Technical Document

GSM PROTOCOL STACK

G23

DTI - DATA TRANSMISSION INTERFACE

INTRODUCTION

Document Number:	8448.202.01.002
Version:	0.3
Status:	Draft
Approval Authority:	
Creation Date:	2001-Oct-16
Last changed:	2015-Mar-08 by XGUTTEFE
File Name:	dti.doc

Important Notice

Texas Instruments Incorporated and/or its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products, software and services at any time and to discontinue any product, software or service without notice. Customers should obtain the latest relevant information during product design and before placing orders and should verify that such information is current and complete.

All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment. TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI products, software and/or services. To minimize the risks associated with customer products and applications, customers should provide adequate design, testing and operating safeguards.

Any access to and/or use of TI software described in this document is subject to Customers entering into formal license agreements and payment of associated license fees. TI software may solely be used and/or copied subject to and strictly in accordance with all the terms of such license agreements.

Customer acknowledges and agrees that TI products and/or software may be based on or implement industry recognized standards and that certain third parties may claim intellectual property rights therein. The supply of products and/or the licensing of software does not convey a license from TI to any third party intellectual property rights and TI expressly disclaims liability for infringement of third party intellectual property rights.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products, software or services are used.

Information published by TI regarding third–party products, software or services does not constitute a license from TI to use such products, software or services or a warranty, endorsement thereof or statement regarding their availability. Use of such information, products, software or services may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Change History

Date	Changed by	Approved by	Version	Status	Notes
2001-Oct-16	MG		0.1		1
2002-Mar-05	STW		0.2		2
2003-May-13	XGUTTEFE		0.3	Draft	

Notes:

1. Initial version
2. Additions and corrections according to DTI1

Table of Contents

List of Figures and Tables

List of References

[ISO 9000:2000] International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000

1.1 References

[1] Rec. T.4 Standardisation of group 3 facsimile apparatus for document transmission;
(CCITT-T.4, 1984)

[2] ITU-T Recommendation T.30; Series T: Terminal equipments and protocols for telematic services;
Procedures for document facsimile transmission in the general switched
telephone network;
(ITU-T.30, 1996)

[3] ITU-T Recommendation T.31; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 1
(ITU-T.31, 1995)

[4] ITU-T Recommendation T.32; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 2
(ITU-T.32, 1995)

[5] Rec. T.35; Terminal equipment and protocols for telematic services;
Procedures for the allocation of CCITT definde codes for non-standard facilities;
(CCITT-T.35, 1991)

[6] ITU-T Recommendation V.25 ter; Series V: data communication over the telephone network;
Interfaces and voiceband modems; Serial asynchronous automatic dialling and control
(ITU-T V.25 ter, 1997)

[7] Rec. V.42 bis Data compression procedures for data circuit terminating equipment (DCE) using error correction procedures;
(CCITT-V.42 bis, 1990)

[8] Rec. V.110 (Blue book, Vol. VIII, Fascicle VIII.1) Support of data terminal equipments (DTEs) with V-series type interfaces by an integrated services digital network (ISDN);
(CCITT-V.110, 1988)

[9] European digital cellular telecommunications system (Phase 2);
GSM Public Land Mobile Network (PLMN) connection types;
(GSM 3.10, September 1994, version 4.3.1)

[10] European digital cellular telecommunications system (Phase 2);
Technical realisation of facsimile group 3 transparent;
(GSM 3.45, September 1995, version 4.5.0)

[11] Digital cellular telecommunications system (Phase 2);
Mobile radio interface layer 3 specification;
(GSM 4.08, November 1996, version 4.17.0)

[12] European digital cellular telecommunications system (Phase 2);
Rate adaptation on the Mobile Station - Base Station System (MS - BSS) Interface;
(GSM 4.21, May 1995, version 4.6.0)

[13] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre (BSS - MSC) interface
(GSM 4.22, September 1994, version 4.3.0)

[14] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre (BSS - MSC) interface
(Amendment prA1 for GSM 4.22, version 4.3.0)
(GSM 4.22, March 1995, version 4.4.0)

[15] European digital cellular telecommunications system (Phase 2);
General on Terminal Adaptation Functions (TAF) for Mobile Stations (MS);
(GSM 7.01, December 1995, version 4.10.0)

[16] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using asynchronous bearer capabilities;
(GSM 7.02, September 1994, version 4.5.1)

[17] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using synchronous bearer capabilities;
(GSM 7.03, September 1994, version 4.5.1)

[18] Digital cellular telecommunications system (Phase 2);
Use of Data Terminal Equipment - Data Circuit terminating Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Services (CBS);
(GSM 7.05, November 1996, version 4.8.0)

[19] Digital cellular telecommunications system (Phase 2);
AT command set for GSM Mobile Equipment (ME)
(GSM 7.07, May 1996, version 4.1.0)

[20] Digital cellular telecommunication system (Phase 2);
Mobile Station (MS) conformance specification;
Part 1: Conformance specification
(GSM 11.10-1, November 1996, version 4.17.0)

