Technical Document - Confidential

GSM FAX & DATA SERVICES

INTERFACE DESCRIPTION

USART

Document Number:	8411.801.98.204
Version:	0.7
Status:	Draft
Approval Authority:	
Creation Date:	1998-April-14
Last changed:	2015-Mar-08 by XINTEGRA
File Name:	Usart.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-Apr-14	Joachim Pilz		0.1		1
1998-Apr-17	Manfred Gutheins		0.2		2
1998-May-04	Joachim Pilz		0.3		3
1998-May-06	Joachim Pilz		0.4		4
1998-May-07	Joachim Pilz		0.5		5
1998-Nov-03	Manfred Gutheins		0.6		6
2003-Jun-03	XINTEGRA		0.7	Draft	

Notes:

1. Initial version
2. Revised Callback functions changed, baudrates added, XON/XOFF functions added, state extended with watermark and break length.
3. Return mechanism of the callback functions changed for HISR usage.
4. Return value US_NOT_READY added to some functions. Unit of the escape guard period defined.
5. SA, SB and X bit added in line state,
   readOutFunc must call US_InpAvail when invoked from ISR.
6. US_ReadData and US_WriteData: Deinstall callback function, when called in non suspended mode.

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

[1] Rec. T.4 Standardisation of group 3 facsimile apparatus for document transmission;
(CCITT-T.4, 1984)

[2] ITU-T Recommendation T.30; Series T: Terminal equipments and protocols for telematic services;
Procedures for document facsimile transmission in the general switched
telephone network;
(ITU-T.30, 1996)

[3] ITU-T Recommendation T.31; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 1
(ITU-T.31, 1995)

[4] ITU-T Recommendation T.32; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 2
(ITU-T.32, 1995)

[5] Rec. T.35; Terminal equipment and protocols for telematic services;
Procedures for the allocation of CCITT definde codes for non-standard facilities;
(CCITT-T.35, 1991)

[6] ITU-T Recommendation V.25 ter; Series V: data communication over the telephone network;
Interfaces and voiceband modems; Serial asynchronous automatic dialling and control
(ITU-T V.25 ter, 1997)

[7] Rec. V.42 bis Data compression procedures for data circuit terminating equipment (DCE) using error correction procedures;
(CCITT-V.42 bis, 1990)

[8] Rec. V.110 (Blue book, Vol. VIII, Fascicle VIII.1) Support of data terminal equipments (DTEs) with V-series type interfaces by an integrated services digital network (ISDN);
(CCITT-V.110, 1988)

[9] European digital cellular telecommunications system (Phase 2);
GSM Public Land Mobile Network (PLMN) connection types;
(GSM 3.10, September 1994, version 4.3.1)

[10] European digital cellular telecommunications system (Phase 2);
Technical realisation of facsimile group 3 transparent;
(GSM 3.45, September 1995, version 4.5.0)

[11] Digital cellular telecommunications system (Phase 2);
Mobile radio interface layer 3 specification;
(GSM 4.08, November 1996, version 4.17.0)

[12] European digital cellular telecommunications system (Phase 2);
Rate adaptation on the Mobile Station - Base Station System (MS - BSS) Interface;
(GSM 4.21, May 1995, version 4.6.0)

[13] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre (BSS - MSC) interface
(GSM 4.22, September 1994, version 4.3.0)

[14] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre (BSS - MSC) interface
(Amendment prA1 for GSM 4.22, version 4.3.0)
(GSM 4.22, March 1995, version 4.4.0)

[15] European digital cellular telecommunications system (Phase 2);
General on Terminal Adaptation Functions (TAF) for Mobile Stations (MS);
(GSM 7.01, December 1995, version 4.10.0)

[16] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using asynchronous bearer capabilities;
(GSM 7.02, September 1994, version 4.5.1)

[17] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using synchronous bearer capabilities;
(GSM 7.03, September 1994, version 4.5.1)

[18] Digital cellular telecommunications system (Phase 2);
Use of Data Terminal Equipment - Data Circuit terminating Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Services (CBS);
(GSM 7.05, November 1996, version 4.8.0)

[19] Digital cellular telecommunications system (Phase 2);
AT command set for GSM Mobile Equipment (ME)
(GSM 7.07, May 1996, version 4.1.0)

[20] Digital cellular telecommunication system (Phase 2);
Mobile Station (MS) conformance specification;
Part 1: Conformance specification
(GSM 11.10-1, November 1996, version 4.17.0)

[21] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 2: Protocol Implementation Conformance Statement (PICS)
proforma specification
(GSM 11.10-2, May 1996, version 4.15.0)

[22] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 3: Layer 3 (L3) Abstract Test Suite (ATS)
(GSM 11.10-3, November 1996, version 4.17.0)

[23] Proposal for Rate Adaptation implemented on a DSP;
(C. Bianconi, Texas Instruments, January 1998, version 1.0)

[24] MCU-DSP Interfaces for Data Applications;
Specification S844
(C. Bianconi, Texas Instruments, March 1998, version 0.1)

[25] Users Guide
6147.300.96.100; Condat GmbH

[26] Service Access Point RA
8411.100.98.100; Condat GmbH

[27] Service Access Point RLP
8411.101.98.100; Condat GmbH

[28] Service Access Point L2R
8411.102.98.100; Condat GmbH

[29] Service Access Point FAD
8411.103.98.100; Condat GmbH

[30] Service Access Point T30
8411.104.98.100; Condat GmbH

[31] Service Access Point ACI
8411.105.98.100; Condat GmbH

[32] Message Sequence Charts RLP
8411.201.98.100; Condat GmbH

[33] Message Sequence Charts L2R
8411.202.98.100; Condat GmbH

[34] Message Sequence Charts FAD
8411.203.98.100; Condat GmbH

[35] Message Sequence Charts T30
8411.204.98.100; Condat GmbH

[36] Message Sequence Charts ACI
8411.205.98.100; Condat GmbH

[37] Proposal for Fax & Data Integration; March 1998
8411.300.98.100; Condat GmbH

