Technical Document – Confidential

GSM PROTOCOL STACK

G23

SER – SERIAL INTERFACE DRIVER

DRIVER INTERFACE

Document Number:	8415.062.99.004
Version:	0.5
Status:	Draft
Approval Authority:	
Creation Date:	1999-Nov-22
Last changed:	2015-Mar-08 by XINTEGRA
File Name:	ser_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-22	TSE		0.1		1
1999-Dec-07	TSE		0.2		2
1999-Dec-13	TSE		0.3		3
1999-Dec-14	TSE		0.4		4
2003-May-20	XINTEGRA		0.5	Draft	

Notes:

1. Initial version

2. Corrections
3. Function parameters updated
4. Function parameters updated/function SER_SetDevice added

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 of the G23 Standard Serial Receiver Transmitter (SER) driver, API. This driver can be used for common communication purposes and includes UART, USART and IRDA devices. This API of the driver is derived from the generic driver interface specification [C_8415.0026].

NOTE: The driver devices need to be configured after initialization. Only the following functions can be called while the driver is not configured: SER_Clear, SER_SetSignal , SER_SetDevice and SER_GetSignal. All other functions return DRV_NOTCONFIGURED. After the Initialization of the driver the communication device must be chosen via the SER_SetDevice() function, after that the device has to be configured with the SER_SetConfig() function.

3 Interface description of the SER 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
T_SER_DCB	Device Control Block

3.1.1 T_SER_DCB – Device Control Block

Definition:

```
typedef struct T_SER_DCB
{
  UBYTE Port
  USHORT       Baud
  UBYTE DataBits
  UBYTE StopBits
  UBYTE Parity
  UBYTE FlowControl
  USHORT       RxBufferSize
  USHORT       TxBufferSize
};
```

Description:

The device control block data type contains all parameters used to configure the serial device. The following table contains a list of the data elements and brief descriptions of them.

The parameter Port is used to configure which of the available I/O ports is used with this driver in case more than one port is available.

Data element	Description
Baud	Transmission rate in bits/sec. See the following corresponding table.
DataBits	Size of a character (5 to 8 bits are possible, refer to the following corresponding table).
StopBits	Number of stop bits attached to each character. The following corresponding table shows possible values.
Parity	Type Parity checking. The following corresponding table shows the possible values.
FlowControl	Type of flow control. The following corresponding table shows the possible values.
RxBufferSize	Size of the receiver buffer in bytes.
TxBufferSize	Size of the transmitter buffer in bytes.

Possible transmission rate
SER_BAUD_115200
SER_BAUD_57600
SER_BAUD_38400
SER_BAUD_33900
SER_BAUD_28800
SER_BAUD_19200
SER_BAUD_14400
SER_BAUD_9600
SER_BAUD_4800
SER_BAUD_2400
SER_BAUD_1200
SER_BAUD_600
SER_BAUD_300
SER_BAUD_150
SER_BAUD_75

Table 1

Data bits
SER_CHAR5
SER_CHAR6
SER_CHAR7
SER_CHAR8

Table 2

Stop bits
SER_STOP1
SER_STOP15
SER_STOP2

Table 3

Flow Controls
SER_FLOWNO
SER_FLOWHW
SER_FLOWSW

Table 4

Parity
SER_PARITYNO
SER_PARITYODD
SER_PARITYEVEN

Table 5

3.2 Constants

Name	Description
SER_BAUD_115200	Transmission rate of 11520Kbit/s
SER_BAUD_57600	Transmission rate of 57600Kbit/s
SER_BAUD_38400	Transmission rate of 38400Kbit/s
SER_BAUD_33900	Transmission rate of 33900Kbit/s
SER_BAUD_28800	Transmission rate of 28800Kbit/s
SER_BAUD_19200	Transmission rate of 19200Kbit/s
SER_BAUD_14400	Transmission rate of 14400Kbit/s
SER_BAUD_9600	Transmission rate of 9600Kbit/s
SER_BAUD_4800	Transmission rate of 4800Kbit/s
SER_BAUD_2400	Transmission rate of 2400Kbit/s
SER_BAUD_1200	Transmission rate of 1200Kbit/s
SER_BAUD_600	Transmission rate of 600Kbit/s
SER_BAUD_300	Transmission rate of 300bit/s
SER_BAUD_150	Transmission rate of 150Kbit/s
SER_BAUD_75	Transmission rate of 75Kbit/s
SER_PARITYNO	no parity
SER_PARITYODD	odd parity
SER_PARITYEVEN	even parity
SER_CHAR5	5-Bit character
SER_CHAR6	6-Bit character
SER_CHAR7	7-Bit character
SER_CHAR8	8-Bit character
SER_STOP1	One stop bit indicates the end of a character
SER_STOP15	One and a half stop bits indicates the end of a character
SER_STOP2	Two stop bits indicate the end of a character
SER_FLOWNO	No flow control takes place (unsecured transmission)
SER_FLOWHW	Hardware handshake takes place
SER_FLOWSW	Software handshake takes place (used for 3-wire interfaces)
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
DEV_UART	UART Device ID
DEV_USART	USART Device ID
DEV_IRDA	IRDA Device ID

