Technical Document - Confidential

GSM PROTOCOL STACK

G23

STE – STATUS OF TERMINAL EQUIPMENT

DRIVER INTERFACE

Document Number:	8454.800.01.002
Version:	0.3
Status:	Draft
Approval Authority:	
Creation Date:	2001-Jun-19
Last changed:	2015-Mar-08 by XINTEGRA
File Name:	Ste_api.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-Jun-19	STW		0.1		1
2001-Aug-08	STW		0.2		2
2003-May-20	XINTEGRA		0.3	Draft	

Notes:

1. Initial version
2. New structure after review

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

[C_8415.0026] 8415.026.99.012; March 19, 1999
Generic Driver Interface – Functional Specification; Condat

2 Introduction

G23 is a software package implementing Layers 2 and 3 of the ETSI-defined GSM air interface signaling protocol, and as such represents the part of a GSM mobile station's protocol software which is both, platform and manufacturer independent. Therefore, G23 can be viewed as a building block providing standardized functionality through generic interfaces for easy integration.

The G23 suite of products consists of the following items:

- Layers 2 and 3 for speech & short message services,
- Layers 2 and 3 for fax & data services,
- Application Control Interface, AT Command Interface
- MMI and MMI Framework (MFW)
- Test and integration support tools.

This document describes the functional interface (API) of the driver, which provides status information of the Terminal Equipment (TE). This API of the driver is derived from the generic driver interface specification [C_8415.0026].

2.1 Data types

2.1.1 T_DRV_CB_FUNC – Driver Callback Function

Definition:

```
typedef void (*T_DRV_CB_FUNC) (T_DRV_SIGNAL * Signal) ;
```

Description:

This type defines a callback function used to signal driver events. The driver calls the signal callback function when a specific event occurs and the driver has been instructed to signal the event to the protocol stack.

The protocol stack can set or reset event signaling by calling one of the driver functions STE_SetSignal() and STE_ResetSignal(). Event signaling can only be performed when a signal callback function has been installed at driver initialization.

The signal callback has only one single parameter **`Signal`** containing all data required to identify the signal. For more information about the T_DRV_SIGNAL data type, refer to 2.1.2.

2.1.2 T_DRV_SIGNAL – Driver Signal

Definition:

```
typedef struct
{
  USHORT            SignalType
  USHORT            DataLength
  T_VOID_STRUCT *   UserData
  USHORT            DrvHandle
} T_DRV_SIGNAL
```

Description:

This type defines the signal information data used to identify a signal. **`SignalType`** must be set to one of the defined values (see STE_SetSignal()). DataLength is not used by this driver. UserData points to an USHORT value which is set to the affected **`Identifier`** (see STE_Read(), STE_Write and STE_SetSignal()). **`DrvHandle`** must be set to the value given by the STE_Init() call.

3 Interface description of the STE driver

3.1 Data types

Name	Description
UBYTE	unsigned 8 bit integer data type
BYTE	signed 8 bit integer data type
USHORT	unsigned 16 bit integer data type
SHORT	signed 16 bit integer data type
ULONG	unsigned 32 bit integer data type
LONG	signed 32 bit integer data type

3.2 Constants

Name	Description
DRV_OK	Return value indicating the function completed successfully
DRV_INITIALIZED	Driver is already initialized
DRV_INPROCESS	The requested function is currently being executed
DRV_INVALID_PARAMS	One or more parameters are out of range or invalid
DRV_SIGFCT_NOTAVAILABLE	The requested event signaling functionality is not available
DRV_INTERNAL_ERROR	Unspecified internal driver error

3.3 Functions

Name	Description
STE_Init	Initialization of the status driver
STE_Exit	De-initialization of the status driver
STE_Read	Get status of the TE
STE_Write	Set status of the TE
STE_SetSignal	Define a signal used to indicated special events
STE_ResetSignal	Un-define a signal the driver uses to indicate an event

3.3.1 STE_Init – Driver Initialization

Definition:

```
USHORT STE_Init
(
        USHORT              DrvHandle,
        T_DRV_CB_FUNC       CallbackFunc
);
```

Parameters:

Name	Description
DrvHandle	unique handle for this driver
CallbackFunc	This parameter points to the function that is called at the time an event that is to be signaled occurs. This value can be set to NULL if event signaling should not be possible.