[38] Test Specification RLP
8411.401.98.100; Condat GmbH

[39] Test Specification L2R
8411.402.98.100; Condat GmbH

[40] Test Specification FAD
8411.403.98.100; Condat GmbH

[41] Test Specification T30
8411.404.98.100; Condat GmbH

[42] Test Specification ACI
8411.405.98.100; Condat GmbH
[43] SDL Specification RLP
8411.501.98.100; Condat GmbH
[44] SDL Specification L2R
8411.502.98.100; Condat GmbH
[45] SDL Specification FAD
8411.503.98.100; Condat GmbH
[46] SDL Specification T30
8411.504.98.100; Condat GmbH
[47] SDL Specification ACI
8411.505.98.100; Condat GmbH
[48] Technical Documentation RLP
8411.701.98.100; Condat GmbH
[49] Technical Documentation L2R
8411.702.98.100; Condat GmbH
[50] Technical Documentation FAD
8411.703.98.100; Condat GmbH
[51] Technical Documentation T30
8411.704.98.100; Condat GmbH
[52] Technical Documentation ACI
8411.705.98.100; Condat GmbH

1.2 Abbreviations

ACI	AT Command Interpreter
AGCH	Access Grant Channel
AT	Attention sequence "AT" to indicate valid commands of the ACI
BCCH	Broadcast Control Channel
BCS	Binary Coded Signals
BS	Base Station
BSIC	Base Station Identification Code
C/R	Command/Response
C1	Path Loss Criterion
C2	Reselection Criterion
CBCH	Cell Broadcast Channel
CBQ	Cell Bar Qualify
CC	Call Control
CCCH	Common Control Channel
CCD	Condat Coder Decoder
CKSN	Ciphering Key Sequence Number
CRC	Cyclic Redundancy Check
DCCH	Dedicated Control Channel
DISC	Disconnect Frame
DL	Data Link Layer
DM	Disconnected Mode Frame
DTX	Discontinuous Transmission
EA	Extension Bit Address Field
EL	Extension Bit Length Field
EMMI	Electrical Man Machine Interface
EOL	End Of Line
F	Final Bit
F&D	Fax and Data Protocol Stack
FACCH	Fast Associated Control Channel
FHO	Forced Handover
GP	Guard Period
GSM	Global System for Mobile Communication
HDLC	High level Data Link Control
HISR	High level Interrupt Service Routine
HPLMN	Home Public Land Mobile Network
I	Information Frame
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
ITU	International Telecommunication Union
IWF	Interworking Function
Kc	Authentication Key
L	Length Indicator
LAI	Location Area Information
LISR	Low level Interrupt Service Routine
LPD	Link Protocol Discriminator
M	More Data Bit
MCC	Mobile Country Code
MM	Mobility Management
MMI	Man Machine Interface
MNC	Mobile Network Code

MS	Mobile Station
MSG	Message phase in the GSM 3.45 protocol
N(R)	Receive Number
N(S)	Send Number
NCC	National Colour Code
NECI	New Establishment Causes included
OTD	Observed Time Difference
P	Poll Bit
P/F	Poll/Final Bit
PCH	Paging Channel
PCO	Point of Control and Observation
PDU	Protocol Description Unit
PL	Physical Layer
PLMN	Public Land Mobile Network
RACH	Random Access Channel
REJ	Reject Frame
RNR	Receive Not Ready Frame
RR	Radio Resource Management
RR	Receive Ready Frame
RTD	Real Time Difference
RTOS	Real Time Operating System
SABM	Set Asynchronous Balanced Mode
SACCH	Slow Associated Control Channel
SAP	Service Access Point
SAPI	Service Access Point Identifier
SDCCH	Slow Dedicated Control Channel
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSCB	Short Message Service Cell Broadcast
SS	Supplementary Services
T.4	CCITT Standardisation for Document coding of Group 3 Facsimile Apparatus
TAP	Test Application Program
TCH	Traffic Channel
TCH/F	Traffic Channel Full Rate
TCH/H	Traffic Channel Half Rate
TDMA	Time Division Multiple Access
TE	Terminal Equipment - e. g. a PC
TMSI	Temporary Mobile Subscriber Identity
UA	Unnumbered Acknowledgement Frame
UI	Unnumbered Information Frame
V(A)	Acknowledgement State Variable
V(R)	Receive State Variable
V(S)	Send State Variable
VPLMN	Visiting Public Land Mobile Network

1.3 Terms

Entity:	Program which executes the functions of a layer
Message:	A message is a data unit which is transferred between the entities of the same layer (peer-to-peer) of the mobile and infrastructure side. Message is used as a synonym to protocol data unit (PDU). A message may contain several information elements.
Primitive:	A primitive is a data unit which is transferred between layers on one component (mobile station or infrastructure). The primitive has an operation code which identifies the primitive and its parameters.
Service Access Point:	A Service Access Point is a data interface between two layers on one component (mobile station or infrastructure).

2 Overview

The Protocol Stacks are used to define the functionality of the GSM protocols for interfaces. The GSM specifications are normative when used to describe the functionality of interfaces, but the stacks and the subdivision of protocol layers does not imply or restrict any implementation.