3.3 Functions

Name	Description
SER_Init	Initialization of the serial communication driver
SER_Exit	De-initialization of serial communication driver
SER_Clear	Re-initialize all buffers
SER_Write	Send data via the serial interface
SER_Read	Read data received via the serial interface
SER_Look	Read data received via the serial interface, but leave data unchanged
SER_Flush	Flush all buffers
SER_SetSignal	Define a signal used to indicated special events
SER_ResetSignal	Un-define a signal the driver uses to indicate an event
SER_SetConfig	Set a device configuration
SER_GetConfig	Retrieve a device configuration
SER_Callback	Callback entry for the driver
SER_SetDevice	Set communication device (UART, USART, IRDA)

3.3.1 SER_Init – Driver Initialization

Definition:

```
USHORT SER_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_INITFAILURE	Initialization failed
DRV_INITIALIZED	Driver already initialized

Description

The function initializes the module and all connected serial devices.

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.

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

The driver uses a default device and a default configuration until the functions SER_SetDevice() and SER_SetConfig() have been called.

3.3.2 SER_Exit – Termination of the driver

Definition:

```
void SER_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” the resources (interrupts, buffers, etc.). The driver terminates regardless of any outstanding data to be sent.

3.3.3 SER_Read - Read data from the driver

Definition:

```
USHORT SER_Read
(
        void*           out_BufferPtr,
        USHORT*         thr_BufferSizePtr
) ;
```

Parameters:

Name	Description
out_BufferPtr	This parameter points to the buffer wherein the data is to be copied
thr_BufferSizePtr	On call: number of characters to read. If the function returns DRV_OK, it contains the number of characters read. If the function returns DRV_INPROCESS, it contains 0.

Return values:

Name	Description
DRV_OK	Function successful
DRV_INPROCESS	The driver is currently reading data. The data is incomplete.
DRV_NOTCONFIGURED	The driver is not yet configured

Description

This function is used to read data from the driver. The data is copied into the buffer to which out_BufferPtr points. The parameter *thr_BufferSizePtr contains the size of the buffer in characters.

In the case of a successful completion, the driver's buffer is cleared. The driver keeps the data available when calling the function *drv*_Look().

If the driver is not configured, the function returns DRV_ NOTCONFIGURED.

NOTE: When calling the function with a buffer size of 0, the function will return DRV_OK. The size of the buffer needed to store the available data is stored in the parameter *thr_BufferSizePtr. In this case, the out_BufferPtr can be set to NULL.

3.3.4 SER_Write – Transmit data via the serial interface

Definition:

```
USHORT SER_Write
(
        void*           in_BufferPtr,
        USHORT*         thr_BufferSizePtr
) ;
```

Parameters:

Name	Description
in_BufferPtr	This parameter points to the buffer that is passed to the driver for further processing
thr_BufferSizePtr	On call: number of characters to write. If the function returns DRV_BUFFER_FULL, it contains the maximum number of characters that can be written. If the function returns DRV_OK, it contains the number of characters written. If the function returns DRV_INPROCESS, it contains 0.

Return values:

Name	Description
DRV_OK	Function successful
DRV_BUFFER_FULL	Not enough space
DRV_INPROCESS	Driver is busy writing data
DRV_NOTCONFIGURED	The driver is not yet configured

Description

This function is used to write data to the driver. The parameter *thr_BufferSizePtr contains the number of characters to write.

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

If the data cannot be written because the storage capacity of the driver has been exhausted, the function returns DRV_BUFFER_FULL and the maximum number of characters that can be written in *thr_BufferSizePtr.

If the driver is currently busy writing data and therefore cannot accept further data to be written, it returns DRV_INPROCESS and sets the parameter *thr_BufferSizePtr to 0.

If the driver is not configured, the function returns DRV_ NOTCONFIGURED.

NOTE: When calling the function with a buffer size of 0, the function will return the number of characters that can be written in the parameter *thr_BufferSizePtr. In this case, the in_BufferPtr can be set to NULL.

3.3.5 SER_Look – Look at received data of the driver

Definition:

```
USHORT SER_Look
(
        void*           out_BufferPtr,
        USHORT*         thr_BufferSizePtr
) ;
```

Parameters:

Name	Description
out_BufferPtr	This parameter points to the buffer wherein the data is to be copied
thr_BufferSizePtr	On call: number of characters to read. If the function returns DRV_OK, it contains the number of characters read. If the function returns DRV_INPROCESS, it contains 0.

