Technical Document – Confidential

GSM PROTOCOL STACK

G23

PWR – POWER

DRIVER INTERFACE

Document Number:	8415.010.99.005
Version:	0.6
Status:	Draft
Approval Authority:	
Creation Date:	1998-Sep-17
Last changed:	2015-Mar-08 by XINTEGRA
File Name:	8415_010.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
1998-Sep-17	LM et al.		0.1		1
1999-Mar-16	LM et al.		0.2		2
1999-Mar-25	MS et al.		0.3		3
1999-Jun-04	LE et al.		0.4		4
2000-Feb-04	UB et al.		0.5		5
2003-May-16	XINTEGRA		0.6	Draft	

Notes:

1. Initial version
2. Now complies with new Version of 8415.026.99.012; API changed
3. English check/New document template
4. Consistency check/Submitted
5. New template

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,
- Slim MMI [02.30] and
- Test and integration support tools.

This document describes the functional interface of the G23 power management device driver interface. This driver is used to control all power related functions, such as charger and battery control. The driver does support multiple devices and therefore no open and close functionality is supported. The driver can be configured to signal different state transitions, for example battery level has reached the "battery low" level. This is done by setting an OS signal or calling a specified call-back function.

3 Interface description of PWR driver

3.1 Data types

Name	Description
pwr_Status_Typ	Status information
pwr_DCB_Type	Device Control Block

3.1.1 pwr_Status_Typ

Definition:

```
typedef struct pwr_Status_Type
{
  UBYTE Status
  UBYTE BatteryLevel
}
```

Description:

This data type represents the driver, respectively the power unit status. The following table contains a list of elements contained in the status information data type and short descriptions of them.

Data element	Description
Status	Specifies the status of different sub-devices of the power unit (e.g. external power status, charger status). This parameter can be a combination of the following statuses as defined in 3.2.: PWR_EXTPOWER_ON, PWR_CHARGER_ON
BatteryLevel	Indicates the current battery level in units defined in the DCB.

3.1.2 pwr_DCB_Type – Device Control Block

Definition:

```
typedef struct pwr_DCB_Type
{
  UBYTE RangeMin
  UBYTE RangeMax
  UBYTE Steps
}
```

Description:

The device control block data type contains all parameters used to configure the power management driver. The following table contains a list of the data elements and short descriptions of them.

Data element	Description
RangeMin	Value representing the minimum (lowest) battery level. when this level is indicated, charging is required.
RangeMax	Value representing the maximum (highest) battery level
Steps	Value representing the number of battery levels between RangeMin and RangeMax

3.2 Constants

Name	Description
PWR_SIGTYPE_EXTPOWER	Signal type: status changes of the external power connector
PWR_SIGTPYE_CHARGER	Signal type: status changes of the charger
PWR_SIGTYPE_BATLEVEL	Signal type: capacity of the battery has changed
PWR_EXTPOWER_ON	External power available
PWR_CHARGER_ON	Battery is charged

3.3 Signals

Signals are used to inform the using process about selected events asynchronously. Signaling is done by passing a signal call-back function to the driver at the time of initialization (see "3.4.1 pwr_Init – Driver Initialization"). When no call-back is defined, event signaling cannot be performed. A signal can be set using the function pwr_SetSignal() which can be found in Chapter 3.4.3. Event signaling can be disabled by calling the function pwr_ResetSignal(), for more details on this function refer to Chapter 3.4.4.

The contents of the drv_SignalID_Type information structures specially defined for this driver are described in the following chapters. The contents of the drv_SignalID_Type information structures for the common signals are described in [1].

3.3.1 PWR_SIGTYPE_EXTPOWER_STATUS

This signal is indicated when the status of the external power sub-unit has changed, i.e. each time a transition from external power on to external power off or vice versa occurs this signal will be indicated. A prerequisite to being informed asynchronously about this event is that the signal has been set using the pwr_SetSignal() function. Aside from being informed about the status change asynchronously, the status may be polled by calling the function pwr_GetStatus() (Chapter 3.4.7).

Paramter	Value
SignalType	PWR_SIGTYPE_EXTPOWER
SignalValue	not used
UserData	Pointer to a buffer of the type pwr_Status_Type

3.3.2 PWR_SIGTYPE_CHARGER

This signal is indicated when the status of the charger power sub-unit has changed, i.e. each time a transition from charging to not charging or vice versa occurs this signal will be indicated. A prerequisite to being informed asynchronously about this event is that the signal has been set using the pwr_SetSignal() function. Aside from being informed about the status change asynchronously, the status may be polled calling the function pwr_GetStatus() (Chapter 3.4.7).

Paramter	Value
SignalType	PWR_SIGTYPE_CHARGER
SignalValue	not used
UserData	Pointer to a buffer of the type pwr_Status_Type

3.3.3 PWR_SIGTYPE_BATLEVEL

This signal is indicated when a change of the battery power level has been detected. A prerequisite to being informed asynchronously about this event is that the signal has been set using the pwr_SetSignal() function. Aside from being informed about the status change asynchronously the status may be polled by calling the function pwr_GetStatus() (Chapter 3.4.7).

