CAN - CBM - COM1

CAN - RS-232,
RS-422, RS-485
or TTY-Interface

Manual of the
Module-Specific Software

Document file:	I:\texte\Doku\MANUALS\CAN\Cbm\CBM_COM1\Englisch\COM1_01S.en9
Date of print copy:	13.05.2002

Software version described:	
CAN kernel:	see manual ‘esd-Protocol for CAN Modules’
esd protocol:	
Module-specific implementation:	Version ‘1IAE’

Changes in the chapters

The changes in the user’s manual listed below affect changes in the **firmware**, as well as changes in the **description** of the facts only.

Manual Rev.	Chapter	Changes versus previous version
1.0	-	First issue
	-	-

Technical details are subject to change without further notice.

N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. **esd** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. **esd** reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of **esd** gmbh.

esd does not convey to the purchaser of the product described herein any license under the patent rights of **esd** gmbh nor the rights of others.

esd electronic system design gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Phone: +49-511-372 98-0
Fax: +49-511-372 98-68
E-mail: info@esd-electronics.com
Internet: www.esd-electronics.com

USA / Canada
7667 W. Sample Road
Suite 127
Coral Springs, FL 33065
USA

Phone: +1-800-504-9856
Fax: +1-800-288-8235
E-mail: sales@esd-electronics.com

Contents

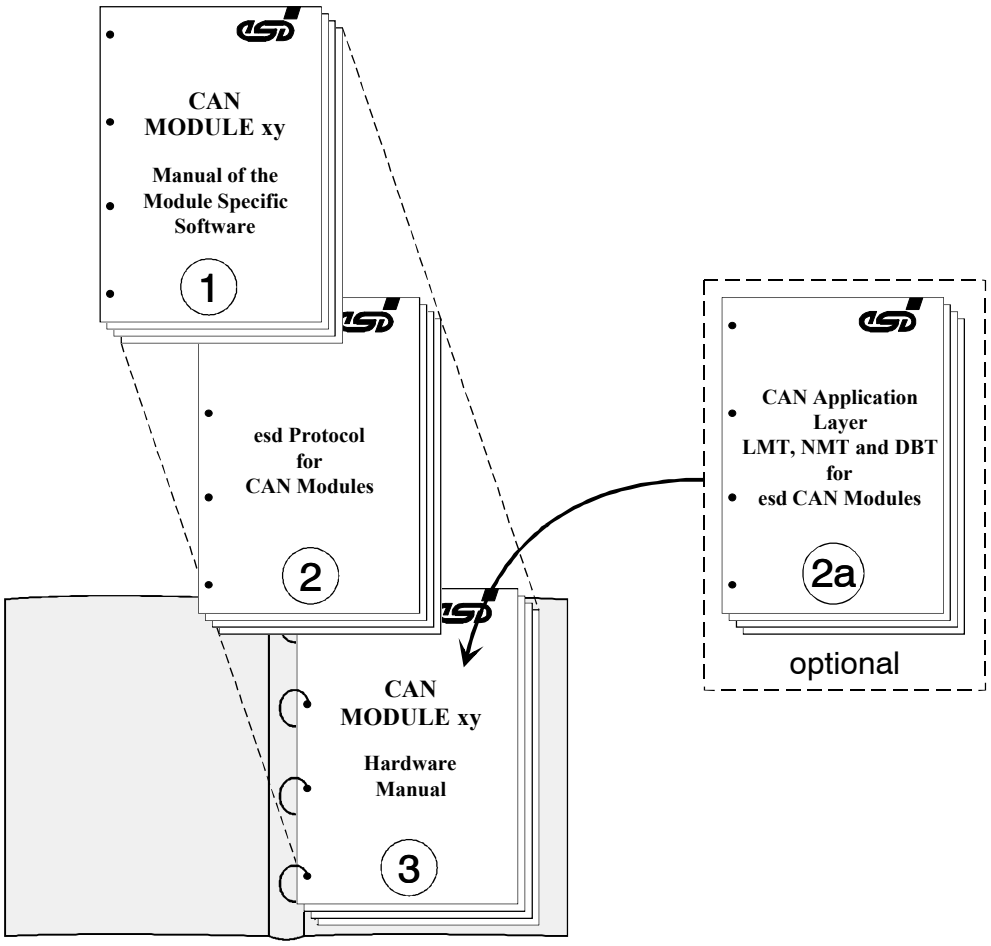
Page

This page is intentionally left blank.

1. Overview

1.1 What is where?

The description of the *esd*-module firmware is divided into **three manuals**, which will be delivered in one ring binder.



The first manual deals with software features and parameters which are only of significance for this module and can be used independently from the protocol.