[21] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 2: Protocol Implementation Conformance Statement (PICS)
proforma specification
(GSM 11.10-2, May 1996, version 4.15.0)

[22] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 3: Layer 3 (L3) Abstract Test Suite (ATS)
(GSM 11.10-3, November 1996, version 4.17.0)

[23] Proposal for Rate Adaptation implemented on a DSP;
(C. Bianconi, Texas Instruments, January 1998, version 1.0)

[24] MCU-DSP Interfaces for Data Applications;
Specification S844
(C. Bianconi, Texas Instruments, March 1998, version 0.1)

[25] Users Guide
6147.300.96.100; Condat GmbH

[26] Service Access Point RA
8411.100.98.100; Condat GmbH

[27] Service Access Point RLP
8411.101.98.100; Condat GmbH

[28] Service Access Point L2R
8411.102.98.100; Condat GmbH

[29] Service Access Point FAD
8411.103.98.100; Condat GmbH

[30] Service Access Point T30
8411.104.98.100; Condat GmbH

[31] Service Access Point ACI
8411.105.98.100; Condat GmbH

[32] Message Sequence Charts RLP
8411.201.98.100; Condat GmbH

[33] Message Sequence Charts L2R
8411.202.98.100; Condat GmbH

[34] Message Sequence Charts FAD
8411.203.98.100; Condat GmbH

[35] Message Sequence Charts T30
8411.204.98.100; Condat GmbH

[36] Message Sequence Charts ACI
8411.205.98.100; Condat GmbH

[37] Proposal for Fax & Data Integration; March 1998
8411.300.98.100; Condat GmbH

[38] Test Specification RLP
8411.401.98.100; Condat GmbH

[39] Test Specification L2R
8411.402.98.100; Condat GmbH

[40] Test Specification FAD
8411.403.98.100; Condat GmbH
[41] Test Specification T30
8411.404.98.100; Condat GmbH
[42] Test Specification ACI
8411.405.98.100; Condat GmbH
[43] SDL Specification RLP
8411.501.98.100; Condat GmbH
[44] SDL Specification L2R
8411.502.98.100; Condat GmbH
[45] SDL Specification FAD
8411.503.98.100; Condat GmbH
[46] SDL Specification T30
8411.504.98.100; Condat GmbH
[47] SDL Specification ACI
8411.505.98.100; Condat GmbH
[48] Technical Documentation RLP
8411.701.98.100; Condat GmbH
[49] Technical Documentation L2R
8411.702.98.100; Condat GmbH
[50] Technical Documentation FAD
8411.703.98.100; Condat GmbH
[51] Technical Documentation T30
8411.704.98.100; Condat GmbH
[52] Technical Documentation ACI
8411.705.98.100; Condat GmbH
[53] Technical Documentation DTI-SAP
8411.110.00.010
[53] Detailed Specification DTILIB
8448.201.01.010

1.2 Abbreviations

ACI	AT Command Interpreter
AGCH	Access Grant Channel
AT	Attention sequence "AT" to indicate valid commands of the ACI
BCCH	Broadcast Control Channel
BCS	Binary Coded Signals
BS	Base Station
BSIC	Base Station Identification Code
C/R	Command/Response
C1	Path Loss Criterion
C2	Reselection Criterion
CBCH	Cell Broadcast Channel
CBQ	Cell Bar Qualify
CC	Call Control
CCCH	Common Control Channel
CCD	Condat Coder Decoder
CKSN	Ciphering Key Sequence Number
CRC	Cyclic Redundancy Check
DCCH	Dedicated Control Channel
DISC	Disconnect Frame
DL	Data Link Layer
DM	Disconnected Mode Frame
DTX	Discontinuous Transmission
EA	Extension Bit Address Field
EL	Extension Bit Length Field
EMMI	Electrical Man Machine Interface
EOL	End Of Line
F	Final Bit
F&D	Fax and Data Protocol Stack
FACCH	Fast Associated Control Channel
FHO	Forced Handover
GP	Guard Period
GSM	Global System for Mobile Communication
HDLC	High level Data Link Control
HISR	High level Interrupt Service Routine
HPLMN	Home Public Land Mobile Network
I	Information Frame
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
ITU	International Telecommunication Union
IWF	Interworking Function
Kc	Authentication Key
L	Length Indicator
LAI	Location Area Information
LISR	Low level Interrupt Service Routine
LPD	Link Protocol Discriminator
M	More Data Bit
MCC	Mobile Country Code
MM	Mobility Management
MMI	Man Machine Interface
MNC	Mobile Network Code

