Technical Document - Confidential

GSM PROTOCOL STACK

G23

AUDIO-AUDIO DRIVER

DRIVER INTERFACE

Document Number:	8415.008.99.006
Version:	0.7
Status:	Draft
Approval Authority:	
Creation Date:	1998-Sep-15
Last changed:	2015-Mar-08 by XGUTTEFE
File Name:	8415_008.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
1998-Sep-15	LM et al.		0.1		1
1999-Mar-10	LM et al.		0.2		2
1999-Mar-15	MS et al.		0.3		3
1999-Jun-4	LE et al.		0.4		4
1999-Jul-5	SAB et al.		0.5		5
2000-Feb-4	UB et al.		0.6		6
2003-May-13	XINTEGRA		0.7		

Notes:

1. Initial version
2. Now complies with new Version of 8415.026.99.012/several new functions/functions removed: Enable, IsEnabled, GetAmplf, IsCNoise, CNoise
3. Format/English check
4. Consistency check/Submitted
5. Add function_GetAmplf
6. New template

Table of Contents

1.1 References 4

2 Introduction 5

3 Interface description of the AUDIO driver 5

3.1 Data types 5
3.1.1 audio_Status_Typ 5
3.2 Constants 5
3.3 Signals 7
3.3.1 AUDIO_SIGTYPE_SOUNDEND 7
3.4 Functions 8
3.4.1 audio_Init – Driver initialization 9
3.4.2 audio_Exit – Driver finalization 10
3.4.3 audio_SetMute – Set an audio device into mute mode 11
3.4.4 audio_GetMute – Get status whether a device is muted 12
3.4.5 audio_GetSoundImage - Retrieve the image of a driver-specific sound/melody 13
3.4.6 audio_SetAmplf – Set amplification value 14
3.4.7 audio_GetAmplf – Get amplification value 15
3.4.8 audio_PlaySoundID – Play driver-specific sound or melody 16
3.4.9 audio_PlaySoundbyImage – Play driver-specific sound or melody 17
3.4.10 audio_StopSoundbyID – Stop playing a sound or melody 18
3.4.11 audio_StopSoundbyImage – Stop playing a sound or melody 19
3.4.12 audio_SetSignal – Setup a signal 20
3.4.13 audio_ResetSignal – Remove a signal 21
3.4.14 audio_GetStatus – Retrieve the Driver Status 22

Appendices 23

A. Acronyms 23
B. Glossary 23

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 audio driver interface. This driver is used to control all audio and audio-related devices of the mobile such as speaker, microphone, etc. A device, e.g. a speaker, may have the capability to play sounds and/or melodies indirectly, i.e. a sound generator may be attached to a speaker or its amplifier. The format of the sound/melody images is implementation dependent and therefore not in the scope of this document. Therefore an application may play predefined, driver specific sounds identifying them via a sound ID. In addition, the API of this driver is equipped so that an application may copy a sound image into its local buffer, modify it and have it play via the driver. In this case, the application must have the knowledge about the sound image format.

3 Interface description of the AUDIO driver

3.1 Data types

Name	Description
audio_Status_Typ	Status Information

3.1.1 audio_Status_Typ

Definition:

```
typedef struct audio_Status_Type
{
  UBYTE min_volume;
  UBYTE max_volume;
}
```

Description:

This data type represents the driver characteristics with respect to the minimum and maximum values for volume setting. The following table contains a list of elements contained in the status information data type and short descriptions of them.

Data element	Description
min_volume	Minimum volume of the audio driver.
max_volume	Maximum volume of the audio driver.

