Technical Document

LLD DTI CONTROL MANAGER

Document Number:	8462.709.02.002
Version:	0.3
Status:	Draft
Approval Authority:	
Creation Date:	2002-Oct-19
Last changed:	2015-Mar-08 by XGUTTEFE
File Name:	LLD_DTI_Cntrl_Mng.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.

No part of this document may be reproduced or transmitted in any form or by any means, electronically or mechanically, including photocopying and recording, for any purpose without the express written permission of TI.

Change History

Date	Changed by	Approved by	Version	Status	Notes
2002-Oct-31	SKA		0.1	Planned	1
2003-Mar-13	TLU		0.2	Planned	2
2003-May-19	XGUTTEFE		0.3	Draft	3

Notes:

1. Initial version

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

A. References, Abbreviations, Terms

[TI 7010.801]	7010.801, References and Vocabulary, Texas Instruments
[TI 8415.052]	8415.052, TI Specific AT Commands
[TI 8462.703]	8462.703, LLD_DTI_Conn_Mng.doc
[TI 1234.567]	1234.123, AT commands for Packet IO.doc
[TI 1234.567]	1234.123, DTI_Lib.doc

1.1 Abbreviations

1.2 Terms

Entity: Program which executes the functions of a layer

Message: A message is a data unit, which is transferred between the entities of the same layer (peer-to-peer) of the mobile and infrastructure side. Message is used as a synonym to protocol data unit (PDU). A message may contain several information elements.

Primitive: A primitive is a data unit, which is transferred between layers on one component (mobile station or infrastructure). The primitive has an operation code, which identifies the primitive and its parameters.

Service Access Point: A Service Access Point is a data interface between two layers on one component (mobile station or infrastructure).

Device: A data sink or source out of GSM/GPRS protocol stack

2 Introduction

This document is a Low Level Design for the new module DTI *Control* Manager, which works together with the new module DTI *Connection* Manager. The document describes the establishing and releasing of DTI channels. It is assumed that the reader has a basic understanding about the current Data Transmission Interface (DTI). Under this term there is currently a library used by entities of the G23M protocol stack, which want to get a data transmission connection. These connections are managed by the DTI Manager, which locates in the ACI entity.

This current DTI Manager will be divided in two parts to get a clearer design of it.

The two parts are called DTI *Control* Manager and DTI *Connection* Manager. This document describes the DTI Control Manager part and necessarily extensions and adaptations in the protocol stack adapter PSA and in the command control handler CMH. In **Figure 1 The DTI Control Manager** these items are gray colorized.

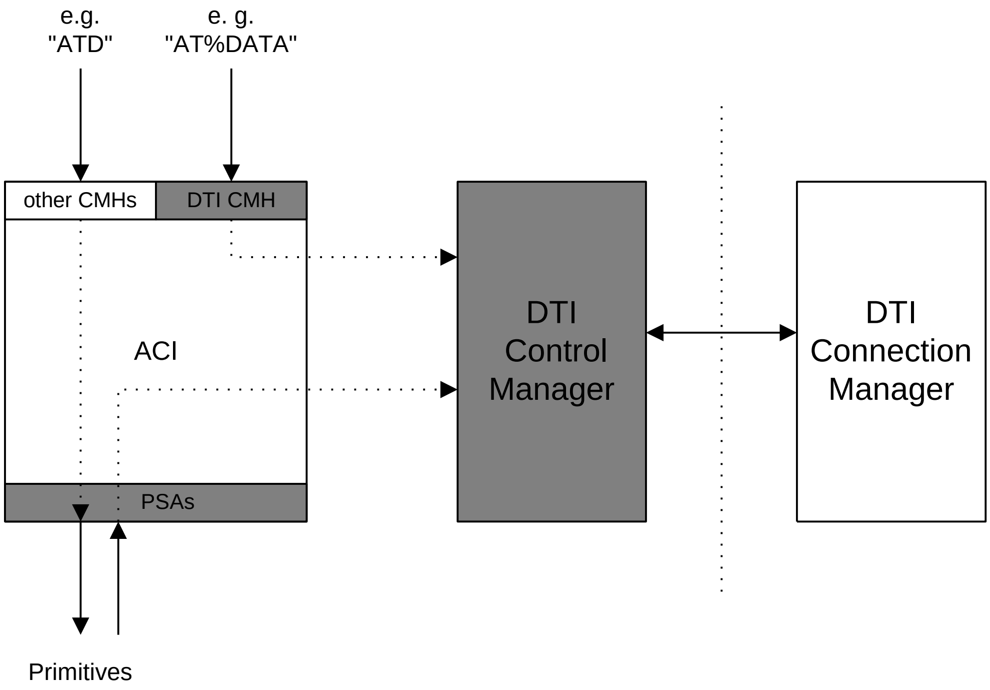

Figure 1 The DTI Control Manager

New AT commands will be introduced to configure the DTI channels and an additional command handler will be added to handle these AT commands. For the protocol stack adapter PSA, there are extensions as well.

2.1 General Description of Establishing a DTI Channel

After power on of the mobile, all devices, which are able to send and receive data, have to send a primitive to ACI to inform the DTI Control Manager about their capabilities. Capabilities means, what kind of data communication over the channel the device is able to:

a) command mode

b) packet data

c) serial data

To establish a DTI channel, at first an AT cmd channel is used by a user/application to query the capability of the devices which will be involved for the data communication and then to configure the DTI channel. After a successfully configuration the AT cmd channel will be switched to a data channel.