MS	Mobile Station
MSG	Message phase in the GSM 3.45 protocol
N(R)	Receive Number
N(S)	Send Number
NCC	National Colour Code
NECI	New Establishment Causes included
OTD	Observed Time Difference
P	Poll Bit
P/F	Poll/Final Bit
PCH	Paging Channel
PCO	Point of Control and Observation
PDU	Protocol Description Unit
PL	Physical Layer
PLMN	Public Land Mobile Network
RACH	Random Access Channel
REJ	Reject Frame
RNR	Receive Not Ready Frame
RR	Radio Resource Management
RR	Receive Ready Frame
RTD	Real Time Difference
RTOS	Real Time Operating System
SABM	Set Asynchronous Balanced Mode
SACCH	Slow Associated Control Channel
SAP	Service Access Point
SAPI	Service Access Point Identifier
SDCCH	Slow Dedicated Control Channel
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSCB	Short Message Service Cell Broadcast
SS	Supplementary Services
T.4	CCITT Standardisation for Document coding of Group 3 Facsimile Apparatus
TAP	Test Application Program
TCH	Traffic Channel
TCH/F	Traffic Channel Full Rate
TCH/H	Traffic Channel Half Rate
TDMA	Time Division Multiple Access
TE	Terminal Equipment - e. g. a PC
TMSI	Temporary Mobile Subscriber Identity
UA	Unnumbered Acknowledgement Frame
UI	Unnumbered Information Frame
V(A)	Acknowledgement State Variable
V(R)	Receive State Variable
V(S)	Send State Variable
VPLMN	Visiting Public Land Mobile Network
DTI	Data Transmission Interface
DTILIB	Data Transmission Interface Library
DESC	Generic Data Descriptor
DESC-LIST	Linked List of Generic Data Descriptors

1.3 Terms

---

2 The DTI SAP

The DTI (Data Transmission Interface) is a generic SAP, which is used to transfer user data between two entities. It provides a simple and generic communication mechanism with a set of primitives for exchanging data. This SAP is used in all entities, which transfer user data (payload).

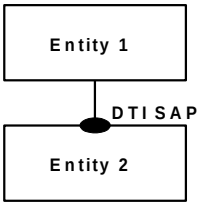

Figure 1: The DTI SAP

The data primitives of the DTI SAP are based on linked lists of data blocks, so that data packets of up to 64 kByte can be transferred without wasting memory for smaller packet sizes.

There are two different versions of the DTI SAP: DTI1 and DTI2. DTI1 is the original version, DTI2 is improved.

3 Motivation for the DTI SAP

DTI has been invented to give the protocol stack the flexibility for new configurations. Since the same interface is used in all entities, which transfer user data, it is possible to connect entities together in a flexible way.

For WAP over Circuit Switched Data for example, IP must be connected to PPP. For WAP over GPRS however, IP must be connected to SNDCP, which is handling IP packets directly without the help of PPP. This is only possible, if IP, PPP and SNDCP use the same interface.

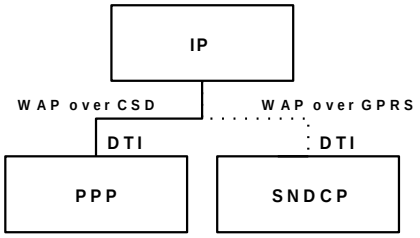

Figure 2: IP connecting to PPP or SNDCP

This flexibility of connecting entities to each other is required for the different protocol stack configurations, which have come along with new technologies like

- GPRS
- UMTS
- WAP
- Bluetooth

Figure 3 shows some of the possible protocol stack configurations. Since the entities UART, L2R, T30, Bluetooth, PPP, SNDCP (GPRS), TRA (transparent CSD), IP, UDP and WAP use the DTI SAP, they can all communicate with each other.

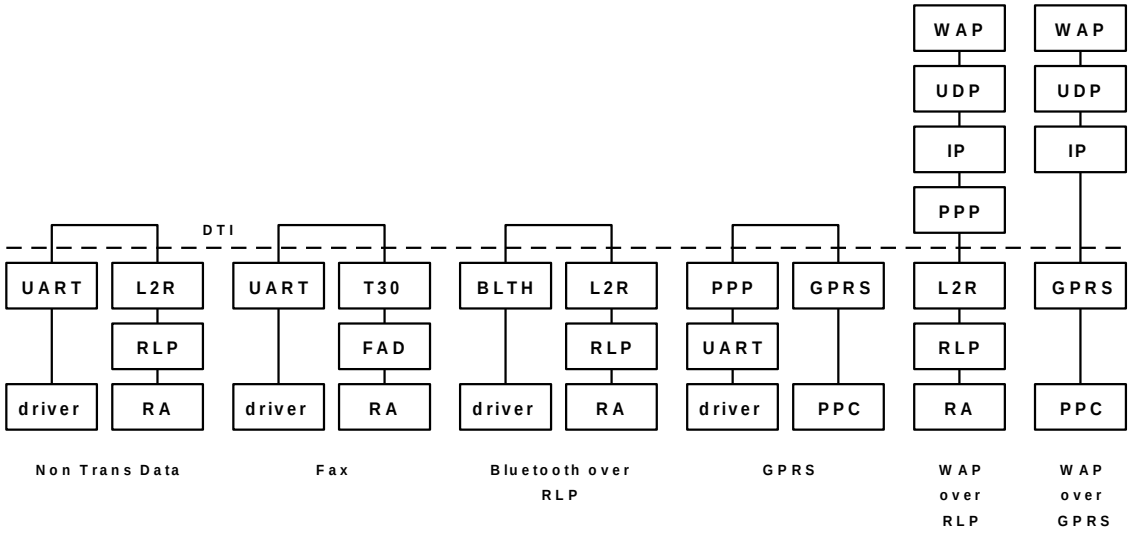

Figure 3: Protocol Stack Configurations

The DTI SAP is also used between WAP-UDP-IP-PPP and between PPP-UART.

