle, &rb1);

      /* rb2 ausfüllen */

      /* rb2 senden an den CP */
      ret = ihi_write (handle, &rb2);

      /* rb3 ausfüllen */

      /* rb3 senden an den CP */
      ret = ihi_write (handle, &rb3);
     /* Abholen der Requestblöcke */
     for (i = 0; i < 2; i++)
     {
         ret = ihi_read (handle, 1, &rb_ptr);
     }

     /* rb ausfüllen */

     ret = ihi_write (handle, &rb1);

     if (ret >= 0)
     {
         /* Abholen des Requestblockes */
         ret = ihi_read (handle, 1, &rb_ptr);
     }

     /* Beenden der Kommunikation */
     ret = ihi_close (handle);
}
```

7 Funktionsaufrufe der SCP-Schnittstelle

In diesem Kapitel werden die SCP-Schnittstellenfunktionen beschrieben, mit denen Sie die FDL-Aufträge übergeben und die Ergebnisse abholen können.

Unter MS-DOS und Windows 3.x sind diese Aufrufe für FDL-Aufträge nicht verfügbar.

Unter Windows 95/98 und Windows NT sind die SCP-Aufrufe für die Entwicklung von neuen FDL-Applikationen vorgesehen.

FDL-Programmier-schnittstelle

Die FDL-Programmierschnittstelle wird der FDL-Applikation in Form einer Library zur Verfügung gestellt. Die Library-Funktionen der FDL-Programmierschnittstelle übernehmen für die FDL-Applikation die Übertragung der FDL-Aufträge zum CP.

Die FDL-Programmierschnittstelle umfaßt folgende Funktionen:

Anmeldefunktion für die FDL-Applikation	SCP_open
Senden von Aufträgen, Daten	SCP_send
Empfangen von Daten (Aufträgen, Quittungen)	SCP_receive
Abmeldefunktion für die FDL-Applikation	SCP_close
Abholen von Fehlerkennungen	SCP_get_errno

Schnittstellen-dynamik

Die Aufrufe der FDL-Programmierschnittstelle müssen durch die FDL-Applikation erfolgen. Dabei ist folgende Reihenfolge einzuhalten: Der erste Aufruf der Schnittstelle ist **SCP_open**. Mit **SCP_send** kann die FDL-Applikation danach Aufträge an den CP abgeben. Mit **SCP_receive** werden die Quittungen und Daten des CPs abgeholt. Am Ende wird die Verbindung zum CP mit **SCP_close** beendet.

Nach jeder Funktion, welche den Wert -1 zurückliefert, kann durch Aufruf von **SCP_get_errno** eine Fehlerkennung zur genaueren Bestimmung der Fehlerursache aufgerufen werden.

Bitte beachten Sie auch die betriebssystemspezifischen Besonderheiten im Anhang.

7.1 SCP_open

Beschreibung der Funktion

Mit der Funktion SCP_open meldet sich die FDL-Applikation beim Treiber an. Der Treiber übergibt den Auftrag an den CP.

Über den Parameter "dev" wird der CP im PG/PC selektiert. Die Funktion liefert ein handle zurück, das bei allen weiteren Aufrufen angegeben werden muß.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     SCP_open (char * dev);
```

Beschreibung der Parameter

dev:	Dieser Parameter selektiert den CP. Syntax: "/name/FLC" name = identisch mit dem im Konfigurationsprogramm gewählten.

Rückgabewert

= 0:	**Erfolg:**	Rückgabewert = handle
= -1:	**Fehler:**	Die genaue Fehlerursache kann durch SCP_get_errno() ermittelt werden.