The querying of the capability of the devices is initiated by sending the AT command:

%DINF=<mode>

The configuration of a DTI channel is initiated by sending of the AT command:

%DATA=<mode>,<des_dev_name>,<des_dev_no>,<cap>[,<src_dev_name>,<src_dev_no>[,<cid>]]

See document [TI 1234.567] "AT commands for Packet IO.doc".

3 DTI Control Manager

The DTI Control Manager is responsible to gather and to maintain information about devices, which want to be connected for data transmission. The actual connection takes place in the DTI Connection Manager.

3.1 Internals of DTI Control Manager

3.1.1 Internal Structures and Types

3.1.1.1 T_DTI_CNTRL

```
typedef  struct
{
  T_DTI_ENTITY_ID           dev_id;        /* id(name) of device                        */
  UBYTE                     dev_no;        /* instance of device                        */
  UBYTE                     sub_no;        /* instance with multiplexed sub channels */
  UBYTE                     capabilities;  /* capabilities of device                    */
  UBYTE                     src_id;        /* what ACI sees as src                      */
  UBYTE                     dti_id;        /* id of DTI channel                         */
  UBYTE                     cur_cap;       /* capability of established DTI channel  */
  T_DTI_CNTRL_REDIRECT      redirect_info; /* union for redirection                     */
} T_DTI_CNTRL;
```

This is the internal maintenance structure used by the DTI Control Manager only. It has all necessary information of the device, the current DTI channel with its dti_id and its current capability. The redirection union is used in such cases where a device is not able to send AT commands to initiate a DTI connection. Then the application has to use a second device which is able to convey AT commands and with the AT%DATA command there will be embedded an order to the DTI Control Manager to redirect the data communication to that device, which was not able to send the AT command.

3.1.1.2 Sub channel numbering for devices able to multiplexing

The structure element `sub_no` of `T_DTI_CNTRL` is used to keep track the logical multiplexed channel of a device, which is able to do so. An example is the UART device when configured according to the ETSI GSM 07.10 recommendation.

3.1.1.3 CAPABILITY

```
#define        DTI_CPBLTY_CMD        0x01
#define        DTI_CPBLTY_PKT        0x02
#define        DTI_CPBLTY_SER        0x04
```

This bit field is used to keep the capability of a device and of a current DTI channel. There are generic macros to set and clear a bit. These macros are defined in dti.h.

3.1.1.4 T_DTI_CNTRL_REDIRECT

```
typedef union
{
  T_DTI_CNTRL_REDIRECT_INTERN *tbl;   /* intern */
  T_DTI_CNTRL_REDIRECT_EXTERN  info;  /* extern */
} T_DTI_CNTRL_REDIRECT;
```

This union is used for two different purposes.

At first, for internal using and data hiding, the first union member `T_DTI_CNTRL_REDIRECT_INTERN` is used to maintain the redirections of data channels.

At second, when a caller wants to get information about a device and its DTI connections (which can be redirected), the caller has to pass a structure pointer of the type `T_DTI_CNTRL` (which points to a memory space allocated by the caller) and the DTI Control Manager will fill it with a copy of the internal structure. If the device has been redirected, then the DTI Control Manager will traverse the internal list through the redirection pointer and fills the structure, passed by the caller, with the actual structure.

3.1.1.5 T_DTI_CNTRL_REDIRECT_EXTERN

```
typedef struct
{
  UBYTE  cid;         /* pdp context id            */
  UBYTE  mode;        /* once/permanent redirection */
  UBYTE  capability;  /* capability of redirection  */
  UBYTE  direction;   /* src/dst of redirection     */
} T_DTI_CNTRL_REDIRECT_EXTERN;
```

What the external sees.

3.1.1.6 T_DTI_CNTRL_REDIRECT_INTERN

```
typedef struct
{
  T_DTI_CNTRL_REDIRECT_TYPE ser_redirect[DTI_MAX_REDIRECTIONS]; /* redirected to a serial de-
vice */
  T_DTI_CNTRL_REDIRECT_TYPE pkt_redirect[DTI_MAX_REDIRECTIONS]; /* redirected to a packet de-
vice */
} T_DTI_CNTRL_REDIRECT_INTERN;
```

What uses the DTI Control Manager. There is a separated maintained filed for serial and packet devices.

3.1.1.7 T_DTI_CNTRL_REDIRECT_TYPE

```
typedef struct
{
  UBYTE         mode;              /* once/permanent                      */
  T_DTI_CNTRL  *redirection;       /* actual pointer to redirected device */
} T_DTI_CNTRL_REDIRECT_INTERN_TYPE;
```

A redirection can be set up as a permanent redirection or as a one time only one. When in once mode, the DTI Control Manager will remove the redirection after a data transmission is finished, where the application is responsible to request a redirection in mode once or permanent.