4 DTI primitives

4.1 The set of primitives

As an universal and generic SAP the DTI SAP has only a small number of primitives. There is a data primitive and a flow control primitive for each direction. Moreover there are primitives for connecting and disconnecting in the version 2 of the DTI SAP.

The data and flow control primitives are:

Primitive	Meaning
DTI_DATA_REQ	uplink data primitive
DTI_DATA_IND	downlink data primitive
DTI_READY_IND	uplink flow control primitive
DTI_GETDATA_REQ	downlink flow control primitive

Figure 4 shows how the data and flow control primitives are used.

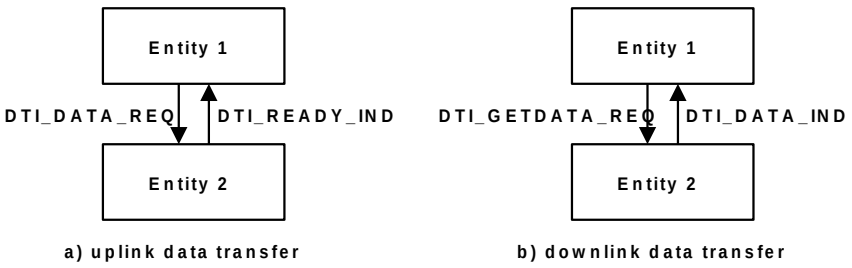

Figure 4: Data and flow control primitives

A data primitive may only be sent, when the other entity has signalled with a flow control primitive, that it is ready to accept data. This means that the first primitive must be a flow control primitive in each direction. After every data primitive a new flow control primitive has to be sent.

To send a DTI_DATA_REQ primitive to the lower layer, a DTI_READY_IND must be received first from that layer.

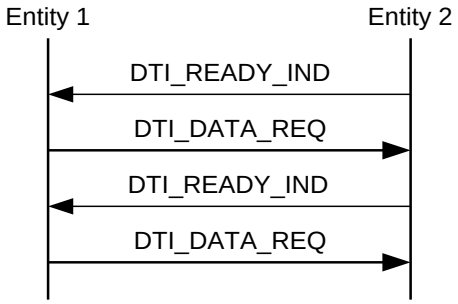

Figure 5: Uplink data transfer

To send a DTI_DATA_IND primitive to the upper layer, a DTI_GETDATA_REQ must be received first from that layer.

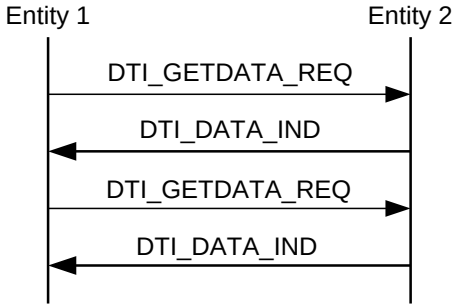

Figure 6: Downlink data transfer

4.2 The structure of data primitives

The DTI SAP must be capable of transferring complete packets for protocols like IP. These packets can have a considerable length. On the other hand the allocation of a big primitive is not efficient, when only a few bytes must be transferred. Therefore the DTI data primitives make use of linked lists of data blocks. This is efficient for transferring a small amount of data, because only a small block must be allocated. It allows however to transfer big packets by linking a number of data blocks into a list. This is also efficient with regards to the memory management, because it can limit the maximum size of a data block to a useful value and nevertheless bigger packets can be used.

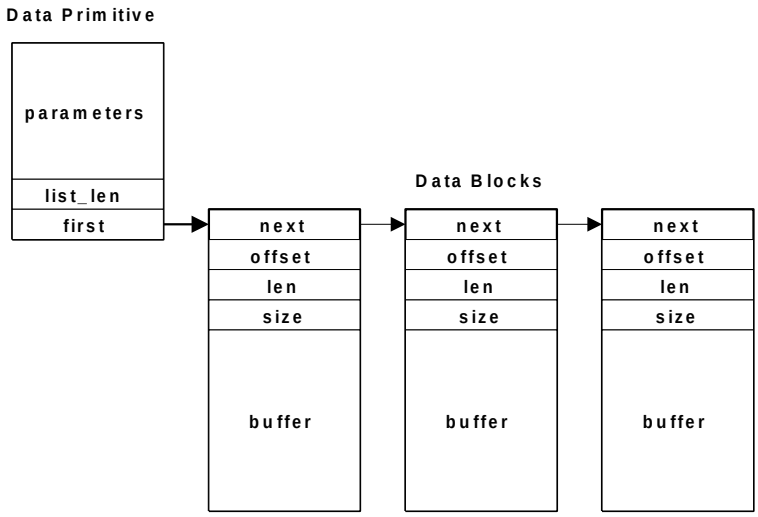

Figure 7: The structure of DTI2 data primitives

The data primitives contain a set of parameters, a pointer to the first data block *first* and the overall number of bytes stored in the linked list *list_len*. This overall number is a 16 bit value and therefore the maximum number of bytes in a DTI data primitive is limited to 64 kBytes.