7.2 SCP_send

Beschreibung der Funktion

Mit der Funktion SCP_send werden Requestblöcke an den CP zur Bearbeitung übergeben.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     SCP_send (int handle, UWORD length, char * rb);
```

Beschreibung der Parameter

handle:	Referenz (siehe SCP_open)
length:	Länge des zu übergebenden Requestblocks in Bytes.
rb:	Adresse des zu übergebenden Requestblocks.

Rückgabewert

= 0:	**Erfolg:**	Auftrag korrekt an den CP übergeben.
= -1:	**Fehler:**	Die genaue Fehlerursache kann durch SCP_get_errno() ermittelt werden.

7.3 SCP_receive

Beschreibung der Funktion

Mit diesem Aufruf erhält die FDL-Applikation Auftragsquittungen und empfangene Daten vom CP zurück. Die Rückgabe erfolgt in einem von der Applikation bereitgestellten Puffer.

Die FDL-Applikation hat die Wahl zwischen einem synchronen Modus, bei dem der Aufruf erst durch den Erhalt eines Requestblocks beendet wird, und einem asynchronen Modus, der die Ergebnisabfrage durch Polling ermöglicht.

Deklaration der Funktion

```
#include "fdl_rb.h"

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

Beschreibung der Parameter

handle:		Referenz (siehe SCP_open)
timeout:		Wartezeit für den Empfangsauftrag. Folgende Werte sind möglich.
	0	Asynchroner Modus (SCP_NOWAIT): Die Funktion kehrt sofort zurück. Falls keine Daten für den Aufrufer vorliegen, ist *data_len = 0.
	FFFFh	Synchroner Modus (SCP_FOREVER): Der Aufruf wird erst beendet, wenn Daten für den Aufrufer eingetroffen sind.
	0 < timeout < FFFFh	Die Funktion kehrt beim Eintreffen von Daten für den Aufrufer, spätestens jedoch nach timeout Sekunden zurück.
data_len:		Zeiger auf Anzahl der Bytes, die empfangen wurden. (Rückgabeparameter)
length:		Länge des Empfangspuffers in Bytes.
buffer:		Adresse des Empfangspuffers.

Rückgabewert

= 0:	**Erfolg:**	Auftrag korrekt bearbeitet.
= -1:	**Fehler:**	Die genaue Fehlerursache kann durch SCP_get_errno() ermittelt werden.

☞ **Synchroner Modus und Asynchroner Modus dürfen nicht gleichzeitig in einem Programm verwendet werden.**

☞ **In Windows-Applikationen darf nur im asynchronen Modus gearbeitet werden.**

7.4 SCP_close

Beschreibung der Funktion

Mit der Funktion SCP_close meldet sich eine FDL-Applikation beim Treiber ab. Nach diesem Aufruf ist keine Produktivkommunikation mit diesem handle mehr möglich.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     SCP_close (int handle);
```

Beschreibung der Parameter

handle:		Referenz (siehe SCP_open).

Rückgabewert

= 0:	**Erfolg:**	Auftrag korrekt ausgeführt. Der Wert 0 wird auch dann zurückgegeben, wenn noch ausstehende Aufträge verworfen wurden.
= -1:	**Fehler:**	Die genaue Fehlerursache kann durch SCP_get_errno() ermittelt werden.

7.5 SCP_get_errno

Beschreibung der Funktion

Mit der Funktion SCP_get_errno kann eine FDL-Applikation die Ursache für den Fehler einer vorhergehenden SCP-Funktion ermitteln.

Deklaration der Funktion

```
#include "fdl_rb.h"
int     WINAPI SCP_get_errno (void);
```

Beschreibung der Parameter

keine

Rückgabewert

= 0:	Letzter Auftrag korrekt ausgeführt
= 202:	Ressourcenengpaß im Treiber oder in der Library
= 203:	Konfigurationsfehler
= 205:	Auftrag zur Zeit nicht erlaubt
= 206:	Parameterfehler
= 207:	Gerät bereits/noch nicht geöffnet.
= 208:	CP reagiert nicht
= 209:	Fehler in der Firmware
= 210:	Speicherengpaß im Treiber
= 215:	Keine Nachricht vorhanden
= 216:	Fehler bei Zugriff auf Anwendungspuffer
= 219:	Timeout abgelaufen
= 225:	Die maximale Anzahl an Anmeldungen ist überschritten
= 226:	Der Auftrag wurde abgebrochen
= 233:	Ein Hilfsprogramm konnte nicht gestartet werden
= 234:	Keine Autorisierung für diese Funktion vorhanden
= 304:	Initialisierung noch nicht abgeschlossen
= 305:	Funktion nicht implementiert
= 4865:	CP-Name nicht vorhanden
= 4866:	CP-Name nicht konfiguriert
= 4867:	Kanalname nicht vorhanden
= 4868:	Kanalname nicht konfiguriert

7.6 Beispiele

Beispiel 1: Aufträge werden hintereinander an den CP gesendet.