3.1.2 Internal Data Organization

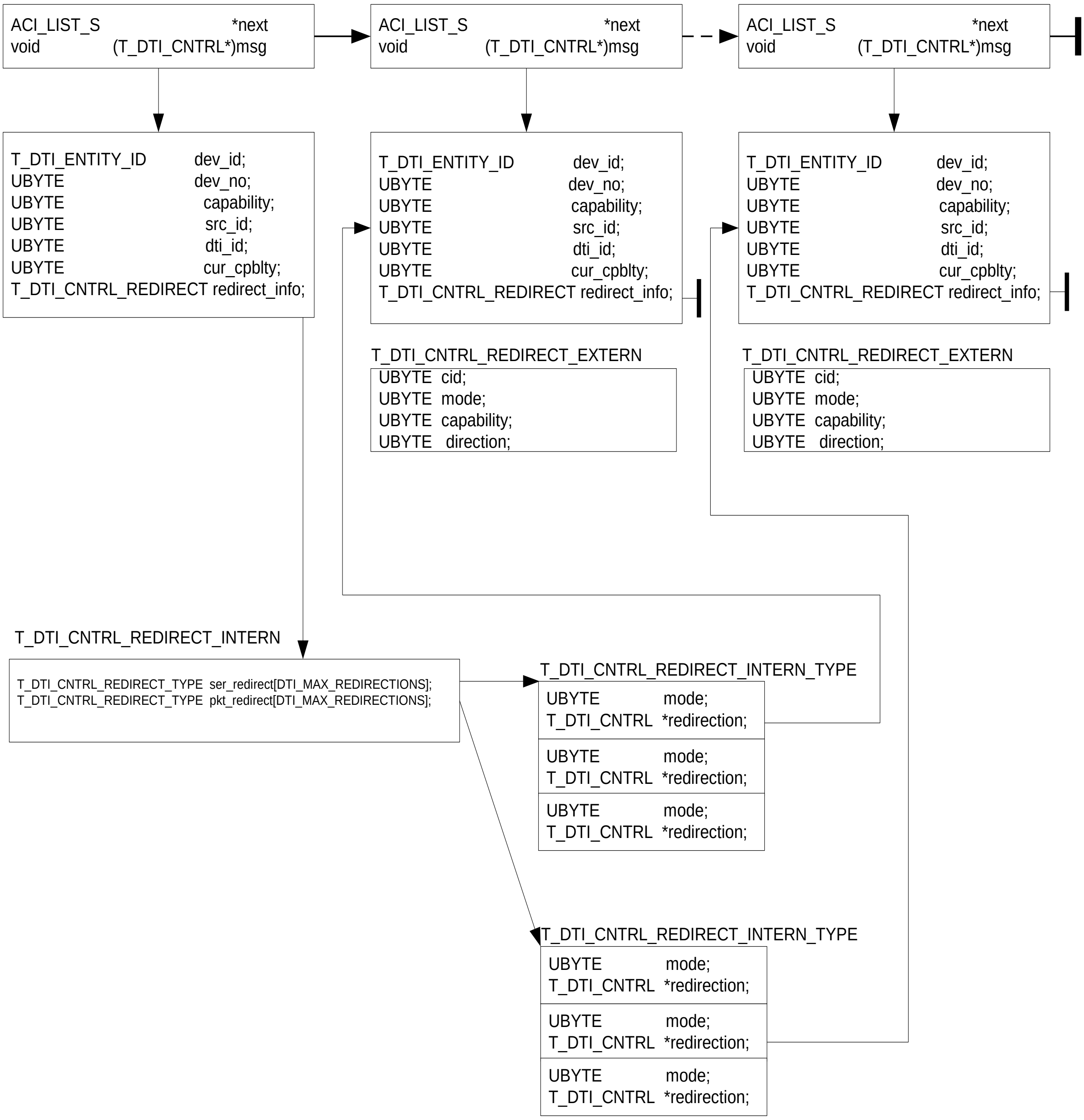

Figure 2 DTI Control Manager Intern

In this figure the generic container list module of ACI is used to build up a maintenance list for the devices and their possible DTI connections. As an example for the case of redirection there is shown for the first list element a redirection to a device with serial capability and a redirection to a device with packet capability.

3.1.3 Internal Functions

3.1.3.1 dti_cntrl_maintain_entity

Prototype: BOOL dti_cntrl_maintain_entity (T_DTI_CONN_LINK_ID link_id,
T_DTI_ENTITY_ID entity_id,
T_DTI_ENTITY_ID peer_entity_id,
UBYTE dti_conn);

Parameter: link_id
entity_id
peer_entity_id
dti_con

Return value: BOOL

Description: This function is used by the DTI Connection Manager, where it sees this function not directly, but as a pointer of that function prototype. The DTI Connection Manager will use this function pointer to notify such entities involved for a data transmission. The registration of this function by a function pointer takes place when the DTI Control Manager calls the DTI Connection Manger function “dti_conn_init()”. Refer to LLD_DTI_Conn_Mng.doc.

4 Public Interface

4.1 Public Structures and Types

4.1.1 T_DTI_CNTRL

```
typedef  struct
{
  T_DTI_ENTITY_ID          dev_id;        /* id(name) of device                   */
  UBYTE                    dev_no;        /* instance of device                   */
  UBYTE                    sub_no;        /* instance with multiplexed sub channels */
  UBYTE                    capability;    /* capability of device                 */
  UBYTE                    src_id;        /* what ACI sees as src                 */
  UBYTE                    dti_id;        /* id of DTI channel                    */
  UBYTE                    cur_cap;       /* capability of established DTI channel  */
  T_DTI_CNTRL_REDIRECT     redirect_info; /* union for redirection                */
} T_DTI_CNTRL;
```

As already explained in chapter 3.1.1.1 the union `T_DTI_CNTRL_REDIRECTION redirection` in `T_DTI_CNTRL` is used to hide the internal maintenance of redirected devices from externals. An external sees the union member `T_DTI_CNTRL_REDIRECT_EXTERN info` only.