3.2 Constants

Name	Description

AUDIO_INT_SPEAKER	Device identifier for the internal speaker
AUDIO_INT_MIC	Device identifier for the internal microphone
AUDIO_EXT_SPEAKER	Device identifier for the external speaker
AUDIO_EXT_MIC	Device identifier for the external microphone
AUDIO_BUZZER	Device identifier for the internal buzzer
AUDIO_SIGTYPE_SOUNDEND	Signal type identifying that a sound has reached its end
AUDIO_FCT_NOTSUPPORTED	Function is not supported by this driver
AUDIO_MUTING_ON	Device is muted
AUDIO_MUTING_OFF	Device is not muted
AUDIO_MIN_VOLUME	Minimum volume of the audio speaker.
AUDIO_MAX_VOLUME	Maximum volume of the audio speaker.
AUDIO_MIN_BUZ_VOLUME	Minimum volume of the audio buzzer.
AUDIO_MAX_BUZ_VOLUME	Maximum volume of the audio buzzer.
AUDIO_MIN_MIC_VOLUME	Minimum volume of the audio microphone.
AUDIO_MAX_MIC_VOLUME	Maximum volume of the audio microphone.

3.3 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 (see "3.4.1 audio_Init – Driver initialization"). When no call-back is defined, event signaling cannot be performed. A signal can be set using the function audio_SetSignal() – see Chapter 3.4.12. Event signaling can be disabled by calling the function audio_ResetSignal(), for more details on this function - refer to Chapter 3.4.13.

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

3.3.1 AUDIO_SIGTYPE_SOUNDEND

This signal is indicated when the driver has stopped playing a sound/melody. A prerequisite to being informed asynchronously about this event is that the signal has been set using the audio_SetSignal() function. The low byte of the SignalValue parameter includes the ID of the device that has played the sound. The high byte of the SignalValue parameter identifies the use of the parameter UserData.

Parameter	Value
SignalType	AUDIO_SIGTYPE_SOUNDEND
SignalValue	LowByte: DeviceID HighByte: Sound identification type 0 = UserData contains sound identifier 1 = UserData contains address of the sound image
UserData	Sound identifier or address of the sound image depending on the value of the high byte in the SignalValue

3.4 Functions

Name	Description
audio_Init	Initialization of EMI
audio_Exit	Termination of EMI
audio_SetMute	Muting function
audio_GetMute	Get state of muting
audio_GetSoundImage	Retrieve the image of a driver-specific sound/melody
audio_PlaySoundID	Start playing a driver-specific melody/sound
audio_PlaySoundbyImage	Start playing a sound/melody image
audio_StopSoundbyID	Stop playing the current melody/sound
audio_StopSoundbyImage	Stop playing the current melody/sound
audio_SetAmpIf	Set the amplification value of the device
audio_GetAmpIf	Get the amplification value of the device
audio_SetSignal	Define a signal which the driver uses to indicate an event
audio_ResetSignal	Un-define a signal which the driver uses to indicate an event
audio_GetStatus	Get the actual status of the audio driver

3.4.1 audio_Init – Driver initialization

Definition:

```
UBYTE audio_Init
(
    drv_SignalCB_Type in_SignalCBPtr
) ;
```

Parameters:

Name	Description
in_SignalCBPtr	This parameter points to the function 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 audio_Exit – Driver finalization

Definition:

```
void audio_Exit
(
    void
) ;
```

Parameters:

Name	Description
-	-

Return values:

Name	Description
-	-

Description

The function is called when the driver functionality is no longer required. The function "de-allocates" all allocated resources and finalizes the driver.

3.4.3 audio_SetMute – Set an audio device into mute mode

Definition:

```
UBYTE audio_SetMute
(
    UBYTE           in_DeviceID
    UBYTE           in_Mode
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
in_Mode	Used to enable muting (AUDIO_MUTING_ON) or disable (AUDIO_MUTING_OFF)

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device unknown or function not support by this device

Description

This function is used to enable or disable muting for the device identified by the parameter in_Device.

If the mode for the specified device could be changed, the function returns DRV_OK.

If the specified device is unknown or the device does not support muting, the function returns DRV_INVALID_PARAMS.

3.4.4 audio_GetMute – Get status whether a device is muted

Definition:

```
UBYTE audio_GetMute
(
    UBYTE           in_DeviceID
    UBYTE *         out_Mode
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
out_Mode	Muting is enabled for device (AUDIO_MUTING_ON) or disabled (AUDIO_MUTING_OFF)

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device unknown or function not support by this device

Description

This function is used to get the status whether muting is enabled or disabled for the device identified by the parameter in_Device.

If the specified device is unknown or the device does not support muting, the function returns DRV_INVALID_PARAMS.