```
#include "fdl_rb.h"

bsp_1 ()
{     int    handle;
      fdl_rb rb;    /* Requestblock, Auftragsblock */
      int    ret;
      UWORD  data_len;

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

      if (handle == -1)
      {
           /* Fehler beim Öffnen der Verbindung
              zum CP */
      }
      /* rb ausfüllen */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      if (ret == 0)
      {
           /* Abholen der Quittung/Daten */
           ret = SCP_receive (handle, 0xffff,
                              &data_len,
                              sizeof(fdl_rb), &rb);
      }

      /* rb ausfüllen */
      /*(siehe Beispiele auf der Diskette) */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      if (ret == 0)
      {
           /* Abholen der Quittung/Daten */
           ret = SCP_receive (handle, 0xffff,
                              &data_len,
                              sizeof(fdl_rb), &rb);
      }

      /* Beenden der Arbeit mit FDL */
      ret = SCP_close (handle);
}
```

Beispiel 2: Mehrere Aufträge werden gleichzeitig auf dem CP bearbeitet.

```
#include "fdl_rb.h"
bsp_2 ()
{     int    handle;
      fdl_rb rb;   /* Requestblock, Auftragsblock */
      int    ret;
      UWORD  data_len;
      int    i;

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

      if (handle == -1)
      {
          /* Fehler beim Öffnen der Verbindung
             zur CP */
      }

      /* rb mit 1. Auftrag ausfüllen */

      /* rb an den CP senden */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* rb mit 2. Auftrag ausfüllen */

      /* rb an den CP senden */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* rb mit 3. Auftrag ausfüllen */

      /* rb an den CP senden */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* Abholen der Requestblöcke */
      for (i = 0; i < 2; i++)
      {
         ret = SCP_receive (handle, 0xffff,
                            &data_len,
                            sizeof(fdl_rb), &rb);
      }

      /* rb mit 4. Auftrag ausfüllen */

      /* rb an den CP senden */
      ret = SCP_send (handle, sizeof(fdl_rb), &rb);

      /* Abholen der Requestblöcke */
      if (ret != -1)
      {
         ret = SCP_receive (handle, 0xffff,
                            &data_len,
                            sizeof(fdl_rb), &rb);
      }

      /* Beenden der Kommunikation */
      ret = SCP_close (handle);
}
```

8 Anhang

8.1 Übersetzen und Binden für Windows 95/98

8.1.1 Arbeiten mit dem MSVC-Compiler von Microsoft

Hinweis

Unter Windows 95/98 wird die SCP-Schnittstelle durch eine DLL zur Verfügung gestellt. Die Import-Library für den MSVC-Compiler 2.2 für Windows 95/98 ist s7onlinx.lib. Sie finden sie im Verzeichnis ..\ fdl.w95\Lib oder der in der Installationsanleitung angegebenen Stelle.

Die Sourcen und Makefiles zum Übersetzen des Beispielprogramms FDLdemo.c befinden sich im Verzeichnis ..\Fdl.w95\Sample\Fdl.

8.2 Übersetzen und Binden für Windows NT

8.2.1 Arbeiten mit dem MSVC-Compiler von Microsoft

Hinweis

Unter Windows NT wird die SCP-Schnittstelle durch eine DLL zur Verfügung gestellt. Die Import-Library für den MSVC-Compiler 2.2 für Windows NT ist s7onlinx.lib. Sie finden sie im Verzeichnis ..\ Fdl.nt\Lib oder der in der Installationsanleitung angegebenen Stelle.

Die Sourcen und Makefiles zum Übersetzen des Beispielprogramms FDLdemo.c befinden sich im Verzeichnis ..\Fdl.nt\Sample\Fdl.

8.3 Besonderheiten für Windows

Windows-Programme unterscheiden sich von Konsole-Programmen u. a. dadurch, daß sie in einer WndProc verzweigen. An einer zentralen Stelle warten Windows-Programme auf Windows-Meldungen, die danach in einer WndProc-Prozedur abgearbeitet werden. Es kann vorkommen, daß bei der Abarbeitung der WndProc die Kontrolle an Windows übergeben wird, und WndProc noch einmal aufgerufen wird.

In einem Windowsprogramm müssen Sie nach SCP_open() die Routine SetSinecHWnd mit einem Windowhandle aufrufen, damit der Treiber weiß, wohin er seine Meldungen zu schicken hat. Wird ein asynchroner Befehl abgegeben, wird beim Empfang der Nachricht eine WM_SINEC-Message an Windows gesendet. In der dazugehörigen WndProc kann es dann verarbeitet werden, indem Sie ein SCP_receive mit timeout 0 durchführen.

Beispiel einer typischen Windowsapplikation :