Paramter	Value
SignalType	PWR_SIGTYPE_BATLEVEL
SignalValue	not used
UserData	Pointer to a buffer of the type pwr_Status_Type

3.4 Functions

Name	Description
pwr_Init	Initialization of PWR
pwr_Exit	Termination of PWR
pwr_SetSignal	Define a signal the driver uses to indicate an event
pwr_ResetSignal	Un-define a signal the driver uses to indicate an event
pwr_SetConfig	Configure the driver
pwr_GetConfig	Retrieve the configuration of the driver
pwr_GetStatus	Retrieve the capacity of the battery

3.4.1 pwr_Init – Driver Initialization

Definition:

```
UBYTE pwr_Init
(
    drv_SignalCB_Type in_SignalCBPtr
) ;
```

Parameters:

Name	Description
in_SignalCBPtr	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_INITIALIZED	Driver already initialized
DRV_INITFAILURE	Initialization failed

Description

The function initializes the driver's internal data. The function returns DRV_OK in the case of a successful completion.

The function returns DRV_INITIALIZED if the driver has already been initialized and is ready to be used or is already in use. In the case of an initialization failure, i.e. the driver cannot be used, the function returns DRV_INITFAILURE.

3.4.2 pwr_Exit – De-initialization of the driver

Definition:

```
void pwr_Exit
(
    void
) ;
```

Parameters:

Name	**Description**
-	-

Return values:

Name	**Description**
-	-

Description

This function is used to indicate PWR that the driver and its functionality are no longer needed.

3.4.3 pwr_SetSignal – Setup a Signal

Definition:

```
UBYTE pwr_SetSignal
(
        drv_SignalID_Type*      in_SignalIDPtr
) ;
```

Parameters:

Name	Description
in_SignalIDPtr	Pointer to the signal information data

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	One or more parameters are out of range or invalid
DRV_SIGFCT_NOTAVAILABLE	Event signaling functionality is not available

Description

This function is used to define a single signal or multiple signals that is/are indicated to the process when the event identified in the signal information data type as SignalType occurs.

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

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

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

None of the standard signals are supported by this driver. The only signals that are supported are described in Chapter 3.3.

3.4.4 pwr_ResetSignal – Remove a Signal

Definition:

```
UBYTE pwr_ResetSignal
(
        drv_SignalID_Type*      in_SignalIDPtr
) ;
```

Parameters:

Name	Description
in_SignalIDPtr	Pointer to the signal information data

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	One or more parameters are out of range or invalid
DRV_SIGFCT_NOTAVAILABLE	Event signaling functionality is not available

Description

This function is used to remove a single signal or multiple signals that have previously been set. The signals that are removed are identified by the Signal Information Data element, "SignalType". All other elements of the Signal Information Data must be identical to the signal(s) that is/are to be removed (process handle and signal value). If the SignalID provided cannot be found, the function returns DRV_INVALID_PARAMS.

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

None of the standard signals are supported by this driver. The only signals that are supported are described in Chapter 3.3.

3.4.5 pwr_SetConfig – Set a driver configuration

Definition:

```
UBYTE pwr_SetConfig
(
    pwr_DCB_Type*          in_DCBPtr
) ;
```

Parameters:

Name	Description
in_DCBPtr	Pointer to the driver control block

Return values:

Name	Description
DRV_OK	Function successfully completed
DRV_INVALID_PARAMS	One or more values are out of range or invalid in that combination

Description

This function is used to configure the driver. For detailed information about the contents of the driver control block, refer to Chapter 3.1.2.

If any value of the configuration is out of range or invalid in combination with any other value of the configuration, the function returns DRV_INVALID_PARAMS.

Call the pwr_GetConfig() function to retrieve the driver's configuration.

3.4.6 pwr_GetConfig – Retrieve the driver configuration

Definition:

```
UBYTE pwr_GetConfig
(
    pwr_DCB_Type*          out_DCBPtr
) ;
```

Parameters:

Name	Description
out_DCBPtr	Pointer to the driver control block

Return values:

Name	Description
DRV_OK	Function successfully completed
DRV_NOTCONFIGURED	The driver is not yet configured

Description

This function is used to retrieve the configuration of the driver. The configuration is returned in the driver control block to which the pointer provided out_DCBPtr points. For detailed information about the contents of the driver control block, refer to Chapter 3.1.2.

If the driver is not configured, the function returns DRV_NOTCONFIGURED.

Call the pwr_SetConfig() function to configure the driver.

3.4.7 pwr_GetStatus – Retrieve the Driver Status

Definition:

```
UBYTE pwr_GetStatus
(
    pwr_Status_Type*        out_StatusPtr

);
```

Parameters:

Name	Description
out_StatusPtr	Pointer to the status information buffer

Return values:

Name	Description
DRV_OK	Function successfully completed
DRV_NOTCONFIGURED	The driver is not yet configured
DRV_INVALID_PARAMS	out_StatusPtr is NULL

Description

This function is used to retrieve the status of the driver respectively the power unit.

In the case of a successful completion, the driver returns DRV_OK and the current status of the driver to which the buffer out_StatusPtr points.

If the driver is not yet configured, it returns DRV_NOTCONFIGURED. In this case, the contents of the buffer out_StatusPtr is invalid.

If out_StatusPtr equals NULL, the driver returns DRV_INVALID_PARARMS.

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/>