Each data block contains a pointer *next* to the next data block. The last *next* pointer has the value NULL. The pointers *first* and *next* are defined as ULONG values, because the ccdgen tools can not handle pointers. Therefore in the program code these pointers must always be casted on the proper types.

The *buffer* can be of any length. After allocating a data block the size of the buffer must be stored in the parameter *size*, this is particularly useful, when a data block is reused by the receiver of a primitive or if data are added to an existing buffer.

The parameters *offset* and *len* mark the starting point and the length of data in the buffer calculated in bytes.

The parameters *offset* and *size* are only available in version 2 of the DTI SAP.

4.3 Parameters in data and flow control primitives

It is possible, that one entity is connected to several other entities via DTI at the same time. This will be shown in more detail in 6. In this case the receiver of a primitive must find out, from where the primitive has been sent and to which channel it belongs to.

In the DTI version 1 there are 3 parameters for specifying the originator and channel etc.:

Name	Meaning	Remarks
tui	transmission unit identifier	This is an id for the sender.
c_id	channel identifier	This is an id, which is unique in the whole system and specifies a single DTI connection.
op_ack	acknowledged operation flag	This parameter is not used any more.

In the DTI version 2 there is only 1 parameter for specifying the originator and channel etc.:

Name	Meaning	Remarks
link_id	link identifier	This is an id, which is unique in the whole system and specifies a single DTI connection.

4.4 Additional parameters in data primitives

There are some parameters, which are transferred along with data, e.g. line states of an UART interface. If a state changes, but no data are to be sent, then an empty primitive containing no data should be sent.

Name	Meaning	DTI1	DTI2	Remarks
p_id	protocol identifier	x	x	This parameter specifies, what type of data packed is contained in the primitive
st_flow	line state flow control	x	x	End to end flow control (X-bit) used in CSD
st_line_sa	line state SA bit	x	x	SA bit used in CSD, mapped onto UART lines
st_line_sb	line state SB bit	x	x	SB bit used in CSD, mapped onto UART lines
st_escape	line state escape	x		Indication of escape sequence (“+++”)
st_break_len	line state BREAK		x	Indication of a BREAK condition

All parameters must be initialised with proper default values before sending a primitive, even if the parameter has no meaning in that particular connection. If for example *st_break_len* is not initialised, this might result in a BREAK processing in L2R, even if the sender of the primitive can actually not generate a BREAK condition.

The parameter *st_escape* has been removed from the DTI SAP in version 2. The escape sequence (“+++”) is entered by the user or the application via the UART to come back into command mode, when a data connection is established. The detection of this escape sequence in the UART is not a matter of the data transfer but an issue which must be handled between the UART task and the ACI.

The parameter *st_break_len* has been introduced in DTI 2. BREAK is a matter of the data transfer in CSD. When a BREAK condition is raised, all buffers must be cleared and the BREAK must be signalled over the air. BREAK was supported previously outside of the DTI SAP.

5 Establishment of a DTI connection

The establishment of a DTI connection is always initiated by a controller instance. There is no way, that two entities can start communication without prior request from the controller. In the G23 protocol stack the controller is the ACI, which is not only responsible for establishing DTI connections but for setting up a complete protocol stack configuration. ACI must set up a call, start different entities and configure them with parameters. Along with this ACI is also setting up DTI connections.

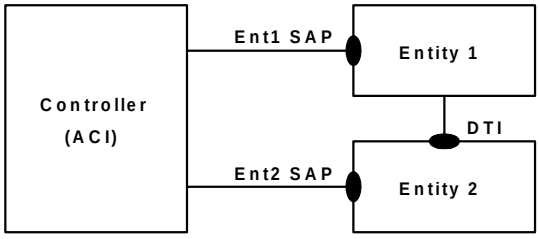

Figure 8: ACI controlling two entities with DTI SAP

For establishing a DTI connection the ACI has to send a primitive to each entity. This primitive contains the DTI related parameters (tui, c_id, op_ack or link_id) and other entity specific parameters (e.g. IP address in the case of IP). In most cases such entity specific parameters are required and therefore the SAP, which is used by ACI to configure the entities, is not generic but different for each entity.

There is always a delay between the set up of the two entities. Therefore a synchronisation is required to make sure, that both entities are ready to transfer data.

In version 1 of the DTI SAP the synchronisation is done by sending a flow control primitive and an empty data primitive to the other entity. Due to the delay it might happen that one entity is receiving these primitives before being ready. In this case the primitives are ignored. As soon as this entity gets ready, it will send a flow control primitive to the other entity, which is al-

ready ready. It will also send an empty data primitive. Then the other entity will acknowledge the data primitive by a flow control primitive. In this way both entities receive a flow control primitive and the DTI connection is synchronised.