```
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  .... /* Anstoss der Funktion */:
            ret = SCP_send (handle,....);
            break;

      case WM_SINEC:
            ret = SCP_receive (handle, 0, &rb_ptr);
            if (ret != -1)
            {
               /* ein Requestblock kam zurück==>  */
               /* Analysiere ihn.                 */
            }
            break;
   }
}
```

Hinweis

Aufrufformat von SetSinecHWnd:

SetSinecHWnd (int handle, HWND hWnd)

Die zu SetSinecHWnd alternative Funktion

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

erlaubt es der FDL-Applikation, sich durch eine selbst definierte Nachricht (msg_id) vom Treiber beim Eintreffen von Daten benachrichtigen zu lassen.

9 Index

Notizen

Glossar

Anlage Gesamtheit aller elektrischen Betriebsmittel. Zu einer Anlage gehören u. a.: Speicherprogrammierbare Steuerung, Geräte für Bedienen und Beobachten, Bussysteme, Feldgeräte, Antriebe, Versorgungsleitungen.

Ansprechüber-wachungszeit Eine im -> DP-Slave einstellbare Überwachungszeit zur Ausfallerkennung des zugeordneten -> DP-Masters.

Basisadresse Logische Adresse einer Baugruppe in S7-Systemen.

Baudrate Übertragungsrate am Bus (Einheit: Bit/sec). Ein -> Busparameter für -> PROFIBUS. Die Angabe bzw. Wahl der Baudrate hängt von verschiedenen Randbedingungen ab, wie beispielsweise Entfernung.

Busparameter Busparameter steuern das Übertragungsverhalten am Bus. Jeder -> Teilnehmer an -> PROFIBUS muß mit den Busparametern anderer Teilnehmer übereinstimmende Busparameter verwenden.

Bussegment Teil eines -> Subnetzes. Subnetze können aus Bussegmenten mittels Segmentübergängen wie Repeater und Bridges gebildet sein. Segmente sind für die Adressierung transparent.

CFB Communication Function Block: Kommunikationsverfahren zum programmgesteuerten Übertragen von Daten von oder zu einer CPU in S7-300/400 durch Verwendung von speziellen Funktionsbausteinen. Diese Funktionsbausteine wurden basierend auf dem Entwurf von IEC 1131-5 definiert. Kommunikationspartner können andere kommunikationsfähige Baugruppen aus S7-300/400, BuB-Systeme, PC oder andere Steuerungen und Rechner sein.

CP Communication Processor. Baugruppe für Kommunikationsaufgaben.

Dezentrale Peripherie Ein- und Ausgabebaugruppen, die dezentral von der CPU (Zentraleinheit der Steuerung) eingesetzt werden. Die Verbindung zwischen dem Automatisierungsgerät und der Dezentralen Peripherie erfolgt über das Bussystem -> PROFIBUS. Automatisierungsgeräten wird der Unterschied zu lokalen Prozeßein- oder Prozeßausgaben verdeckt.

Dienste Angebotene Leistungen eines Kommunikationsprotokolls.

DP-E/A-Modul DP-Slaves sind modular aufgebaut. Ein -> DP-Slave besitzt mindestens ein DP-E/A-Modul.

DP-E/A-Typ DP-E/A-Typ bezeichnet ein -> DP-E/A-Modul. Zu unterscheiden sind:
- Eingabemodul
- Ausgabemodul
- Ein-/Ausgabemodul

DP-Master Ein -> Teilnehmer mit Masterfunktion bei -> PROFIBUS-DP. Der DP-Master wickelt den Nutzdatenverkehr mit den ihm zugeordneten -> DP-Slaves ab.

DP-Modulliste In der DP-Modulliste werden die zu einem -> DP-Slave gehörenden Module verwaltet. Einträge in die DP-Modulliste werden bei der Projektierung eines -> DP-Masters mit dem -> COM PROFIBUS vorgenommen.

