Technical Document - Confidential

GSM PROTOCOL STACK

G23

TM – HARDWARE TIMER DRIVER

DRIVER INTERFACE

Document Number:	8415.065.99.004
Version:	0.5
Status:	Draft
Approval Authority:	
Creation Date:	1999-Nov-12
Last changed:	2015-Mar-08 by XINTEGRA
File Name:	Timer_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
1999-Nov-12	TSE		0.1		1
1999-Dec-06	TSE		0.2		2
1999-Dec-13	TSE		0.3		3
2000-Jan-21	TSE		0.4		4
2003-May-21	XINTEGRA		0.5	Draft	

Notes:

1. Initial version
2. Restructured
3. Function parameters updated
4. TM_SIGTYPE_ELAPSED updated

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 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 of the G23 hardware timer driver. This driver is used to control hardware timer registers (i.e. for watchdogs, sounds..) which could generate interrupts.

Interface description of the Timer driver

1.1 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

Constants

Name	Description
TM_SIGTYPE_ELAPSED	Signal type identifying that a timer has elapsed
DRV_BUFFER_FULL	The internal buffer is exhausted
DRV_DISABLED	Driver is not enabled
DRV_ENABLED	Driver is enabled
DRV_NOTCONFIGURED	Driver is not configured
DRV_INITFAILURE	Driver initialization failed
DRV_INITIALIZED	Driver is already initialized
DRV_INTERNAL_ERROR	Unspecified internal driver error
DRV_INPROCESS	The requested function is currently being executed
DRV_INVALID_PARAMS	One or more parameters are out of range or invalid
DRV_NOTCONFIGURED	Driver is not configured
DRV_OK	Return value indicating the function completed successfully
DRV_UNKNOWN	unknown device accessed
DRV_SIGFCT_NOTAVAILABLE	The requested event signaling functionality is not available

1.2 Signals

Signals are used to asynchronously inform the process using the services about selected events. Signaling is done by passing a signal call-back function to the driver at the time of initialization . When no call-back is defined, event signaling cannot be performed. A signal can be set using the function TM_SetSignal(). Event signaling can be disabled by calling the function TM_ResetSignal().

The following chapters describe the contents of the T_DRV_SIGNAL information structures defined specially for this driver. The contents of the T_DRV_SIGNAL information structures for the common signals are defined in [C_8415.0026].

1.2.1 TM_SIGTYPE_ELAPSED

This signal is indicated when the Timer has elapsed. A prerequisite to being informed asynchronously about this event is that the signal has been set using the TM_SetSignal() function. The low byte of the SignalValue parameter includes the ID of the timer that has elapsed. The high byte of the SignalValue parameter identifies the use of the parameter UserData.

Parameter	Value
SignalType	TM_SIGTYPE_ELAPSED
DataLength	6
UserData	1st 16 bit: LowByte: Timer ID HighByte : 0 ... Data represent timer expirations 3rd Byte ff: Counter of timer expirations in case of autoreload (4 byte)

1.3 Functions

Name	Description
TM_Init	Initialization of the driver
TM_Exit	Termination of the driver
TM_ResetTimer	Reset timer
TM_StopTimer	Stop timer
TM_StartTimer	Start timer
TM_ReadTimer	Read timer value
TM_DisableWatchdog	Disable watchdog interrupt
TM_EnableWatchdog	Enable watchdog interrupt
TM_ResetWatchdog	Reset Watchdog
TM_SetSignal	Define a signal which the driver uses to indicate an event
TM_ResetSignal	Un-define a signal which the driver uses to indicate an event

1.3.1 TM_Init – Driver Initialization

Definition:

```
USHORT TM_Init
(
    USHORT              DrvHandle
    T_DRV_CB_FUNC       in_SignalCBPtr
    T_DRV_EXPORT **     DrvInfo
) ;
```

Parameters:

Name	Description
DrvHandle	unique handle for this driver
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.
DrvInfo	pointer to the driver parameters (see GDI specification document for a description of T_DRV_EXPORT).

Return values:

Name	Description
DRV_OK	Initialization successful
DRV_INITIALIZED	Driver already initialized
DRV_INITFAILURE	Initialization failed

Description

This function needs to be implemented in all drivers.

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 case of an initialization failure, which means that the driver cannot be used, the function returns DRV_INITFAILURE.

The driver exports its properties like its name, the functions to access driver functionality and a bitfield called flags by the parameter DrvInfo. If the driver is called by ISR, Bit (0) in the bitfield is set, otherwise this bit is cleared.