Return values:

Name	Description
DRV_OK	Initialization successful
DRV_INTERNAL_ERROR	Internal driver error
DRV_INITIALIZED	Driver already initialized

Description

The function initializes the module and all connected devices.

The driver stores the DrvHandle and passes it in the T_DRV_SIGNAL structure of the **`Signal`** parameter to the calling process every time the callback function is called.

The function returns DRV_INITIALIZED if the driver has already been initialized and is ready to be used or is already in use.

3.3.2 STE_Exit – Termination of the driver

Definition:

```
void STE_Exit
(
);
```

Parameters:

Name	Description
-	-

Return values:

Name	Description
-	-

Description

The function is called when the driver functionality is no longer required. The function “de-allocates” the resources (interrupts, buffers, etc.). The driver terminates regardless of any outstanding data to be sent.

3.3.3 STE_Read - Get status of Terminal Equipment

Definition:

```
USHORT STE_Read
(
    USHORT      Identifier,
    ULONG*      Value
);
```

Parameters:

Name	Description
Identifier	Identifies the kind of status information.
Value	Status of Terminal Equipment that belongs to the specified **`Identifier`**.

Return values:

Name	Description
DRV_OK	Function successful
DRV_INVALID_PARAMS	The specified **`Identifier`** is not defined
DRV_INTERNAL_ERROR	Internal driver error

Description

This function returns the status of the Terminal Equipment.

If the specified **`Identifier`** is not defined then the function returns DRV_INVALID_PARAMS and remains the **`Value`** parameter unchanged.

The parameter **`Identifier`** is used in STE_Read() and in STE_Write(). The following values are defined for this parameter:

Identifier	Description
STE_IDENTIFIER_POWER	Power status of Terminal Equipment

The parameter **`Value`** is used in STE_Read() and in STE_Write(). The following values are defined for this parameter:

Identifier	Value	Description
STE_IDENTIFIER_POWER	STE_POWER_ON	the TE is powered on
	STE_POWER_OFF	the TE is powered off

3.3.4 STE_Write – Set status of Terminal Equipment

Definition:

```
USHORT STE_Write
(
        USHORT          Identifier,
        ULONG           Value,
        ULONG           Mask
);
```

Parameters:

Name	Description
Identifier	Identifies the kind of status information
Value	Status of Terminal Equipment that belongs to the specified **`Identifier`**. If **`Value`** is a bit field only the bits which are marked with the „set"-access can be used to change the state of this bit.
Mask	If **`Value`** is a bit field this parameter is a bit field with the same structure as **`Value`**. Each bit in **`Value`** corresponds to a bit in **`Mask`**. Only those status bits are manipulated by the driver, which are marked by a 1 in the **`Mask`** bit field and which are settable according to the table in the specification of STE_Read(). If **`Value`** is not a bit field **`Mask`** is not used.

Return values:

Name	Description
DRV_OK	Function successful
DRV_INVALID_PARAMS	The specified states are not setable
DRV_INPROCESS	The driver is busy setting the new states.
DRV_INTERNAL_ERROR	Internal driver error

Description

This function is used to change the states of the TE.

If one or more of the new states are not setable then the function returns DRV_INVALID_PARAMS and sets non of the new states.

In case of a successful completion, the function returns DRV_OK.

In case the driver can not change the states of the TE immediately, the function returns DRV_INPROCESS. When the states of the TE are changed the driver calls the signal callback function with the **`SignalType`** DRV_SIGTYPE_READ and the given **`Identifier`**.