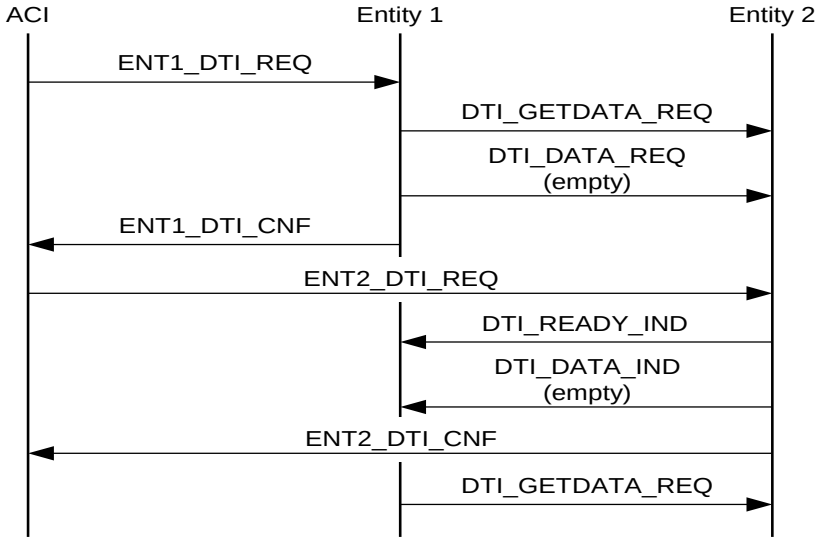

Figure 9: DTI 1 synchronisation

In version 2 of the DTI SAP there are dedicated primitives for connecting the two entities. The connection can be started by either side. Therefore a DTI_CONNECT_REQ or a DTI_CONNECT_IND must be send. When the receiver of this primitive is ready, it will respond with the corresponding DTI_CONNECT_CNF or DTI_CONNECT_RES (s. Figure 10). After the exchange of CONNECT primitives each entity will send a flow control primitive to the other entity.

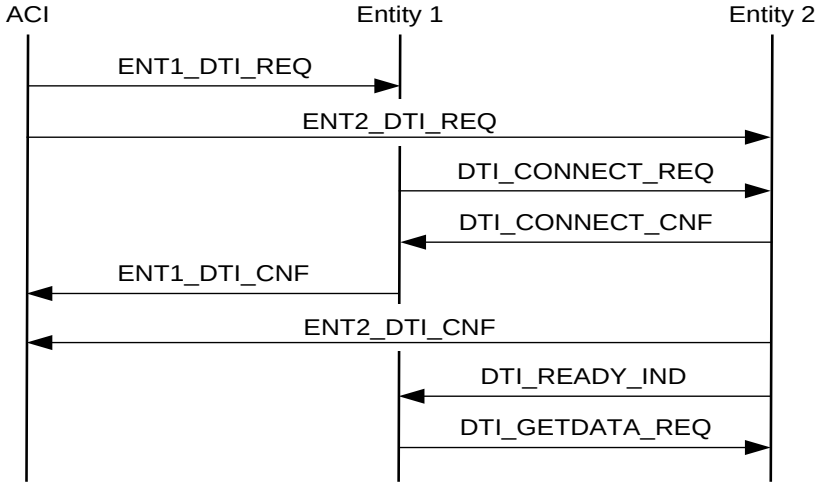

Figure 10: DTI 2 synchronisation

There is also a case, in which both entities start sending a CONNECT primitive at the same time. In this case both entities should respond with the proper acknowledgement. The DIT connection is regarded to be synchronised, when both acknowledgements have been sent. This case is not shown in a diagram here.

6 Multiple DTI connections in one entity

There are some cases, where an entity must handle more than one DTI connection.

It may be, that between two entities several channels are required. An example for this is SNDCP and PPP. If more then one PDP context is activated, then for each PDP context a separate DTI connection is established between PPP and SNDCP. In DTI 1 the parameter *c_id* is used to differentiate between channels.

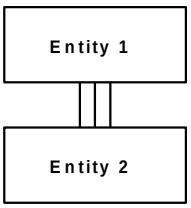

Figure 11: Multiple Channels

In another case one entity must communicate with different entities via DTI. A typical example is IP, which must be capable to handle different transport layer protocols like TCP and UDP and different data link layer protocols. In DTI 1 the parameter *c_id* is used to differentiate between the neighbour entities.

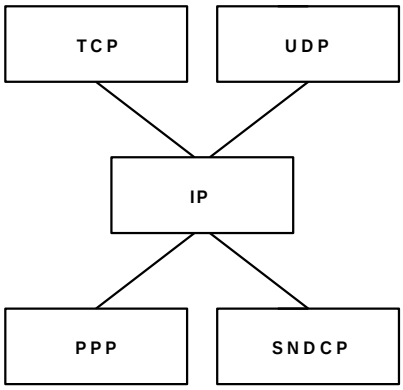

Figure 12: Multiple neighbour entities

If an entity exists in multiple instances, then each instance may communicate with different or with the same entities. The receiver of a primitive has to know, which of its instances is addressed, because all instances share the same primitive input queue. An example of this is PPP which is invoked as a PPP client in the case of a WAP over CSD call and as a PPP server in case of a normal GPRS connection. In DTI 1 the parameter *c_id* is used to differentiate between instances.