4.1.1.1 T_DTI_CNTRL_REDIRECT

```
typedef union
{
  T_DTI_CNTRL_REDIRECT_INTERN *tbl;   /* intern  !!! NOT VISIBLE FOR EXTERNAL !!! */
  T_DTI_CNTRL_REDIRECT_EXTERN  info;  /* extern */
} T_DTI_CNTRL_REDIRECT;
```

The second union member is used for the public interface. There is no need of a union controller/discriminator to select the appropriate union member. The DTI Control Manager will always use the second union member when there is a request to get information about devices by calling of the functions dti_cntrl_get_info_from_src_id/dti_id/dev_id. These functions pass a pointer of the type `T_DTI_CNTRL` , which causes to use the second union member.

4.1.1.2 T_DTI_CNTRL_REDIRECT_EXTERN

```
typedef struct
{
  UBYTE  cid;         /* context id                */
  UBYTE  mode;        /* once/permanent redirection */
  UBYTE  capability;  /* capability of redirection  */
  UBYTE  direction;
} T_DTI_CNTRL_REDIRECT_EXTERN;
```

This structure is used to keep information about redirections requested by an external.

4.1.2 UBYTE src_id

The Source ID relates to ACI only, whereas the DTI Control Manager sets an association between a Source Id and a DTI Id.

It is possible, that there is no association between a Source ID and a DTI Id, due to the fact that some devices are not able to send AT commands. Then the Source Id is set to 0xFF = NO_SRC_ID. In such a case a redirection will be taking place.

4.1.3 UBYTE dti_id

The DTI id is set by the DTI Connection Manager and is read only by the DTI Control Manager.

4.1.4 UBYTE cid

The cid is used to identify the PDP context.

4.1.5 ULONG link_id

The DTI Control Manager itself never uses or manipulates this identifier, but passes it transparently between an external and the DTI Connection Manager.

4.1.6 T_DTI_ENTITY_ID entity_id

Identifier for a GSM/GPRS entity (SNDCP, L2R, TRA, SMS, …) handled by DTI Connection Manager.

4.1.7 UBYTE mode

How long a redirection exists: once/permanent. Used only by the DTI Control Manager

Attention ! For the two functions “dti_cntrl_est_dpath” and “dti_cntrl_est_dpath_indirect” there are parameters of the type T_DTI_CONN_MODE. These parameters are transparently passed to the DTI Connection Manager and have no meanings in the DTI Control Manager.

4.1.8 T_DTI_CONN_CB

This function pointer is defined in “LLD_DTI_Conn_Mng.doc”.

4.1.9 T_DTI_EXT_CB

This function pointer is used to setup a callback function of an external with the prototype:

typedef BOOL (*T_DTI_EXT_CB)(ULONG link_id, T_DTI_ENTITY_ID peer_entity_id, UBYTE dti_conn);

4.2 Public DTI Control Manager Functions

4.2.1 dti_cntrl_init

Prototype:	void dti_cntrl_init	(void)
Parameter:	void	

Return value: void

Description: This function is used to initialize the DTI Control Manager. The function itself will call dti_conn_init (T_DTI_CONN_MNT_ENT_CB *cb) to initialize the DTI Connection Manager and to register its own internal function "dti_cntrl_maintain_entity". These actions take place only once after power on.

4.2.2 dti_cntrl_new_dti

Prototype: UBYTE dti_cntrl_new_dti (UBYTE dti_id)

Parameter: dti_id /* identifier of an end-to-end connection */

Return value: UBYTE

Description: This function is used as a wrapper for the DTI Connection Manager function "dti_conn_new".

4.2.3 dti_cntrl_new_device

Prototype:	BOOL dti_cntrl_new_device	(UBYTE	src_id,
		UBYTE	dev_id,
		UBYTE	dev_no,
		UBYTE	sub_no,
		UBYTE	capability)

Parameter:	src_id	/* when not NO_SRC_ID → there is a instance of command interpreter */
	dev_id	/* device id */
	dev_no	/* instance of the device */
	sub_no	/* sub-channel number (e.g for multiplexer) */
	capability	/* capabilities of the device (CMD,PKT,SER) */

Return value: BOOL

Description: This function is used to register a new device in the DTI Control Manager. If the capability informs about AT command ability, then for this (instance of) device there will be one AT interpreter.

4.2.4 dti_cntrl_est_dpath

Prototype:	BOOL dti_cntrl_est_dpath	(UBYTE	dti_id,
		T_DTI_ENTITY_ID	*entity_list,
		UBYTE	num_entities,
		T_DTI_CONN_MODE	mode,
		T_DTI_CONN_CB	*cb)

Parameter:	dti_id	/* DTI ID */
	*entity_list	/* list of entities to connect (is variable) */
	num_entities	/* number of entities in that list */
	mode	/* split or append */
	cb	**/*** call back function pointer */

Return value: BOOL

Description: This function is used to establish a data path between a **NOT** registered device and entity/entities.