DP-Modulname Bezeichnung eines in der ->DP-Modulliste eingetragenen -> DP-E/A-Moduls.

DP-Modultyp Bezeichnung für die Identifikation eines -> DP-E/A-Moduls in den -> Gerätestammdaten eines -> DP-Slaves nach nach EN 50 170 Vol. 2.

DP-Slave Ein -> Teilnehmer mit Slavefunktion bei -> PROFIBUS-DP.

DP-Slavekatalog Im DP-Slavekatalog werden für die Projektierung von -> DP-Mastern Gerätebeschreibungen für -> DP-Slaves nach -> DP-Norm hinterlegt. Der DP-Slavekatalog steht bei der Projektierung mit dem -> COM PROFIBUS zur Verfügung.

DP-Slave-Name Zur Identifikation eines -> DP-Slaves in der projektierten DP-Konfiguration wird ein DP-Slave-Name in der DP-Slaveliste eingetragen.

DP-Subnetz PROFIBUS-(Sub)netz, an dem nur -> Dezentrale Peripherie betrieben wird.

DP-Subsystem Ein -> DP-Master und alle -> DP-Slaves, mit denen dieser DP-Master Daten austauscht.

Enhanced Mode Erweiterte Betriebsart unter Windows 3.x für Personal Computer mit einem Intel 386 oder einem kompatiblen Prozessor.

FDL Fieldbus Data Link. Schicht 2 bei -> PROFIBUS.

FREEZE-Modus Der FREEZE-Modus ist eine DP-Betriebsart, bei der von allen (oder von einer Gruppe von) DP-Slaves zeitgleich Prozeßdaten erfaßt werden. Der Erfassungszeitpunkt wird durch das FREEZE-Kommando (das ist ein Steuertelegramm zur Synchronisation) signalisiert.

Gap-Aktualisierungs-faktor Ein freier Adreßbereich zwischen zwei aktiven -> Teilnehmern wird zyklisch von dem Teilnehmer mit der kleineren -> PROFIBUS-Adresse durchsucht um festzustellen, ob ein weiterer Teilnehmer in den logischen Ring aufgenommen werden möchte. Die Zykluszeit für diese Überprüfung wird bestimmt durch:

Gap-Aktualisierungsfaktor x Target rotation time

Gateway Intelligentes Schnittstellengerät, das auf ISO-Schicht 7 verschiedenartige lokale -> Netze miteinander verbindet.

GD-Paket Zusammenfassung von ggf. im Automatisierungssystem verteilt abgelegten Daten (z. B. Merker, Datenbaustein), die mit dem Verfahren -> Globale Daten übertragen werden sollen.

GD-Kreis GD-Kreis bezeichnet eine Zusammenfassung von -> Teilnehmern, die miteinander GD-Daten austauschen. Ein -> GD-Paket wird somit an die zum GD-Kreis gehörenden Teilnehmer gesendet.

Gerätestamm-daten Gerätestammdaten (GSD) enthalten DP-Slave-Beschreibungen nach DIN E 19245 Teil 3. Die Nutzung von GSD erleichtert die Projektierung des -> DP-Masters sowie der -> DP-Slaves.

Globale Daten Globale Daten (GD) bezeichnet ein Kommunikationsverfahren zum zyklischen Austausch von begrenzten Datenmengen aus STEP 7-Datenbereichen zwischen CPUs der S7-300/400. Gesendete Daten können gleichzeitig von mehreren CPUs empfangen werden.

Globale Peripherie Ein Teil des Peripheriebereiches bei SIMATIC S5-AGs kann für den globalen Datenaustausch zwischen SIMATIC S5-AGs über -> PROFIBUS genutzt werden. Ein wesentliches Merkmal des verwendeten Übertragungsverfahrens ist der zyklische Austausch der Daten, die sich gegenüber dem jeweils vorhergehenden Zyklus geändert haben.

Gruppen-identifikation DP-Slaves können über eine Gruppenidentifikation einer oder mehreren Gruppen zugewiesen werden. Die -> DP-Slaves können dann über die Gruppenidentifikation bei der Übertragung von Steuertelegrammen gezielt angesprochen werden.