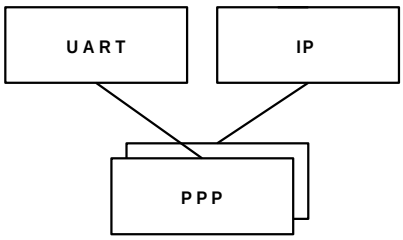

Figure 13: Multiple instances

In DTI 2 all these cases of multiple DTI connections are handled with the parameter *link_id*. Each DTI connection must be uniquely identified with the *link_id*. Then the receiver of a primitive can always derive from the *link_id* where the primitive was send from and to which channel or instance it belongs. The uniqueness of the *link_id* is guaranteed, because the values are managed and distributed by the ACI.

7 Relaying

The DTI SAP advocates a layered approach, in which a higher layer and a lower layer communicate with each other. Therefore there are two different data primitives and two different flow control primitives. The reason for this is, that such an approach is clearer and easier to work with in an protocol stack environment, which is based on layers by nature.

However there is an disadvantage of the layered approach, when a relay functionality is required. This is the case, when two entities of the same layer must communicate with each other. An example is L2R and the UART task. Both entities expect to communicate with a higher layer, but in fact they are connected to each other.

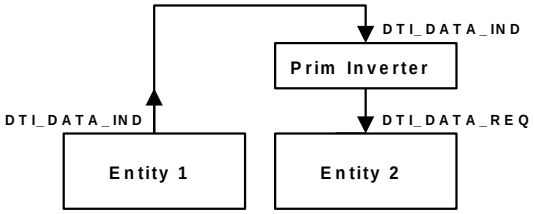

Figure 14: Relaying by primitive inversion

To solve the problem one of the entities must somehow invert primitives, i.e. send requests to the higher layer and receive indications from there. This can be accomplished by adding a primitive inverter to the entity, which is changing the opcode of all primitives, before sending and after receiving them. In this way the primitive inversion becomes transparent for the entity itself.

8 Testing

For testing an entity with DTI SAP in the simulation environment on a PC one additional feature is required. The test environment does not support pointers or linked lists in the data primitives. The only way to transfer user data is within SDUs, i.e. a block of data in the primitive itself.

Therefore two more data primitives are defined for testing. These primitives are identical to the data primitives, but contain the data itself in a SDU rather then a pointer to the data.

For the test environment the entity must be compiled such, that it sends test primitives instead of normal data primitives.

In order to test the entity code completely, the entity should generate a normal data primitive, i.e. a linked list, and then before sending the primitive it should be copied into a test primitive. When receiving a test primitive, the data should first be converted into a linked list and then be processed in the normal way. It should be made sure, that the block size for data in the linked list is small enough, so that real linked lists with several data blocks are generated. Otherwise it can not be tested, if the entity handles linked lists properly.

This can be regarded as a converter, that is added to the entity and which converts normal data primitives into test primitives and vice versa.

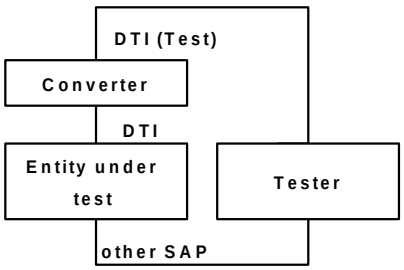

Figure 15: Entity test with test converter

There is a limitation to this kind of testing. The SDUs have a maximum size, whereas DTI linked lists can be very long. The test cases must be written in such a way, that the entity does not generate data primitives, that are too long to be converted into SDUs. Otherwise the behaviour of the system is unpredictable.

9 The DTI library

All entities using the DTI SAP require the same or similar functionality for handling the DTI SAP and DTI primitives. Therefore a library module DTILIB has been written, which contains functions to handle the DTI SAP.

The advantages of the DTILIB are:

- reduced code size, because the same code is used by different entities

- better software quality, because only one well tested module handles the DTI instead of different implementations in the entities.
- simplified usage of the DTI SAP for the entity developer.
- easier migration to newer DTI versions, because many details of the DTI SAP are hidden by the library.

The DTILIB is not an entity, but a library, which provides a set of functions. These functions can be called from every entity that uses the DTI SAP. An entity using the DTILIB must also provide a callback function to the DTILIB, so that events can be reported to the entity.

The DTILIB is linked to the protocol stack. To use it, an entity can just call the functions, that are provided by the DTILIB. The communication between entities is still via primitives, but the entities don't have to handle the sending and receiving of primitives, since this is done in the DTILIB.