4.2.5 dti_cntrl_est_dpath_indirect

Prototype:

```
BOOL dti_cntrl_est_dpath_indirect (UBYTE             src_id,
                                   T_DTI_ENTITY_ID   *entity_list,
                                   UBYTE             num_entities
                                   T_DTI_CONN_MODE   mode,
                                   T_DTI_CONN_CB     *cb,
                                   UBYTE             capability,
                                   UBYTE             cid)
```

Parameter:

src_id	/* when not NO_SRC_ID → there is a instance of command interpreter */
*entity_list	/* list of entities to connect (is variable), but excluding device */
num_entities	/* number of entities in that list */
mode	/* split / append */
cb	/* all back function pointer */
capability	/* CMD|PKT|SER */
cid	/* PDP context */

Return value: BOOL

Description: This function is used to establish a data path between a registered device and entity/entities.

4.2.6 dti_cntrl_close_dpath_from_src_id

Prototype:

```
BOOL dti_cntrl_close_dpath_from_src_id   ( UBYTE        src_id)
```

Parameter: src_id /* when not NO_SRC_ID → there is a instance of command interpreter */

Return value: BOOL

Description: This function is used to close a data transmission path by a given src_id.

4.2.7 dti_cntrl_close_dpath_from_dti_id

Prototype:

```
BOOL dti_cntrl_close_dpath_from_dti_id   ( UBYTE        dti_id)
```

Parameter: dti_id /* DTI ID */

Return value: BOOL

Description: This function is used to close a data transmission path by a given dti_id.

4.2.8 dti_cntrl_is_dti_channel_connected

Prototype:

```
BOOL dti_cntrl_is_dti_channel_connected (T_DTI_ENTITY_ID     ent_id,
                                         UBYTE               dti_id)
```

Parameter:

ent_id	/* entity ID */
dti_id	/* DTI ID */

Return value: BOOL

Description: This function returns TRUE if the DTI Connection given by dti_id is connected and the entity with the given ent_id is part of the connection. This is a wrapper function for the DTI Connection Manager.

4.2.9 dti_cntrl_is_dti_channel_disconnected

Prototype: BOOL dti_cntrl_is_dti_channel_connected (UBYTE dti_id)

Parameter: dti_id /* DTI ID */

Return value: BOOL

Description: This function returns TRUE if the DTI Connection given by dti_id is disconnected. This is a
wrapper function for the DTI Connection Manager.

4.2.10 dti_cntrl_get_info_from_src_id

Prototype: BOOL dti_cntrl_get_info_from_src_id (UBYTE src_id,
T_DTI_CNTRL *info)

Parameter: src_id /* when not NO_SRC_ID → there is a instance of command interpreter */
info / to get all information of redirection */

Return value: BOOL

Description: This function is used to get information of redirection initiated by a src_id.

4.2.11 dti_cntrl_get_info_from_dti_id

Prototype: BOOL dti_cntrl_get_info_from_dti_id (UBYTE dti_id,
T_DTI_CNTRL *info)

Parameter: dti_id /* identifier of an end-to-end connection */
info / to get all information of redirection */

Return value: BOOL

Description: This function is used to get information of redirection initiated by a dti_id.

4.2.12 dti_cntrl_get_info_from_dev_id

Prototype: BOOL dti_cntrl_get_info_from_dev_id (UBYTE dev_id,
UBYTE dev_no,
UBYTE sub_no,
T_DTI_CNTRL *info)

Parameter: dev_id /* device id */
dev_no /* instance of the device */
sub_no /* sub-channel number */
info / to get all information of redirection */

Return value: BOOL

Description: This function is used to get information of redirection initiated by a dev_id.

4.2.13 dti_cntrl_set_redirect_from_src

Prototype:	BOOL dti_cntrl_set_redirection_from_src	(UBYTE	src_id,
		UBYTE	mode,
		UBYTE	dst_dev_id,
		UBYTE	dst_dev_no,
		UBYTE	dst_sub_no,
		UBYTE	capability,
		UBYTE	cid)

Parameter:	src_id	/* when not NO_SRC_ID → there is a instance of command interpreter */
	mode	/* redirection exists once / permanent */
	dst_dev_id	/* device id */
	dst_dev_no	/* instance of the device */
	dst_sub_no	/* sub-channel number (e.g for multiplexer) */
	capability	/* CMD|PKT|SER */
	cid	/* PDP context */

Return value:

Description: This function is used to redirect the data transmission of the current (AT cmd) channel to different data channel.

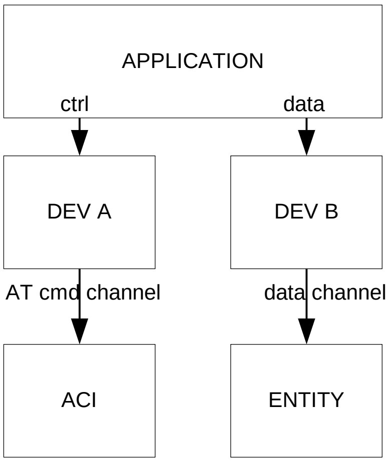

Figure 3

In this example after an ATD12345678 the AT cmd channel of “DEVA” will NOT be switched to a data channel of “DEV A”, rather there is a redirection to “DEV B” for the data transmission.