The **`Identifier`** STE_IDENTIFIER_POWER is used to get or set the power state of the TE. In case of STE_Write() the **`Value`** parameter can only be set to STE_POWER_ON. The **`Mask`** parameter is not used for this **`Identifier.`**

3.3.5 STE_SetSignal – Setup a Signal

Definition:

```
USHORT STE_SetSignal
(
    USHORT          SignalType
);
```

Parameters:

Name	Description
SignalType	Signal type to be set

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	invalid signal type
DRV_INTERNAL_ERROR	Internal driver error
DRV_SIGFCT_NOTAVAILABLE	Event signaling functionality is not available

Description

This function is used to define a single or multiple signals that is/are indicated to the process when the event identified in the signal information data type as **`SignalType`** occurs. The driver uses the following signal types:

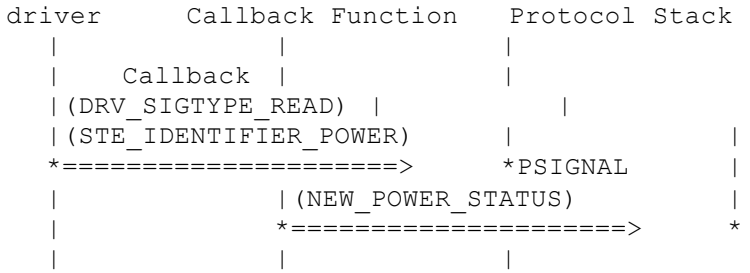
Signal	Value
DRV_SIGTYPE_READ	0x0002

To remove a signal, call the function STE_ResetSignal().

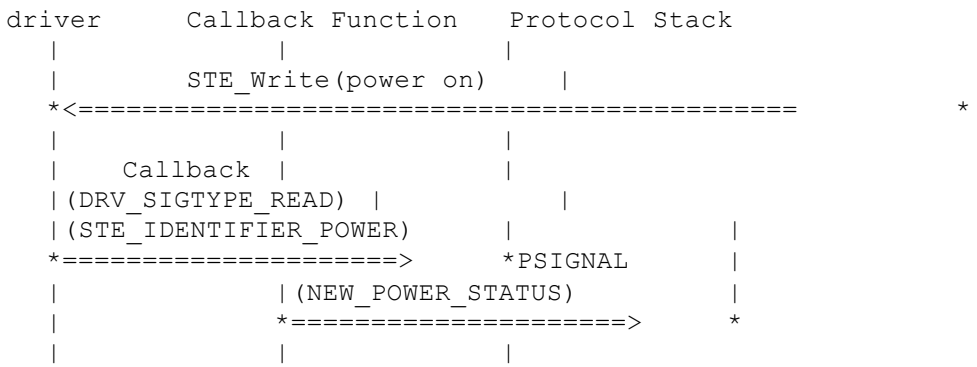
If one of the parameters of the signal information data is invalid, the function returns DRV_INVALID_PARAMS.

If no signal callback function has been defined at the time of initialization, the driver returns DRV_SIGFCT_NOTAVAILABLE.

Power: To inform the protocol stack about a change of the the power condition of the TE, the driver has to call the signal callback function, with the **`SignalType`** DRV_SIGTYPE_READ. **`UserData`** must point to a USHORT value which is set to STE_IDENTIFIER_POWER. The value **`DataLength`** of the T_DRV_SIGNAL stucture is not used. The driver should call this function every time when the TE changes the power state.



If the TE is powered on because of protocol stack initiation then the driver has also to call the signal callback function with the **`SignalType`** DRV_SIGTYPE_READ and **`UserData`** set to STE_IDENTIFIER_POWER.



3.3.6 STE_ResetSignal – Remove a Signal

Definition:

```
USHORT STE_ResetSignal
(
        USHORT              SignalType
);
```

Parameters:

Name	Description
SignalType	Signal type to be reset

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	invalid signal type
DRV_INTERNAL_ERROR	Internal driver error
DRV_SIGFCT_NOTAVAILABLE	Event signaling functionality is not available

Description

This function is used to remove previously set single or multiple signals. The signals that are removed are identified by the **`SignalType`**. If the **`SignalType`** provided cannot be located, the function returns DRV_INVALID_PARAMS.

If no signal callback function has been defined at the time of initialization, the driver returns DRV_SIGFCT_NOTAVAILABLE.

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/>