The protocol stack for fax and data transmission consists of several entities. Each entity has one ore more service access points, over which the entity provides a service for the upper entity. The entity, which is described in this document, is coloured grey in the following figure :

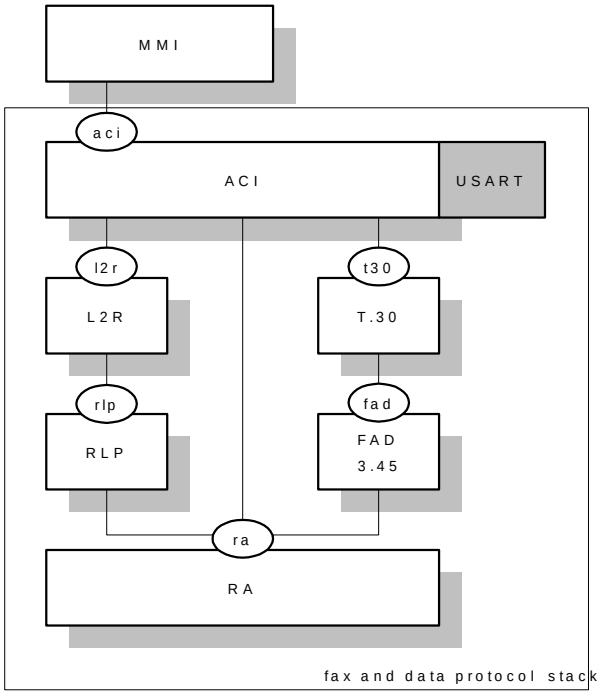


Figure 2-1: Architecture of the fax and data protocol stack

The information units passed via the SAPs are called primitives and consists of an operation code and several parameters. See the Users Guide for details.

The entities of the fax and data protocol stack are:

2.1 RA - Rate Adaptation

This entity performs an adaptation between an asynchronous or synchronous data stream with several bit rates on to the fixed bit rate used at the TCH. This is performed by the rate adaptation functions RA1' and RA0 described in GSM 04.21.

2.2 RLP - Radio Link Protocol

This entity provides a Layer 2 protocol for asynchronous reliable data transfer as specified in GSM 04.22. It includes error correction, sequence numbers and a mechanism for repeating corrupted and lost messages.

2.3 L2R - Layer 2 Relay Functionality

The L2R provides relay functions in order to adapt the character-oriented data received from the TE via USART to the bit-oriented RLP protocol.

2.4 FAD 03.45 - Fax Adaptation Protocol

The fax adaptation protocol, as specified in GSM 03.45, provides synchronisation with the BCS and MSG modems of the peer entity. It uses byte repetition in conjunction with a voting algorithm to handle corruption on the TCH data stream. The non-transparent fax protocol in accordance with GSM 03.46 is not part of this implementation.

The fax adapter enables T.30 to send BCS at 300 BPS and T.4 MSG in 2400, 4800, 7200 and 9600 BPS.

2.5 T.30 - Fax Protocol Entity

The protocol uses binary coded signals packed in HDLC frames to set up and release a connection in the message phase of the FAX transmission. This entity is specified in the ITU-T.30. The main tasks of this unit are:

- Building the HDLC frames with CRC.
- Performing bit stuffing/de-stuffing.
- Executing a sequence of 5 phases: 1.) set up, 2.) pre-message procedures, 3.) transmission/reception, 4.) post message procedures, 5.) waiting for call release.

2.6 ACI - AT Command Interpreter

The ACI is specified in GSM 07.07. It is responsible for call establishment via the GSM voice protocol stack and terminal adaptation for asynchronous transparent character-oriented data transmission. The ACI is able to receive AT commands and send the replies over the USART driver to a remote PC. This makes it possible to control the voice and data protocol stack from a remote application running on a PC. The ACI also provides a unique interface for an internal MMI in the MS.

2.7 USART - Universal Synchronous Asynchronous Receiver Transmitter Driver

The USART is a hardware component that facilitates a connection between the mobile station and terminal equipment (e.g. a PC). This interface uses some of the circuits described in V.24.

The data exchange provided by this unit is serial and asynchronous (synchronous communication is not in the scope of this document). A driver that uses interrupts to manage a circular buffer for the sending and receiving direction is necessary in order to use this component in the F&D. The driver has to be able to perform flow control.

3 Interface description of the USART driver

The USART driver should be implemented as an interrupt driven module which provides several functions for character oriented asynchronous I/O to the F&D protocol stack. The driver should be coded and maintained by TI because of the dependency of the used chip-set design.

This definition is based on the proposal made in [25] but the functionality of the character I/O functions differ. The main difference is the callback mechanism. The changes are necessary to speed up the character transfer in the protocol stack and to interchange the characters between different memory models. The different structure of the used memory models like the dual ported ram or the ram of the normal MCU requires different copy functions to transfer the received or transmittable characters.

TI has already implemented a USART driver for trace facilities. The main difference are the character I/O functions and the fact of detecting an escape sequence which depends on the timing and the contents of received data. In the following section a specification for a USART driver is given.

Functions of the synchronous operation mode are not specified here and should be negotiated in case of supporting synchronous data services.

3.1 Data types

The following definitions of C-prototypes uses data types that are defined for the GSM protocol stack environment.

BYTE	char	(8 bit)
UBYTE	unsigend char	(8 bit)
SHORT	short int	(16 bit)
USHORT	unsigned short int	(16 bit)
LONG	long int	(32 bit)
ULONG	unsigned long int	(32 bit)
BOOL	unsigned char	(8 bit)
T_USRET	short int	(16 bit)

3.2 Return Values

Most of the defined functions return a value which reports the success of the function call. The following values should be defined:

US_OK = 0

US_SUSPENDED = -1

US_NOT_SUPPORTED = -2

US_NOT_READY = -3

US_INTERNAL_ERR = -9

3.3 Initialisation

```
T_USRET   US_Init   (UBYTE usartNo)
```

Description

Initialises the USART hardware. This function should install the driver specific interrupt service routines/vectors and set up the communication parameters to a default state. The driver for the indexed usart should stay disabled until the US_Enable call is performed.