3.4.5 audio_GetSoundImage - Retrieve the image of a driver-specific sound/melody

Definition:

```
UBYTE audio_GetSoundImage
(
    UBYTE           in_SoundID,
    UBYTE *         out_SoundImagePtr
) ;
```

Parameters:

Name	Description
in_SoundID	Identifies the sound/melody to be played
out_SoundImagePtr	Address of the buffer to which the sound image is copied

Return values:

Name	Description
DRV_OK	Configuration successful
DRV_INVALID_PARAMS	Font not available
AUDIO_FCT_NOTSUPPORTED	Function is not supported by this driver

Description

This function is used to copy the image of a driver internal sound image into an application specific sound image buffer. The application may modify the sound.

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

If the size of the buffer wherein the sound image is to be copied is too small, the driver returns DRV_INVALID_PARAMS.

If a specific driver implementation does not support this functionality, the driver returns DSPL_FCT_NOTSUPPORTED.

3.4.6 audio_SetAmpIf – Set amplification value

Definition:

```
UBYTE audio_SetAmpIf
(
    UBYTE           in_DeviceID
    UBYTE           in_AmpIf
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
in_AmpIf	Absolute amplification value

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device unknown or function not support by this device

Description

This function is used to set the amplification for the device identified by the parameter in_DeviceID.

In the case of a speaker, this is the volume for a microphone - the pre-amplifier that regulates the sensitivity of the microphone. The valid range depends on the hardware used.

If the amplification could be changed for the specified device, the function returns DRV_OK.

If the amplification value (in_AmpIf) is out of range or the specified device is unknown or the specified device does not support the amplification setting, the function returns DRV_INVALID_PARAMS.

3.4.7 audio_GetAmpIf – Get amplification value

Definition:

```
UBYTE audio_GetAmpIf
(
    UBYTE           in_DeviceID
    UBYTE *         out_AmpIf
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
out_AmpIf	Absolute amplification value

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device unknown or function not support by this device

Description

This function is used to get the amplification for the device identified by the parameter in_DeviceID.

In the case of a speaker, this is the volume for a microphone - the pre-amplifier that regulates the sensitivity of the microphone. The range depends on the hardware used.

If the specified device is unknown the function returns DRV_INVALID_PARAMS.

3.4.8 audio_PlaySoundID – Play driver-specific sound or melody

Definition:

```
UBYTE audio_PlaySoundID
(
    UBYTE       in_DeviceID,
    UBYTE       in_SoundID,
    BYTE        in_RelVolume,
    UBYTE       in_Repeats
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
in_SoundID	Identifies the sound/melody to be played
in_RelVolume	Volume relative to the basic volume set for the device (audio_SetAmplf)
in_Repeats	Identifies the number of repetitions of the sound (0 = endless)

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device unknown or function not support by this device
DRV_INPROCESS	The requested sound is currently being played

Description

This function is used to play a sound or melody. The function returns immediately after the “play process” has been activated. It is implementation-dependent if the driver/device supports playing multiple sounds simultaneously, i.e. accepting multiple calls of audio_PlaySound(). If the calling process should be notified when the sound has stopped playing automatically, the signal AUDIO_SIGTYPE_SOUNDEND must be set using the audio_SetSignal() function.

If the special driver implementation or the device does not support volume control, the driver ignores the value.

If the sound can be played, the function returns DRV_OK.

If the device is currently playing the sound identified by the parameter in_SoundID, the function returns DRV_INPROCESS.

If the driver/device is currently playing a sound, but does not support playing multiple sounds simultaneously, the driver returns DRV_INPROCESS. The driver **does not** stop the current sound and start playing the new sound. Instead, the process using the services handles this situation and may stop the playing of the current sound by calling the function audio_StopSound() or may also queue the sound requests.

If the device specified does not support playing sounds or any other value is out of range, the function returns DRV_INVALID_PARAMS.

3.4.9 audio_PlaySoundbyImage – Play driver-specific sound or melody

Definition:

```
UBYTE audio_PlaySoundbyImage
(
    UBYTE       in_DeviceID,
    void *      in_SoundImagePtr,
    BYTE        in_RelVolume,
    UBYTE       in_Repeats
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
in_SoundImagePtr	Address of the buffer containing the sound image
in_RelVolume	Volume relative to the basic volume set for the device (audio_SetAmplf)
in_Repeats	Identifies the number of repetitions of the sound (0 = endless)

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device unknown or function not support by this device
DRV_INPROCESS	The requested sound is currently being played

Description

This function is used to play a sound or melody. The image of the sound/melody is passed to the driver (the sound image format is implementation-dependent).