4.2.14 dti_cntrl_set_redirect_from_device

Prototype: BOOL dti_cntrl_set_redirection_from_device (UBYTE mode,
UBYTE dst_dev_id,
UBYTE dst_dev_no,
UBYTE dst_sub_no,
UBYTE src_dev_id,
UBYTE src_dev_no,
UBYTE src_sub_no,
UBYTE capability,
UBYTE cid)

Parameter:	mode	/* redirection exists once / permanent */
	dst_dev_id	/* device id of destination */
	dst_dev_no	/* instance of destination device */
	dst_sub_no	/* dest. sub-channel number (e.g for multiplexer) */
	src_dev_id	/* device id of source */
	src_dev_no	/* instance of source device */
	src_sub_no	/* source sub-channel number (e.g for multiplexer) */
	capability	/* CMD|PKT|SER */
	cid	/* PDP context */

Return value:

Description: This function is used to redirect a further cmd channel (which is identified by a tupel src_dev_id/srv_dev_no) to a data channel.

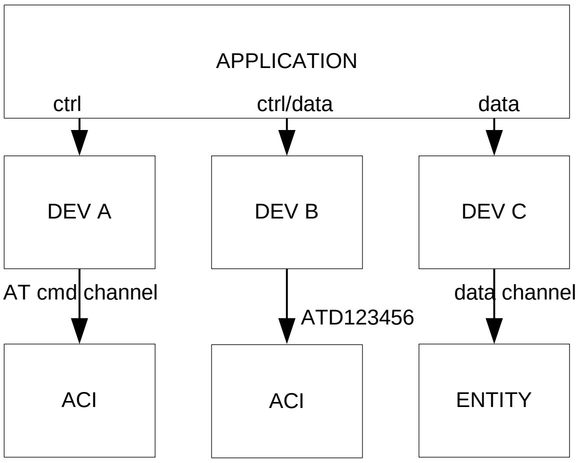

Figure 4

In this example the configuration of the redirection took place over DEV A (with its own instance of an AT interpreter), the initiation of the data transmission by ATD123456 took place over DEV B (with its own instance of an AT interpreter), but the actual data transmission takes place over DEV C.

4.2.15 dti_cntrl_get_first_device

Prototype: BOOL dti_cntrl_get_first_device (T_DTI_CNTRL *redirect)

Parameter: *redirect /* to get information of the redirected device */

Return value: BOOL

Description: This function is used in conjunction with AT%DATA? to start the querying form the first maintained device.

4.2.16 dti_cntrl_get_next_device

Prototype: BOOL dti_cntrl_get_next_device (T_DTI_CNTRL *redirect)

Parameter: *redirect /* to get information of the redirected device */

Return value: BOOL

Description: This function is used in conjunction with AT%DATA? to get the next maintained device.

4.2.17 dti_cntrl_get_first_redirection

Prototype: BOOL dti_cntrl_get_first_redirection (UBYTE src_id,
UBYTE capability,
T_DTI_CNTRL *redirect)

Parameter: src_id /* when not NO_SRC_ID → there is a instance of command interpreter */
capability /* CMD|PKT|SER */
redirect / to get information of the redirected device */

Return value: BOOL

Description: This function is used in conjunction with AT%DATA? and when an application wants to use the redirection of data transmission.

4.2.18 dti_cntrl_get_next_redirection

Prototype: BOOL dti_cntrl_get_next_redirection (UBYTE src_id,
UBYTE cid,
UBYTE capability,
T_DTI_CNTRL *redirect)

Parameter: src_id /* when not NO_SRC_ID → there is a instance of command interpreter */
cid /* PDP context */
capability /* CMD|PKT|SER */
redirect / to get information of the redirected device */

Return value: BOOL

Description: This function is used in conjunction with AT%DATA? and when an application wants to use the redirection of data transmission.

4.2.19 dti_cntrl_entitiy_connected

Prototype: void dti_cntrl_entitiy_connected (ULONG link_id,
T_DTI_ENTITY_ID entitity_id,

T_DTI_CONN_RESULT result)

Parameter: link_id /* unique identifier of a DTI connection between two entities */

entitity_id /* maintained by DTI Connection Manager */

result /* DTI_OK or DTI_ERROR */

Return value: void

Description: This function is used to indicate a connected entity (called by PSA).

4.2.20 dti_cntrl_entitiy_disconnected

Prototype: void dti_cntrl_entitiy_disconnected (ULONG link_id,
T_DTI_ENTITY_ID entitity_id)

Parameter: link_id /* unique identifier of a DTI connection between two entities */

entity_id /* maintained by DTI Connection Manager */

Return value: void

Description: This function is used to indicate a disconnected entity (called by PSA).

4.2.21 dti_cntrl_reg_new_fct

Prototype: void dti_cntrl_reg_new_fct (T_DTI_ENTITY_ID entity_id,
T_DTI_EXT_CB *fct)

Parameter: entity_id /* maintained by DTI Connection Manager */
fct / call back function pointer */

Return value: void

Description: This function is used to register a callback function of an external. The calling of a registered external function takes place in “dti_cntrl_maintain_entity”, where this function itself is a call back function used by the DTI Connection Manager.

4.2.22 dti_cntrl_erase_entry

Prototype: void dti_cntrl_erase_entry (UBYTE dti_id)

Parameter: dti_id /* DTI ID */

Return value: void

Description: This function is used to erase/remove a list element from the internal list maintained by the DTI control Manager. This can be the case, when the UART is reconfigured from a multiplexed device to a non-multiplexed device.