Parameters

usartNo is the index of an USART channel. This parameter is necessary to handle more than one hardware units within a single driver module.

Return values

US_OK	Successful operation
US_NOT_SUPPORTED	If the usart number does not specify a real port.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be only called once.

```
T_USRET   US_Enable  (UBYTE   usartNo,
                      BOOL    enable)
```

Description

A call to this function should enable or disable the functionality of the USART driver for the given hardware unit. In the deactivated state, all information about the communication parameters should be stored and recalled if the driver is enabled again. Only the RX and TX buffers should be cleared in this state.

Parameters

usartNo is the index of an USART channel.

enable is a boolean value where TRUE(1) enables the driver and FALSE(0) disables it.

Return values

US_OK	Successful operation
US_NOT_SUPPORTED	If the usart number does not specify a real port.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

3.4 Setup

These functions are used to set up and retrieve the parameters of the USART driver.

```
T_USRET   US_SetComPar (UBYTE                 usartNo,
                        T_baudrate            baudrate,
                        T_bitsPerCharacter    bpc,
                        T_stopBits            sb,
                        T_parity              parity)
```

Description

This function sets up the communication parameters: baud rate, bits per character number of stop bits and the parity for the serial communication with the TE.

Parameters

usartNo is the index of an USART channel.

baudrate specifies the used baud rate for the TE connection. The US_BAUD_AUTO should be used for auto detection of the baud rate and the framing format (if supported by the USART). The expected character for automatic detection is a "A" or "a". After successful detection of the speed and character framing the driver should work with the measured values. If one of the parameters bpc, sb or parity is specifed as xx_auto the driver should enter the same automatic format detection mode as in case of US_BAUD_AUTO.

The type *T_baudrate* is defined:

```
typedef enum
{
   US_BAUD_AUTO,
   US_BAUD_75,
   US_BAUD_150,
   US_BAUD_300,
   US_BAUD_600,
   US_BAUD_1200,
   US_BAUD_2400,
   US_BAUD_4800,
   US_BAUD_7200,
   US_BAUD_9600,
   US_BAUD_14400,
   US_BAUD_19200,
   US_BAUD_28800,
   US_BAUD_33900,
   US_BAUD_38400,
   US_BAUD_57600,
   US_BAUD_115200
   US_BAUD_203125
   US_BAUD_406250
   US_BAUD_812500
} T_baudrate
```

bpc specifies the used bits per character for the TE connection.

The type *T_bitsPerCharacter* is defined:

```
typedef enum
{
   bpc_7,
   bpc_8
} T_bitPerCharacter
```

sb specifies the used stop bits for the serial TE connection.

The type *T_stopBits* is defined:

```
typedef enum
{
   sb_1,
   sb_2
} T_stopBits
```

parity specifies the used parity for the serial TE connection.

The type *T_parity* is defined:

```
typedef enum
{
   pa_none
   pa_even
   pa_odd
   pa_space
} T_parity
```

Return values

US_OK	Successful operation
US_NOT_SUPPORTED	If the specified parameter does not fit to the capabilities of the usart hardware or the usart number does not specify a real port.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

```
T_USRET   US_SetBuffer   (UBYTE    usartNo,
                          USHORT   bufSize,
                          USHORT   rxThreshold,
                          USHORT   txThreshold)
```

Description

This function sets up the size of the circular buffer to be used in the USART driver and the callback condition for suspended I/O operations.

Parameters

bufSize specifies the size of the circular buffer to use in the USART driver. A maximum size is to be defined in the driver module. The driver should calculate a high watermark for the automatic flow control mode.

rxThreshold specifies the amount of received bytes that leads to a notification call to the suspended read-out function which is passed to the function US_ReadData.

txThreshold specifies the low watermark of the TX buffer which activates the call back of a suspended writeInFunction.

Return values

US_OK	Successful operation
US_NOT_SUPPORTED	If the bufSize exceed the maximal possible capabilities of the driver or if the threshold values does not correspond to the bufSize. This values should be returned too if the usart number does not specify a real port.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation but only if the related USART is disabled with US_Enable (usartNo, **0**).

```
T_USRET   US_GetComPar   (UBYTE                usartNo,
                          T_baudrate           *baudrate,
                          T_bitsPerCharacter   *bpc,
                          T_stopBits           *sb,
                          T_parity             *parity)
```

Description

Request the communication parameters from the USART driver. This function is only useful if automatic format detection is implemented. For the definition of the parameter types look at the definition of US_SetComPar.

Parameters

usartNo is the index of an USART channel.

baudrate specifies an address of a variable which after a successful call contains the adjusted baud rate for the TE connection.

bpc specifies an address of a variable which after a successful call contains the adjusted bits per character for the TE connection.

sb specifies an address of a variable which after a successful call contains the adjusted stop bits for the serial TE connection.

parity specifies an address of a variable which after a successful call contains the adjusted parity for the serial TE connection.

In case of automatic mode the values for bpc, sb, parity are set to xxx_auto and the baudrate is set to US_BAUD_AUTO, if there is still no speed and framing detected. For the definition of *T_baudrate, T_bitsPerChar, T_stopBits and T_parity* look at the function US_SetComPar.

Return values

US_OK	Successful operation
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

```
T_USRET   US_SetFlowCtrl  (UBYTE           usartNo,
                           T_flowCtrlMode  fcMode
                           UBYTE           XON
                           UBYTE           XOFF)
```

Description

This function changes the flow control mode of the USART driver. If the mode is set to none no automatic flow control is performed by the driver. In this case a loss of characters should be prevented by the protocol stack. The values for XON and XOFF are passed in case of inband flow control (fc_xoff). In the flow control mode fc_rts the driver should activate the CT106 (CTS) line, if it is able to store the received characters into the input buffer. This means also that the driver monitors the input of the CT133 line which is usually mapped onto CT105 (RTS) to detect the receive ability of the TE before he can send a character. The corresponding behaviour is also to be implemented for XON/XOFF and DTR/DSR. If the hardware does not support the V.24 lines RTS/CTS or DTR/DSR the set up to this modes should lead to an US_NOT_SUPPORTED return value.