The function returns immediately after the “play process” has been activated. It is implementation dependent if the driver/device supports playing multiple sound simultaneously, i.e. accepting multiple calls of audio_PlaySound(). If the calling process should be notified when the sound has stopped playing automatically, the signal AUDIO_SIGTYPE_SOUNDEND must be set using the audio_SetSignal() function.

If the special driver implementation or the device does not support volume control, the driver ignores the value.

If the sound can be played, the function returns DRV_OK.

If the device is currently playing the same sound image that the parameter in_SoundImagePtr addresses, the function returns DRV_INPROCESS.

If the driver/device is currently playing a sound, but does not support playing multiple sounds simultaneously, the driver returns DRV_INPROCESS. The driver **does not** stop the current sound and start playing the new sound. Instead, the process using the services handles this situation and may stop playing the current sound by calling the function audio_StopSound() or may also queue the sound requests.

If the device specified does not support playing sounds or any other value is out of range, the function returns DRV_INVALID_PARAMS.

3.4.10 audio_StopSoundbyID – Stop playing a sound or melody

Definition:

```
UBYTE audio_StopSoundbyID
(
    UBYTE           in_DeviceID,
    UBYTE           in_SoundID
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
in_SoundID	Identifies the sound/melody to be stopped

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device/Image unknown or function not supported by this device

Description

This function is used to manually stop playing a sound/melody. When a sound is stopped manually, no signal is created as to whether or not the signal AUDIO_SIGTYP_SOUNDEND has been defined.

If the function could stop playing the specified sound, the function returns DRV_OK.

If the device is unknown or does not support this function or the specified sound ID is invalid, the function returns DRV_INVALID_PARAMS.

3.4.11 audio_StopSoundbyImage – Stop playing a sound or melody

Definition:

```
UBYTE audio_StopSoundbyImage
(
    UBYTE           in_DeviceID,
    void*           in_SoundImagePtr,
) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
in_SoundImagePtr	Address of the buffer containing the sound image

Return values:

Name	Description
DRV_OK	Function completed successfully
DRV_INVALID_PARAMS	Device/Image unknown or function not support by this device

Description

This function is used to manually stop playing a sound/melody. When a sound is stopped manually, no signal is created as to whether or not the signal AUDIO_SIGTYP_SOUNDEND has been defined.

If the function could stop playing the specified sound image, the function returns DRV_OK.

If the device is unknown or does not support this function or the specified image cannot be identified, the function returns DRV_INVALID_PARAMS.

3.4.12 audio_SetSignal – Setup a signal

Definition:

```
UBYTE audio_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 out of range or invalid
DRV_SIGFCT_NOTAVAILABLE	Event signaling functionality not available

Description

This function must be implemented in all communication drivers.

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 audio_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 supported are described in Chapter 3.3.

3.4.13 audio_ResetSignal – Remove a signal

Definition:

```
UBYTE audio_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 out of range or invalid
DRV_SIGFCT_NOTAVAILABLE	Event signaling functionality not available

Description

This function is used to remove a previously set single or multiple signals. 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 supported are described in Chapter 3.3.

3.4.14 audio_GetStatus – Retrieve the Driver Status

Definition:

```
UBYTE audio_GetStatus
(
    UBYTE                in_DeviceID,
    audio_Status_Type*   out_StatusPtr

) ;
```

Parameters:

Name	Description
in_DeviceID	Device identifier
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 of a specific device respectively the minimum and maximum values for volume control.

The minimum and maximum values represent the physical characteristics of the hardware drivers and therefore should only be changed when hardware drivers will be replaced (not doing so dynamically by a function audio_SetStatus).

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 or device is unknown, 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/>