4.2.23 dti_cntrl_close_all_connections

Prototype: void dti_cntrl_close_all_connections()

Parameter:

Return value: void

Description: This function closes all established DTI channels and erases its entries from DTI channel list in the DTI Connection Manager. Further all other DTI channels are removed from DTI channel list.

4.2.24 dti_cntrl_set_conn_parms

Prototype: BOOL dti_cntrl_set_conn_parms (T_DTI_CONN_LINK_ID link_id,
T_DTI_ENTITY_ID ent_id,
UBYTE dev_no,
UBYTE sub_no)

Parameter: link_id /* unique identifier of a DTI connection between two entities */
entitity_id /* entity ID */
dev_no /* instance of the device */
sub_no /* sub-channel number (e.g for multiplexer) */

Return value: BOOL

Description: This function saves the association between the given parameters. This is necesarry, because some SAPs do not provide the link_id in its DTI confirmation primitives.

4.2.25 dti_cntrl_get_link_id

Prototype: T_DTI_CONN_LINK_ID dti_cntrl_get_link_id (T_DTI_ENTITY_ID ent_id,
UBYTE dev_no,
UBYTE sub_no)

Parameter: entitity_id /* entity ID */

dev_no /* instance of the device */

sub_no /* sub-channel number (e.g for multiplexer) */

Return value: T_DTI_CONN_LINK_ID

Description: This function restores the link id by using the given parameters. This is necesarry, because some SAPs do not provide the link_id in its DTI confirmation primitives.

4.2.26 dti_cntrl_get_peer

Prototype: T_DTI_ENTITY_ID dti_cntrl_get_peer (T_DTI_ENTITY_ID ent_id,

UBYTE dev_no,

UBYTE sub_no)

Parameter: entitity_id /* entity ID */

dev_no /* instance of the device */

sub_no /* sub-channel number (e.g for multiplexer) */

Return value: T_DTI_ENTITY_ID

Description: This function restores the communication peer of a tuple connection by using the given parameters. The given entity should only have only one peer to get a correct return value.

4.2.27 dti_cntrl_clear_conn_parms

Prototype: void dti_cntrl_clear_conn_parms (UBYTE dti_id)

Parameter: dti_id /* identifier of an end-to-end connection */

Return value: void

Description: This function clears the parameters, which were stored by dti_cntrl_set_conn_parms().

4.2.28 dti_cntrl_set_dti_id_to_reconnect

Prototype: void dti_cntrl_set_dti_id_to_reconnect (UBYTE dti_id)

Parameter: dti_id /* identifier of an end-to-end connection */

Return value: void

Description: This function sets an internal flag ('Reconnect-to-ACI'). If this flag is set then when the current DTI connection is completely disconnected then the registered device is reconnected to the ACI.

4.2.29 dti_cntrl_clear_dti_id_to_reconnect

Prototype: void dti_cntrl_clear_dti_id_to_reconnect (UBYTE dti_id)

Parameter: dti_id /* identifier of an end-to-end connection */

Return value: void

Description: This function clears the 'Reconnect-to-ACI' flag.

4.2.30 dti_cntrl_is_dti_id_to_reconnect

Prototype: BOOL dti_cntrl_is_dti_id_to_reconnect (UBYTE dti_id)

Parameter: dti_id /* identifier of an end-to-end connection */

Return value: BOOL

Description: This function returns TRUE if the 'Reconnect-to-ACI' flag is set for the given *dti_id*.

4.3 Public AT Command related functions

4.3.1 sAT_PercentDINF

Prototype:	T_ACI_RETURN sAT_PercentDINF	(T_ACI_CMD_SRC	srcId,
		UBYTE	mode,
		T_DINF_PARAM	*device_param)

Parameter: srcId

mode

device_param

Return value: T_ACI_RETURN

Description: This function is used to get information about AT cmd and data channels and their capabilities.

4.3.2 sAT_PercentDATA

Prototype:	T_ACI_RETURN sAT_PercentDATA	(T_ACI_CMD_SRC	srcId,
		UBYTE	redir_mode,
		CHAR	*des_devname,
		UBYTE	des_devno,
		UBYTE	des_subno,
		CHAR	*dev_cap,
		CHAR	*src_devname,
		UBYTE	src_devno,
		UBYTE	src_subno,
		UBYTE	pdp_cid)

Parameter: */	srcId	/* ACI source ID
	redir_mode */	/* deactivate or activate (one time / permanent) the data path
	des_devname */	/* name of the destination device
	des_devno */	/* instance of the destination device
	des_subno */	/* sub-channel number of the destination device
	dev_cap */	/* capability of destination device
	src_devname */	/* name of the source device
	src_devno */	/* instance of the source device
	src_subno */	/* sub-channel number of the source device
	pdp_cid	/* specifies a PDP context

Return value: T_ACI_RETURN

Description: This function is used to configure the data flow.

5 Adaptations and Extensions

5.1 Command Handler

Is described in its own document.

5.2 Protocol Stack Adapter

Is described in its own document.

6 Message Sequence Charts

Only the most important function's paramters are shown in the following figures. For example, the PSA passes a call back function pointer (atiUART_dti_cb) to the DTI Control Manager, where itself passes this function pointer to the DTI Connection Manager. The DTI Connection Manager uses this function pointer to inform the ATI about the established connection. See Figure 5.

6.1 Setup an ATI Channel

This example assumes as device the UART.