Parameters

usartNo is the index of an USART channel.

fcMode is defined as follows:

```
typedef enum
{
   fc_none  = 0,
   fc_rts   = 1,
   fc_dtr   = 2,
   fc_xoff  = 3
} T_flowCtrlMode
```

XON defines the ASCII code of the character which should be detected/send as XON. This parameter should be ignored if the fcMode is not set to fc_xoff.

XOFF defines the ASCII code of the character which should be detected/send as XOFF. This parameter should be ignored if the fcMode is not set to fc_xoff.

Return values

US_OK	Successful operation
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit or the mode is not supported by the hardware or driver.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

```
T_USRET   US_SetEscape   (UBYTE    usartNo,
                          char     escChar,
                          USHORT   guardPeriod)
```

Description

To return to the command mode at the ACI while a data connection is established, normally the escape sequence „+++“ has to be detected. To distinguish between user data and the escape sequence a defined guard period is necessary before and after this sequence. This timing can only be detected by the driver because the receive buffer does not contain timing information for each byte. With this function the escape character and the guard period can be set up. In case of detecting the escape sequence the next US_ReadData call should lead to read out only that characters which are received before the escape sequence.

Parameters

usartNo is the index of an USART channel.

escChar ASCII character which could appear three times as an escape sequence.

guardPeriod Denotes the minimal duration of the rest before the first and after the last character of the escape sequence, and the maximal receiving duration of the whole escape string. The unit of this parameter should be milliseconds.

Return values

US_OK	Successful operation
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit or the mode is not supported by the hardware or driver.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

3.5 Character I/O

T_USRET **US_InpAvail** (*UBYTE* **usartNo**)

Description

The function returns the number of characters available in the RX buffer of the driver. If the driver is not enabled or during the automatic format detection mode the function should return 0.

Parameters

usartNo is the index of an USART channel.

Return values

>= 0	Successful operation. The returned value are the amount of data in the receive buffer.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit.
US_NOT_READY	The function is called while the callback of the readOutFunc is activated and still not terminated.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

T_USRET **US_OutpAvail** (*UBYTE* **usartNo**)

Description

The function returns the number of free bytes in the send buffer of the driver. If the driver is not enabled or during the automatic format detection mode the function should return 0.

Parameters

usartNo is the index of an USART channel.

Return values

>= 0	Successful operation. The returned value are the amount of data in the transmit buffer.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit.
US_NOT_READY	The function is called while the callback of the writeInFunc is activated and still not terminated.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

```
T_USRET   US_ReadData  (UBYTE           usartNo,
                        T_suspendMode   suspend,
                        void            (readOutFunc  (BOOL
cldFromIrq,
                                                       T_reInstMode
*reInstall,
                                                       UBYTE
nsource,
                                                       UBYTE
*source[],
                                                       USHORT
size[],
                                                       ULONG          state
                                                       )
                                        )
                       )
```

Description

To read the received characters out of the RX buffer of the driver the address of a function is passed. If characters are available, the driver should call this function and pass the address(es) and the amount(s) of readable characters. Normally the driver has implemented the buffer as a circular buffer, so the callback function will be called with more then one addresses of buffer fragments to read out the whole buffer.

The readOutFunc modifies the contents of the size array to return the driver the number of processed characters. Each array entry is incremented by the number of read bytes in the fragment. These values should cause the driver to modify the read-pointer of the circular RX buffer. If the US_ReadData function is called, while the receive buffer is empty, it depends on the suspend parameter to suspend the callback or to leave this function without operation. In the case of suspension the return value of the US_ReadData should be US_SUSPENDED. A later callback should be performed by the driver if the receive buffer reaches the adjusted threshold (**rxThreshold** of US_SetBuffer) or if the state of a V.24 input line changed or an break/escape is detected. If no suspension is necessary and the driver can call the readOutFunc it should return the number of processed bytes. If the driver is called with no suspension, then any previously installed callback function must be deinstalled.

The callback function activates a Nucleus HISR in case it is called from a LISR. The processing of the HISR (which performs the read-out) is delayed until the LISR terminates. The calling LISR in the driver has to modify the read-out pointer of the circular RX buffer. To determine the termination of the copy process in the HISR, the driver should watch (e.g. while the next IRQ) the content of the reInstall variable which should be set to rm_notDefined by the driver before the callback is performed. Moreover the callback function must call US_InpAvail after completing the copy process in order to guarantee, that the driver is activated. Otherwise, if no characters are received because of flow control, no IRQ would occur and the driver could not update its pointers.

Parameters

usartNo is the index of an USART channel.

suspend denotes the mode of suspension in case of an empty RX buffer in the driver.

It is defined as follow:

```
typedef enum
{
   sm_noSuspend  = 0,
   sm_suspend    = 1
} T_suspendMode
```

readOutFunc specifies the call back function to read out the RX buffer. The driver should pass the following parameters to this function:

cldFromIrq	This parameter should be set to TRUE(1) if the driver calls the function from the context of an interrupt service routine. Otherwise the driver should set this parameter to FALSE(0).

`reInstall` The address variable located in the driver. The driver should set the contents of this variable to rm_notDefined before it calls the callback function. If all necessary copy operations are performed, the callback function sets the content to rm_reInstall to signal the driver that it wants to be called again as soon as the level of the RX buffer reaches the rxThreshold. If the callback function sets the content to rm_noInstall the driver should not call this function again, except it will be reinstalled by a call to US_ReadData. To avoid a colliding call from an interrupt, the callback function ensures that the variable reInstall is modified only at the end of the function (or HISR).

```
typedef enum
{
      rm_notDefined   = 0,
      rm_reInstall    = 1,
      rm_noInstall    = 2
} T_reInstMode
```

`nsource` This parameter informs the callback function about the number of fragment which are ready to copy from the circular RX buffer.