The driver stores the DrvHandle and passes it over the SignalID to the calling process every time the callback function is called.

1.3.2 TM_Exit – Driver finalization

Definition:

```
void TM_Exit
(
    void
) ;
```

Parameters:

Name	Description
-	

Return values:

Name	Description
-	-

Description

This function needs to be implemented in all drivers.

The function is called when the driver functionality is no longer needed. The function “de-allocates” all allocated resources and finalizes the driver.

1.3.3 TM_ResetTimer - Reset Timer

Definition:

```
void  TM_ResetTimer (
          int          timerNum,
          int          countValue,
          USHORT       autoReload,
          USHORT       clockScale
                  ) ;
```

Parameters:

Name	Description
timerNum	timer ID (in case of multiple hardware timers, defined by driver)
countValue	initial value of timer (in ticks)
autoReload	reload timer after expiration (1=yes, 0=no)?
ClockScale	timer tick scale (timer prescaler value) : ratio of clock cycles per tick (1 tick = multiple of milliseconds)

Return values:

Name	Description
-	-

Description

This function sets the settings for the timer . The countValue is a multiple of timer ticks, so the timer duration in Seconds must be computed according to the timer prescaler. If autoreload is set, the timer re-starts after expiration

automatically with the value countValue. If the timer expires and the timer's interrupt service routine resides within the driver, the event specified by the TM_SIGTYPE_ELAPSED Signal is generated an the callback routine is called. If the timer interrupt is connected with an ISR outside the driver, that ISR becomes running and no event will be generated.

1.3.4 TM_StopTimer - Stop Timer

Definition:

```
void  TM_StopTimer (
          int       timerNum);
```

Parameters:

Name	Description
timerNum	timer ID

Return values:

Name	Description
-	-

Description

This function stops the timer immediately. The current timer value will not be changed, until the timer is started again.

1.3.5 TM_StartTimer – Start Timer

Definition:

```
void  TM_StartTimer (
          int       timerNum);
```

Parameters:

Name	Description
timerNum	timer ID

Return values:

Name	Description
-	-

Description

This function starts the timer immediately using the current timer value .

1.3.6 TM_ReadTimer – Read Current Timer Value

Definition:

```
USHORT TM_ReadTimer (
        int     timerNum)
```

Parameters:

Name	Description
timerNum	timer ID

Return values:

Name	Description
-	current timer value in ticks

Description

This function reads the current timer value.

1.3.7 TM_DisableWatchdog - Disable watchdog interrupt

Definition:

void TM_DisableWatchdog(void);

Parameters:

Name	Description
-	

Return values:

Name	Description
-	

Description

This function disables a watchdog which is build of a hardware timer. The watchdog interrupt will never be generated, until TM_EnableWatchdog() is called.

1.3.8 TM_EnableWatchdog - Enable watchdog interrupt

Definition:

void TM_EnableWatchdog(void);

Parameters:

Name	Description
-	

Return values:

Name	Description
-	

Description

This function enables a watchdog which is build of a hardware timer (the timer's interrupt should be connected to a RESET pin of the MCU). The watchdog interrupt is enabled. In case of the watchdog interrupt no event could be generated.

TM_ResetWatchdog - Reset watchdog

Definition:

```
void TM_ResetWatchdog(
        int     count);
```

Parameters:

Name	Description
count	new watchdog value in ticks

Return values:

Name	Description

-

Description

This function resets a watchdog to avoid watchdog interrupt. The watchdog timer duration is defined by the parameter Count.

TM_SetSignal - Define a signal which the driver uses to indicate an event

Definition:

```
USHORT TM_SetSignal
(
        USHORT              in_SignalID
) ;
```

Parameters:

Name	Description
in_SignalID	Signal type to be set

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 signal that indicates keyboard status changes to the process. A keyboard status change is an event identified in the signal information data type as SignalType. The only signal that can be set is the signal defined in the following table.

Signal	Value
TM_SIGTYPE_ELAPSED	DRV_SIGTYPE_USER

1.3.9 TM_ResetSignal – Remove a Signal

Definition:

```
USHORT TM_ResetSignal
(
        USHORT                  in_SignalID
) ;
```

Parameters:

Name	Description
in_SignalID	Signal type to be reset

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 signal that has previously been set. The signal that is removed is identified by in_SignalID.

If no signal call-back 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/>