Return values:

Name	Description
DRV_OK	Function successful
DRV_INPROCESS	The driver is currently reading data. The data is incomplete.
DRV_NOTCONFIGURED	The driver is not yet configured

Description

This function is used to read data from the driver. The data is copied into the buffer to which out_BufferPtr points. The parameter *thr_BufferSizePtr contains the size of the buffer in bytes. The internal buffer is not cleared.

In the case of a successful completion, the function returns DRV_OK and sets the value of *thr_BufferSizePtr to the number of bytes read.

In the case of successful completion, consecutive calls to SER_Look() will lead to identical results. Call SER_Read() to implicitly clear the driver's internal buffer.

If the driver is not configured, the function returns DRV_ NOTCONFIGURED.

NOTE: When calling the function with a buffer size of 0, the function will return DRV_OK. The size of the buffer needed to store the available data is stored in the parameter *thr_BufferSizePtr. In this case, the out_BufferPtr can be set to NULL.

3.3.6 SER_Clear – Clear internal driver buffers

Definition:

```
USHORT SER_Clear
(
        USHORT              in_BufferType
) ;
```

Parameters:

Name	Description
in_BufferType	Bit-mask used to specify if the read, write or read and write buffer is cleared

Return values:

Name	Description
DRV_OK	Function successful
DRV_INPROCESS	The driver could not complete the clearance of the buffers at once. The driver is busy clearing the buffers.

Description

This function is used to clear the device's internal buffers. The parameter in_BufferType is used to specify which buffer is to be cleared. The value of in_BufferType can be one of the values or a combination of the values defined in [C_8415.0026]. DRV_BUFTYPE_READ and DRV_BUFTYPE_WRITE can be combined to clear the read and write buffers at once.

3.3.7 SER_Flush – Flush internal driver buffers

Definition:

USHORT SER_Flush (void) ;

Parameters:

Name	Description
-	

Return values:

Name	Description
DRV_OK	Function successful
DRV_INPROCESS	The driver could not complete flushing the buffers at once. The driver is busy flushing the buffers.
DRV_NOTCONFIGURED	The driver is not yet configured

Description

This function is used to flush the device's internal buffers. This means that data that is currently stored in the device's internal write buffer is written to the USART at once.

If the driver is not configured, the function returns DRV_ NOTCONFIGURED.

3.3.8 SER_SetSignal – Setup a Signal

Definition:

```
USHORT SER_SetSignal
(
    USHORT          SignalType) ;
```

Parameters:

Name	Description
SignalType	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 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 only the standard signals defined in [C_8415.0026].

To remove a signal, call the function SER_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.

3.3.9 SER_ResetSignal – Remove a Signal

Definition:

```
USHORT SER_ResetSignal
(
        USHORT                  SignalType
) ;
```

Parameters:

Name	Description
SignalType	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 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 call-back function has been defined at the time of initialization, the driver returns DRV_SIGFCT_NOTAVAILABLE.

3.3.10 SER_SetConfig – Setup a device configuration

Definition:

```
USHORT SER_SetConfig
(
    T_SER_DCB*          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 serial device (port, transmission rate, flow control, etc). The driver can be configured at any one time. The parameters that can be configured are included in the device control block T_SER_DCB. For detailed information about the contents of the device control block, refer to Chapter 3.1.1.

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 SER_GetConfig() function to retrieve the driver's configuration.

The driver needs to be configured after initialization. Only the following functions can be called while the driver is not configured: SER_Clear, SER_SetSignal and SER_GetSignal. All other functions return DRV_NOTCONFIGURED.

3.3.11 SER_GetConfig – Retrieve a driver configuration

Definition:

```
USHORT SER_GetConfig
(
    T_SER_DCB*          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 serial device. 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 device control block, refer to Chapter 3.1.1. The configuration can be requested at any one time.

If the driver is not configured, the function returns DRV_ NOTCONFIGURED.

3.3.12 SER_CallBack – Callback entry for the driver

Definition:

```
void SER_CallBack
(
    T_DRV_SIGNAL        * Signal
) ;
```

Parameters:

Name	Description
Signal	Pointer to the signal information data

Return values:

Name	Description

-

Description

Callback entry for the hardware device driver.

3.3.13 SER_SetDevice - Set communication device (UART, USART, IRDA)

Definition:

```
USHORT  SER_SetDevice
(
  USHORT ComDevice
) ;
```

Parameters:

Name	Description
ComDevice	Communication Device (see table below)

Return values:

Name	Description
DRV_OK	Function successfully completed
DRV_INVALID_PARAMS	Device not found

Description

Device Codes :

ComDevice Value		Description
DEV_UART		Use UART device driver for communication
DEV_USART	USART Device	Use USART device driver for communication
DEV_IRDA	IRDA Device ID	Use IRDA device driver for communication

This Function sets the communication device used for all functions. The driver uses the corresponding hardware driver, until the function is called again.

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/>