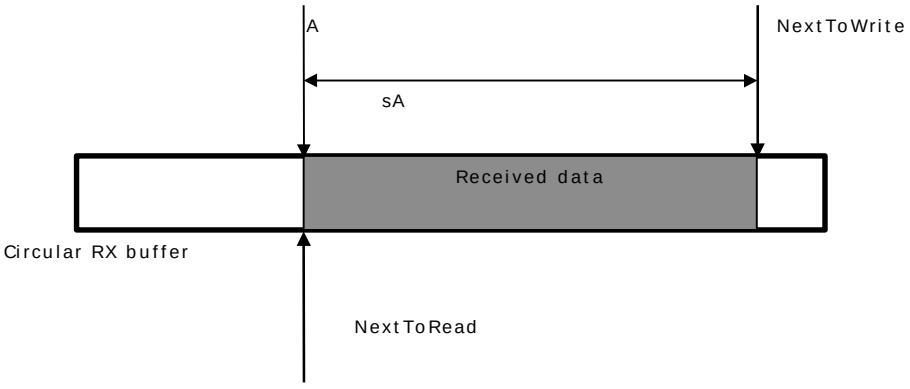
`source` Corresponding to the nsource parameter this array contains the addresses of the circular-RX-buffer fragments which are ready to read out. The driver should ensure that the start addresses of the fragments are sorted in sense of the receive order of the character sequence.

`size` Corresponding to the nsource and source parameter this array contains the amounts of bytes to read-out of the circular-RX-buffer fragments which are specified by source[]. The call back function will decrement the contents of this array with the amount of processed bytes for each fragment.

`state` the driver should pass the status of the V.24 lines and the break/escape detection to the call back function. The state parameter is described in the specification of US_GetLineState.

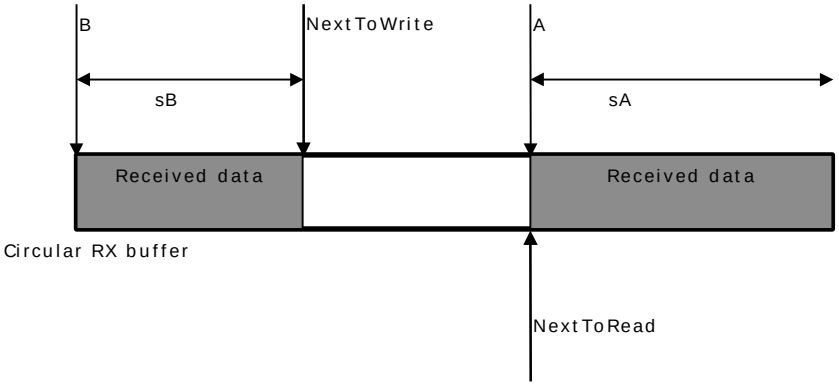
Example

This examples show the two cases of callback parameters for a circular RX buffer.



The readOutFunc should be called with nsource = 1, source[0] = A, size[0] = sA, reInstall = rm_notDefined.

The NextToRead pointer should be increased by (sA - size[0]) as soon as the content of the reInstall variable contains a value which is not equal to rm_notDefined.



The readOutFunc should be called with nsource = 2, source[0] = A, size[0] = sA

source[1] = B, size[1] = sB, reInstall = rm_notDefined

The NextToRead pointer should be increased by ((sA - size[0]) + (sB - size[1])) as soon as the content of the reInstall variable contains a value which is not equal to rm_notDefined.

After incrementing the NextToRead pointer it should be corrected with a circular algorithm.

Return values

>= 0	Successful operation. The returned value is the amount of processed data.
US_SUSPENDED	The callback was suspended until the buffer or state condition change.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit.
US_NOT_READY	The function is called while the callback is activated and still not terminated.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation and should suspend the call back in case of a disabled driver.

```
T_USRET   US_WriteData (UBYTE           usartNo,
                        T_suspendMode   suspend,
                        void            (writeInFunc   (BOOL
cldFromIrq,
                                                        T_reInstMode
*reInstall,
                                                        UBYTE          ndest,
                                                        UBYTE
*dest[],
                                                        USHORT         size[]
                                                        )
                                        )
                       )
```

Description

To write transmittable characters into the TX buffer of the driver the address of a function is passed. If free space is available in the buffer, the driver should call this function and pass the destination addresses and the amounts of space for several fragments of the RX buffer. Normally the driver has implemented the buffer as a circular buffer, so the callback function will be called with more then one addresses of buffer fragments to read out the whole buffer.

The writeInFunc modifies the contents of the size array to return the driver the number of processed characters. Each array entry is incremented by the number of written bytes in this fragment. These values should cause the driver to modify the write-pointer of the circular TX buffer. If the US_WriteData function is called while the transmit buffer is full, it depends on the suspend parameter to suspend the callback or to leave this function without operation. In the case of suspension the return value of the US_WriteData should be US_SUSPENDED (otherwise the function does not perform the callback and returns 0). A delayed callback should be performed by the driver if the buffer level of the TX buffer decreases to the adjusted threshold (**txThreshold** of US_SetBuffer). If no suspension is necessary and the driver can call the writeInFunc it should return the number of processed bytes. If the driver is called with no suspension, then any previously installed callback function must be deinstalled.