Höchste PROFIBUS-Adresse Ein -> Busparameter für -> PROFIBUS. Gibt die höchste -> PROFIBUS-Adresse eines aktiven -> Teilnehmers an PROFIBUS an. Für passive Teilnehmer sind PROFIBUS-Adressen größer als HSA zulässig (Wertebereich: HSA 1..126).

Knotentabelle Die Knotentabelle gilt für alle -> Netze innerhalb einer -> Anlage. Jeder Eintrag in der Knotentabelle beschreibt die Schnittstelle eines Automatisierungssystems (oder einer beliebigen anderen Station) zu einem -> Subnetz. Die Einträge in der Knotentabelle werden vom System benutzt, um Verbindungen zwischen Stationen zu finden und aufzubauen.

PROFIBUS-Adresse Die PROFIBUS-Adresse ist eine eindeutige Kennung eines an -> PROFIBUS angeschlossenen -> Teilnehmers. Zur Adressierung eines Teilnehmers wird die PROFIBUS-Adresse im -> Telegramm übertragen.

Master Aktiver Teilnehmer an -> PROFIBUS, der unaufgefordert -> Telegramme senden kann, wenn er im Besitz des Token ist.

Maximum Station Delay

Ein -> Busparameter für -> PROFIBUS. Die Maximum Station Delay (max. TSDR) gibt die größte, bei einem der -> Teilnehmer im -> Subnetz benötigte Zeitspanne an, die zwischen dem Empfang des letzten Bits eines unquittierten -> Telegramms bis zum Senden des ersten Bits des nächsten Telegramms vergehen muß. Ein Sender darf nach dem Senden eines unquittierten Telegrammes erst nach Ablauf der Zeitspanne max. TSDR ein weiteres Telegramm senden.

Minimum Station Delay

Ein -> Busparameter für -> PROFIBUS. Die Minimum Station Delay (min. TSDR) gibt die Zeitspanne an, die der Empfänger eines -> Telegramms bis zum Senden der Quittung oder bis zum Senden eines weiteren Telegrammes mindestens warten muß. Die min. TSDR richtet sich nach der größten, bei einem Teilnehmer im Subsystem benötigten Zeitspanne zur Entgegennahme einer Quittung nach dem Senden des Telegrammes.

Netz

Ein Netz besteht aus einem oder mehreren verknüpften -> Subnetzen mit einer beliebigen Zahl von -> Teilnehmern. Es können mehrere Netze nebeneinander bestehen. Für jedes Subnetz gibt es eine gemeinsame -> Knotentabelle.

Offset

Bei der FDL-Programmierschnittstelle Länge des reservierten Bereichs am Beginn eines Datenpuffers.

PROFIBUS

Ein Feldbus nach EN 50 170 Vol. 2.

PROFIBUS DP

DP-Betriebsart nach EN 50 170 Vol. 2.

PROFIBUS PA

PROFIBUS PA ist eine Richtlinie der PROFIBUS Nutzerorganisation (PNO), die PROFIBUS nach EN 50 170 Vol. 2 um den Einsatz im eigensicheren Bereich ergänzt.

Protokoll

Verfahrensvorschrift für die Übermittlung in der Datenübertragung. Mit dieser Vorschrift werden sowohl die Formate der Nachrichten als auch der Datenfluß bei der Datenübertragung festgelegt.

Prozeßabbild

Das Prozeßabbild ist ein besonderer Speicherbereich im Automatisierungssystem. Am Anfang des zyklischen Programmes werden die Signalzustände der Eingabebaugruppen zum Prozeßabbild der Eingänge übertragen. Am Ende des zyklischen Programmes wird das Prozeßabbild der Ausgänge als Signalzustand zu den Ausgabebaugruppen übertragen.

Reorganisation Tokenring