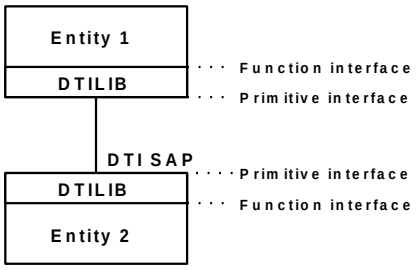

Figure 16: Entities using the DTILIB

To send data from entity 1 to entity 2, entity 1 calls a function of the DTILIB. This functions sends a primitive via the DTI SAP to entity 2. A callback function is called there by the DTILIB, which passes the data to the entity 2. Therefore the entities don't have to use the DTI SAP directly, but can make use of the function interface of the DTILIB.

Some additional features of the DTILIB are:

- Support of a send queue
- Support of DTI 1 - and DTI 2 SAP
- Support of test primitives
- Support of different ways of flow control transparent to the entity
- Support of relaying by primitive inversion

The DTILIB specification contains two sets of functions:

- Functions which handle the sending and receiving of primitives.
- Utility functions, which are helpful for reading and writing data primitives. These utility functions are not implemented yet.

9.1 The DTILIB database

The entity related variables for the connection and for the data transmission are placed in a database. Every entity that uses DTILIB has such a database. There is an entry in the database for every DTI connection, also called a *link*.

The entity allocates the required memory for this database with a function call during the initialisation. The DTILIB does not store any data in static variables. As a result the DTILIB is reentrant and can be called from different entities at the same time without conflicts.

A part of the database is the link structure, which contains the required parameters for one link. If the entity uses more than one link, then the link structures are kept in a linked list. DTILIB selects the correct link structure with a parameter called *link_id*, related to DTI SAP 2. In DTI SAP 1, DTILIB selects the correct channel structure according to the parameter *c_id*. The structure of the database is described in more detail in the specification of the DTILIB [53].

The database contains also a pointer to the entity Call Back Function, which is activated to indicate an event to the entity. Detailed information about this callback function are given in the specification of the DTILIB [53].

9.2 Instance, Interface and Channel

In version 2 every link is identified by the *link_id* in the DTI SAP. On the function interface of the DTILIB however the link is identified by the triple *instance*, *interface* and *channel*. The values of these parameters can be selected by the entity in a useful way. They have no meaning to the DTILIB.

When a primitive is received, the DTILIB translates the *link_id* into the corresponding values of the parameters *instance*, *interface* and *channel* and outputs this to the entity. When a primitive is to be sent, DTILIB translates these parameters into a *link_id*. Therefore every entry in the database contains the *link_id* along with the corresponding parameter triple.

Although the three parameters can be used in any way, it is suggested to use them in a way, which is shown in the following example:

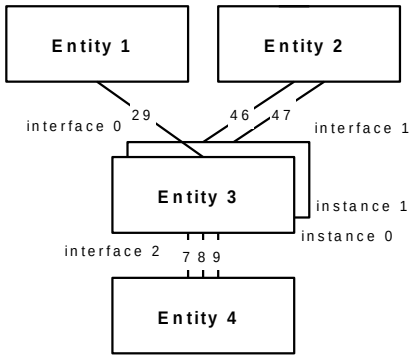

Figure 17: Entity with multiple links

Entity 3 communicates with 3 other entities via DTI. In most cases an upper layer and a lower layer requires different functionality from the entity. Therefore it is a good idea to differentiate between the connections to the upper layer and the connection to the lower layer. This can be done with the parameter *interface*. Entity 1 and entity 2 may also require different functionality or parameters. Therefore in this example 3 different interfaces 0, 1 and 2 are defined.

Moreover entity 3 has 2 instances, which are distinguished by the parameter *instance*.

There are 3 links to entity 4 and 2 links to entity 2. They are distinguished by the parameter *channel*. Figure 17 does not show the channel numbers, but the *link_id* 1, 2,...6, which is fixed by the ACI.

link_id	in-stance	inter-face	channel	direc-tion
29	0	0	0	up
46	1	1	0	up
47	1	1	1	up
7	0	2	0	down
8	0	2	1	down
9	0	2	2	down

Figure 18: Database of entity 3

The advantage of using the parameter triple as shown in the example is, that the entity can use all these parameters as an index for it's own operation. The *instance* can be used directly to select the data space of the instance. The *interface* can be used to select tables or variables related to the interface. The parameter *channel* is useful to address an array, that holds data for every channel. Of course all this could be done by the entity itself, but then it would have to translate the *link_id* by itself into useful values.

The parameter *direction* is used to determine, if indication or request primitives must be sent. With this parameter relaying as described in chapter 7 becomes possible.

There is also a number of other parameters in the database, which are not shown here.

Appendices

A. Acronyms

DS-WCDMA Direct Sequence/Spread Wideband Code Division Multiple Access

B. Glossary

International Mobile Telecommunication 2000 (IMT-2000/ITU-2000) Formerly referred to as FPLMTS (Future Public Land-Mobile Telephone System), this is the ITU's specification/family of standards for 3G. This initiative provides a global infrastructure through both satellite and terrestrial systems, for fixed and mobile phone users. The family of standards is a framework comprising a mix/blend of systems providing global roaming. <URL: http://www.imt-2000.org/>