The callback function activates a Nucleus HISR in case it is called from a LISR. The processing of the HISR (which performs the write-in) is delayed until the LISR terminates. The calling LISR in the driver has to modify the write-in pointer of the circular TX buffer. To determine the termination of the copy process in the HISR, the driver should watch (e.g. while the next IRQ) the content of the reInstall variable which should be set to rm_notDefined by the driver before the callback is performed.

Parameters

usartNo is the index of an USART channel.

suspend denotes the mode of suspension in case of a full TX buffer in the driver. It is defined in the specification of the function US_ReadData.

writeInFunc specifies the call back function which writes into the TX buffer. The driver should pass the following parameters to this function:

`cldFromIrq` This parameter should be set to TRUE(1) if the driver calls the function from the context of an interrupt service routine. Otherwise the driver should set this parameter to FALSE(0).

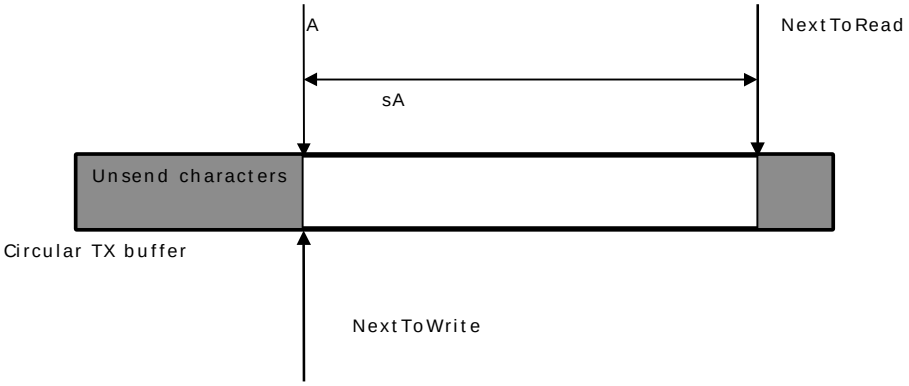
`reInstall` The address variable located in the driver. The driver should set the contents of this variable to rm_notDefined before it calls the callback function. If all necessary copy operations are performed, the callback function sets the content of this variable to rm_reInstall to signal the driver that it wants to be called again as soon as the level of the TX buffer decrease to the txThreshold. If the callback function sets the content to rm_noInstall the driver should not call this function again, except it will be reinstalled by a call to US_WriteData. To avoid a colliding call from an interrupt, the callback function ensures that the variable reInstall is modified only at the end of the function (or HISR).

```
typedef enum
{
      rm_notDefined    = 0,
      rm_reInstall     = 1,
      rm_noInstall     = 2
} T_reInstMode
```

ndest	This parameter informs the callback function about the number of fragment which are available in the TX buffer of the driver.
dest	Corresponding to the ndest parameter this array contains the addresses of the circular-TX-buffer fragments which are available to write in the characters which should be send. The driver should ensure that the start addresses of the fragments are sorted in sense of the transmitting order of the character sequence.
size	Corresponding to the ndest and dest parameter this array contains the amounts of bytes available in the circular-TX-buffer fragments. The call back function will decrement the contents of this array with the amount of processed bytes for each fragment.

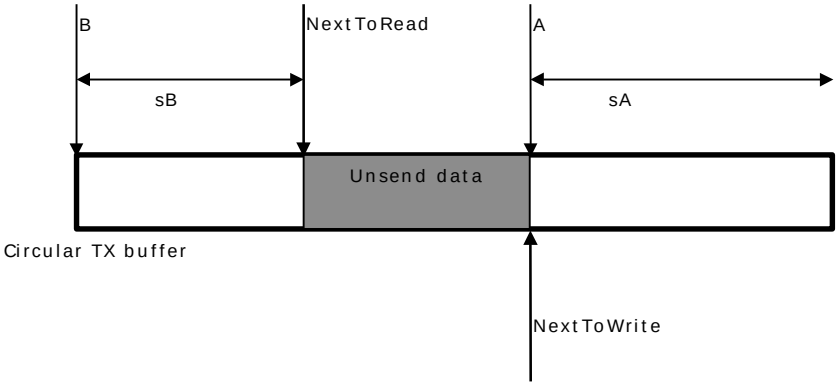
Example

This examples show the two cases of callback parameters for a circular TX buffer.



The writeInFunc should be called with ndest = 1, dest[0] = A, size[0] = sA, reInstall = rm_notDefined.

The NextToWrite pointer should be increased by (sA - size[0]) as soon as the content of the reInstall variable contains a value which is not equal to rm_notDefined.



The writeInFunc should be called with ndest = 2, dest[0] = A, size[0] = sA

dest[1] = B, size[1] = sB, reInstall = rm_notDefined

The NextToWrite pointer should be increased by ((sA - size[0]) + (sB - size[1])) as soon as the content of the reInstall variable contains a value which is not equal to rm_notDefined.

After incrementing the NextToWrite pointer it should be corrected with a circular algorithm.

Return values

>= 0	Successful operation. The returned value is the amount of processed data.
US_SUSPENDED	The callback was suspended until the buffer condition change.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit.
US_NOT_READY	The function is called while the callback is activated and still not terminated.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation and should suspend the call back in case of a disabled driver.

3.6 Flow Control and Control of the V.24 Lines

The local flow control is implemented in the USART driver. Nevertheless a description of the local flow control is given here, which might be helpful in understanding the whole concept.

For the data which are sent towards the network an entity can activate and deactivate the local flow control by sending a signal to the USART driver. In the other direction a local flow control condition is signalled by the USART driver to the entity.

The USART driver is responsible for controlling the signal lines in case of HW flow control (outband) and for managing XON/XOFF characters in case of SW flow control (inband). The USART driver itself has character buffers for sending and receiving and it is the task of this driver to manage different flow control conditions.