Alle -> Master am -> PROFIBUS bilden einen logischen Tokenring. Innerhalb dieses Tokenrings wird die Sendeberechtigung (Token) von Station zu Station weitergegeben. Wird nun die Übertragung des Tokens gestört oder wird ein Master vom Tokenring entfernt, so führt dies bei der Tokenweitergabe zu einem Fehler (Token wird von dieser Station nicht angenommen), was eine Ausgliederung dieser Station aus dem Tokenring zur Folge hat. Die Anzahl der Ausgliederungen werden im internen Token_error_counter gezählt. Erreicht dieser Zähler einen oberen Grenzwert, dann wird der logische Tokenring neu aufgebaut (reorganisiert).

SCOPE PROFIBUS Diagnoseprodukt für -> PROFIBUS, mit dem der Telegrammverkehr am -> Netz erfaßt und analysiert werden kann.

Segment Synonym für -> Bussegment.

Setup Time Ein -> Busparameter für -> PROFIBUS. Die Setup Time gibt den Mindestzeitabstand zwischen dem Empfang einer Quittung bis zum Senden eines neuen Aufruftelegrammes durch den Sender an.

SIMATIC NET Siemens Network and Communication. Produktbezeichnung für -> Netze und Netzkomponenten bei Siemens. (frühere Bezeichnung: SINEC)

PROFIBUS SIMATIC NET Bussystem für den Industrieeinsatz auf PROFIBUS-Basis.

PROFIBUS-DP PROFIBUS-Dezentrale Peripherie.

PROFIBUS-DP Master Ein -> Teilnehmer mit Masterfunktion bei -> PROFIBUS-DP.

PROFIBUS-FMS PROFIBUS-Fieldbus Message Specification. Obere Teilschicht von Schicht 7 des ISO/OSI-Referenzmodells bei PROFIBUS.

Slot Time Ein Busparameter für -> PROFIBUS. Die Slot Time (TSL) ist die Überwachungszeit eines Senders eines -> Telegramms auf die Quittung des Empfängers.

Subnetz Ein Subnetz ist ein Teil eines -> Netzes, dessen -> Busparameter (z. B. -> PROFIBUS-Adressen) abgeglichen werden müssen. Es umfaßt die Buskomponenten und alle angeschlossenen Stationen. Subnetze können beispielsweise mittels -> Gateways zu einem Netz gekoppelt werden.
Eine -> Anlage besteht aus mehreren Subnetzen mit eindeutigen -> Subnetznummern. Ein Subnetz besteht aus mehreren ->Teilnehmern mit eindeutigen -> PROFIBUS-Adressen.

Subnetznummer Eine -> Anlage besteht aus mehreren -> Subnetzen mit eindeutigen Subnetznummern.

SYNC-Modus Der SYNC-Modus ist eine DP-Betriebsart, bei der mehrere oder alle -> DP-Slaves zu einem bestimmten Zeitpunkt Daten an ihre Prozeßausgänge übergeben. Der Übergabezeitpunkt wird durch das SYNC-Kommando (das ist ein Steuertelegramm zur Synchronisation) signalisiert.

Target rotation time Ein -> Busparameter für -> PROFIBUS. Der Token ist die Sendeberechtigung für einen -> Teilnehmer an PROFIBUS. Ein Teilnehmer vergleicht eine von ihm gemessene Token-Umlaufzeit mit der Target rotation time und steuert davon abhängig das Senden hoch- und niederpriorer Telegramme.

Teilnehmer Ein Teilnehmer wird durch eine -> PROFIBUS-Adresse an -> PROFIBUS identifiziert.

Telegramm Nachricht eines PROFIBUS-Teilnehmers an einen anderen.

Telegramm-header Ein Telegrammheader besteht aus einer Kennung des -> Telegramms sowie der Quell- und Zielteilnehmeradresse.

Telegrammtrailer Der Telegrammtrailer besteht aus einer Prüfsumme und der Endekennung des -> Telegramms.

Treiber Software, die dem Datenaustausch von Applikationen mit dem -> CP dient.

Watchdog Mechanismus zur Überwachung der Betriebsbereitschaft.

❑