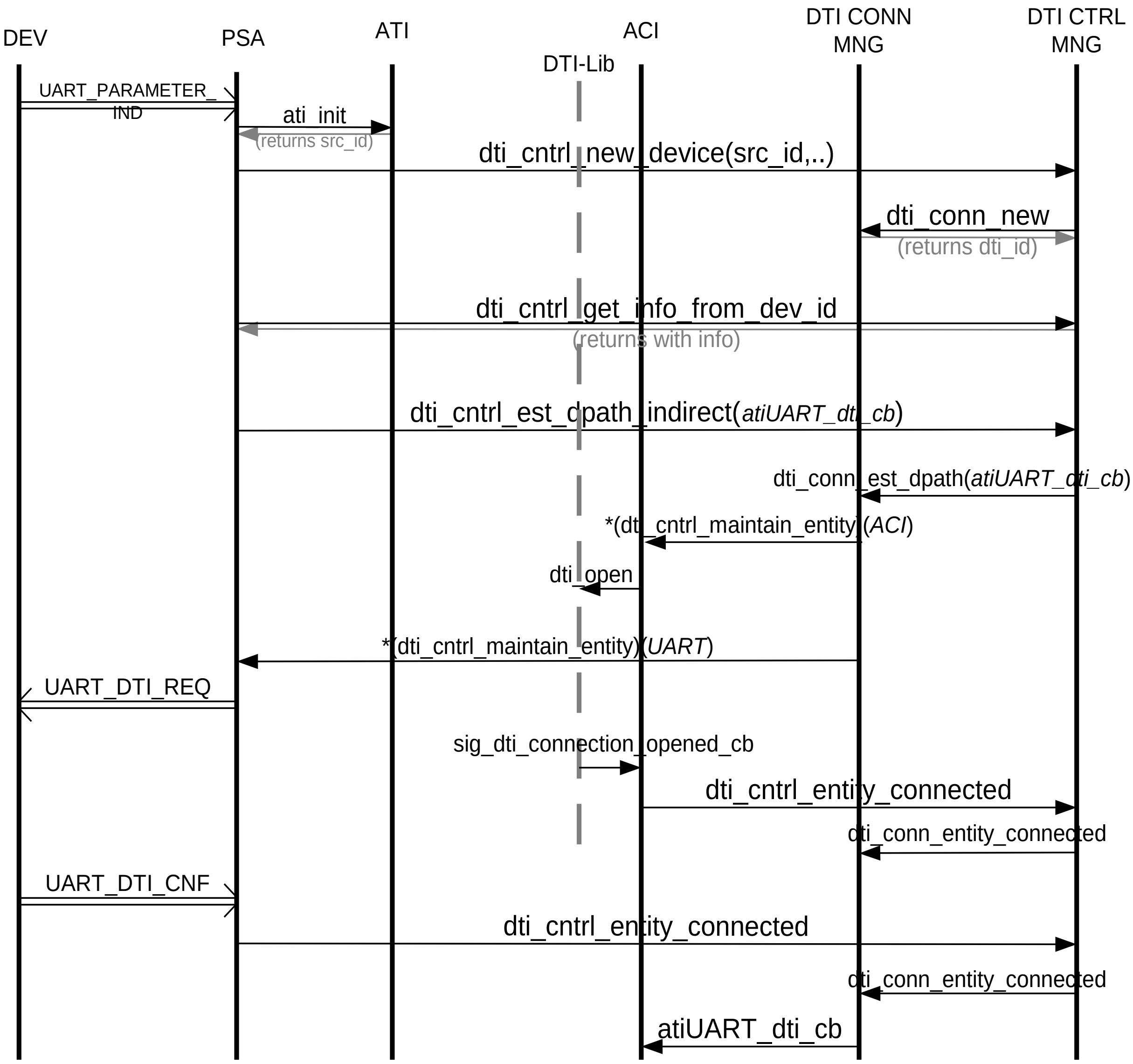
DEV
PSA
ATI
ACI
DTI CONN MNG
DTI CTRL MNG
DTI-Lib
UART_PARAMETER_IND
ati_init
(returns src_id)
dti_cntrl_new_device(src_id,..)
dti_conn_new
(returns dti_id)
dti_cntrl_get_info_from_dev_id
(returns with info)
dti_cntrl_est_dpath_indirect(atiUART_dti_cb)
dti_conn_est_dpath(atiUART_dti_cb)
*(dti_cntrl_maintain_entity)(ACI)
dti_open
*(dti_cntrl_maintain_entity)(UART)
UART_DTI_REQ
sig_dti_connection_opened_cb
dti_cntrl_entity_connected
dti_conn_entity_connected
UART_DTI_CNF
dti_cntrl_entity_connected
dti_conn_entity_connected
atiUART_dti_cb

Figure 5

Some special notes:

- due to the fact that a PSA is semantically very close to a device, the PSA knows the capability of the device. In case of a UART device, the PSA knows of the command capability. Therefore the PSA has to call “ati_init” to get the src_id.
- A PSA is informed about the capability of the device through a primitive (with the expection of the UART device). Therefore, the PSA can check the capability information for command capability. If so, then the PSA has to call “ati_init”.
- If the device is not capable for commands, then the PSA does not call “ati_init”. Rather it sets the parameter src_id of the function “dti_cntrl_new_device” to NO_SRC_ID.

Appendices

B. Acronyms

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

C. 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/>