The driver activates local flow control towards the TE if its own buffer is filled up to a certain level or if it has received a flow control active condition from the entity. It deactivates local flow control when neither its own buffer is filled nor the entity has signalled active flow control.

In the other direction the USART driver passes the local flow control condition of the TE on to the entity. Flow control is not used to prevent the send buffer of the USART driver from filling up. Here the entity does not pass more data to the USART driver than can be placed in the buffer. If more data are coming from the network than can be transferred to the TE, the entity receive buffer fills up and flow control will be activated by the entity, when the threshold of the receive buffer is reached (back pressure).

Two methods of local flow control are allowed:

Outband (HW handshake)

- From TE: CT133 shall be turned OFF to indicate flow control active, and ON to indicate flow control inactive. CT133 is usually mapped onto CT105 (RTS) on standard connectors.
- From the mobile: CT106 (CTS) shall be turned OFF to indicate flow control active, and ON to indicate flow control inactive.

Inband (SW handshake)

- From TE: XOFF is sent to indicate flow control active. XON is sent to indicate flow control inactive. The XON/XOFF characters are extended by the USART driver from the data stream and are not sent across the radio interface. Where XON/XOFF is utilised, the mobile will generate flow control active/inactive immediately, i.e. the XON/XOFF characters do not enter any buffers.
- From the mobile: As from TE.

If the outband method is used, the entity will pass the DC1/DC3 characters as data, i.e. no flow control indications will be generated on receipt of DC1/DC3.

T_USRET **US_StopRec** (*UBYTE* **usartNo**)

Description

If a flow control mode is set, this function should tell the TE, that no more data can be received by the protocol stack. This may be done by sending a XOFF to the TE or deactivating CTS or respectively DSR. Usually the driver stops the TE automatically in case the RX buffer reaches a high water mark and start the TE again if enough buffer

size become available. If the US_StopRec function is called only a call to US_StartRec or an initialisation restarts the transmission from the TE.

Parameters

usartNo is the index of an USART channel.

Return values

US_OK	Successful operation.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

T_USRET **US_StartRec** (*UBYTE* **usartNo**)

Description

If a flow control mode is set, this function should tell the TE that the receiver is able to receive data. If the buffer in the driver has already reached the high water mark the driver should send the signal only if the buffer drains to a low water mark. The signal for the TE may be an XON or activating the RTS or respectively DTR line.

Parameters

usartNo is the index of an USART channel.

Return values

US_OK	Successful operation.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

```
T_USRET   US_GetLineState  (UBYTE   usartNo,
                            ULONG   *state)
```

Description

This function returns the state of the V.24 lines, the flow control state and the result of the break/escape detection process as a bit field. Additionally this parameter contains the relation of level of the

Parameters

usartNo is the index of an USART channel.

state is the address of a variable which contains after the successful function call the state in a bit-field representation.

A possible definition for the bit field is:

signal	bitpos	set/get	meaning
CTS	0	set	state of the V.24 line
RTS	1	get	state of the V.24 line
RI	1	set	state of the V.24 line
DSR	2	set	state of the V.24 line
DTR	3	get	state of the V.24 line
DCD	4	set	state of the V.24 line
BRK	5	set/get	break received/to be send
ESC	6	get	escape sequence detected
TXSTP	7	get	transmitter is stopped
RXSTP	8	get	receiver is stopped
BRKLEN	9-16	set/get	length of the break signal in characters
RXBLEV	17-28	get	number of free bytes in the RX buffer
SA	29	set/get	state of SA bit
SB	30	set/get	state of SB bit
X	31	set/get	state of X bit

The three bits SA, SB and X are mapped onto the corresponding lines. They are provided to give the user of the driver a more abstract view of the status lines. The mapping depends on the mode of flow control used:

signal		mode of flow control			
		none	rts	dtr	xon
SA	set	CT107 = DSR	CT107 = DSR	ignore	CT107 = DSR
	get	CT108 = DTR	CT108 = DTR	ON = 0	CT108 = DTR
SB	set	CT109 = DCD	CT109 = DCD	CT109 = DCD	CT109 = DCD
	get	CT105 = RTS	ON = 0	CT105 = RTS	CT105 = RTS
X	set	ignore	CT106 = CTS	CT107 = DSR	mapped onto XON/XOFF
	get	ON = 0	CT105 = RTS	CT108 = DTR	

The SA bit is always mapped onto DSR/DTR unless the mode of flow control is dtr. In this case the driver ignores any setting of the SA bit and returns a 0 (ON) as SA bit.

The SB bit is always mapped onto DCD/RTS unless the mode of flow control is rts. In this case RTS is used for the flow control and the driver returns always a 0 (ON) as SB bit.

The X bit is either mapped on CTS/RTS or DSR/DTR in case of outband flow control or on XON/XOFF in case of inband flow control. When no flow control is used, the driver ignores any setting of the X bit and returns a 0 (ON) as X bit.

Any unused line must be held in ON condition (0) by the driver.

Return values

US_OK	Successful operation.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit or the signal is not supported by the driver.
US_NOT_READY	The function is called while the callback of the readOutFunc is activated and still not terminated.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

```
T_USRET   US_SetLineState   (UBYTE   usartNo,
                             ULONG   state,
                             ULONG   mask)
```

Description

This function sets the states of the V.24 lines according to the bit field of the parameter **state**.

Parameters

usartNo is the index of an USART channel.

state is the bit-field which is defined in the specification for US_GetLineState. Only the signals which are marked with the „set"-access can be used to change the state of this signal.

mask is a bit-field with the same structure as **state**. Each bit in state 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 US_GetLineState.

Return values

US_OK	Successful operation.
US_NOT_SUPPORTED	If the usart index does not specify a real hardware unit or the signal is not supported by the driver.
US_INTERNAL_ERR	Internal problems with the hardware.

Context

This function should be callable any time after initialisation.

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/>