CAN-CBM-COM1 **CAN - RS-232, RS-422, RS-485 or TTY** **Manual of the Module-Specific Software**

Here, e.g., the functions of the type specific firmware, the assignment of the identifiers and the assignment of the user parameters are explained.

The second manual contains general software specifications which are valid for all esd-CAN modules run by the same protocol.

Two different protocols are available for the modules: The esd-CAN protocol and the CMS protocol. The protocols are independent of each other and are used alternatively. Depending on the implemented protocol only one of the two following manuals is applicable for the module, therefore:

The esd-CAN protocol is described in the manual:

esd-Protocol for CAN Modules

It provides the user with the possibility to parameterize the *esd*-CAN modules by an initialisation identifier ($700). By this protocol it is possible to assign identifiers to the modules, set user parameters and activate watchdog functions.

Alternatively in addition to this it is possible to control the modules by the CMS protocol. If this protocol is implemented, the manual

CAN Application Layer LMT, NMT and DBT on esd Modules

has to be consulted. Here the realization of the CMS services of
the layer management (LMT),
the network management (NMT) and
the identifier distributors (DBT)
on *esd*-CAN modules is explained.

The third manual contains the hardware specifications of the module. It contains general and module specific descriptions. Here you can find e.g. assembly notes and connector arrangements.

CAN-CBM-COM1
CAN - RS-232, RS-422, RS-485 or TTY
Hardware Manual

1.2 Default Settings

The default settings of the module are active, if one or more of the following conditions applies:

- A default RESET was triggered on the module via the esd-CAN protocol.
- The data of the I²C-EEPROM is not OK (or the EEPROM is not equipped).
- The coding switches were set to '00' after a RESET or Power-On and have been set to a different value afterwards.

Individual parameters can be changed without influencing the default setting of other parameters. Changes in parameters only remain active after a RESET, if they have been buffered in the EEPROM.

Default values in operation of the module with esd-CAN protocol	
INIT-ID	in all operating modes $700
Rx-identifier, Tx-identifier	RxId1, TxId1 -> exchange of serial data RxId2, TxId2 -> only with CANlink (see page 3-12) The default values of the identifiers correspond to the settings via coding switches and jumper X100. A detailed description of settings can be taken from the hardware manual. (*)
Module No.	= setting of coding switches
CAN bit rate	= setting of jumper X100 (125 Kbit/s)

(*) The identifiers of the following modules have to be selected with an offset of at least +2, because otherwise identifiers would overlap!

Table 1.2.1: Default settings of the module when operated with esd-protocol

Default values of user parameters (independent from the protocol used)		
First Tx-activate Delay	10.000 ms	
CAN-Tx-Mode	$1411, i.e.	*MinChar, MaxChar* = 1, *Inhibit-Time* = 20 ms
Serial-Mode	$2273, i.e.	CTS active, 9600 baud, 2 stop bits, no parity, 8 bits/character
Tx_Start/Protocol	$0080, i.e.	*TxMinCharStart* = 0 *Protocol* = $80 (no protocol)
Handshake_On/Off	$0AF6, i.e.	handshake at 250 ... 10 bytes in buffer
Transfer_at_Rx_Timeout	$0000, i.e.	no transfers at Rx-timeout
End_Character	$0A0D, i.e.	carriage return and line feed available
Special_Baudrates	$0034, i.e.	baud rate = 9600 baud (if enabled via *Serial-Mode*)

Table 1.2.2: Default settings of user parameters of the module

Explanations of terms of user parameters in table 1.2.2:

First Tx-activate Delay... Delay after a RESET before the module starts transmitting messages to the CAN or serial interfaces.

CAN-Tx-Mode... *MinChar* and *MaxChar* determine the minimum and maximum number of data bytes which are to be transmitted within a CAN frame on the CAN. *Inhibit-Time* specifies the delay of the CAN controller between the last successful transmission on the CAN and the start of the following transmissions.

Serial-Mode... Via *Serial-Mode* the serial interfaces are configured.

Tx_Start/Protocol... Here, the minimum number of CAN-data bytes to be received before the data is transmitted on the serial interface is determined.
In addition, protocol parameters can be transmitted.

Handshake_On/Off... By means of the parameters specified here the number of data bytes in the receive buffer required to activate/deactivate the handshake functions of the serial interface is determined.

Transfer_at_Rx_Timeout... If no new data was received within the time specified in the timeout, the transmission fo CAN data is started, even if the minimum number of serial data bytes has not yet been received.

End_Character... By means of this parameter the characters which are to be evaluated as end characters of a serial data package can be specified.

Special_Baudrates... Should the baud rates which can be selected via user parameter *Serial-Mode* not be sufficient, further baud rates can be set via this user parameter.

This page is intentionally left blank.

2. Description of Data Transfer

Serial data is buffered between CAN and serial interfaces in both directions via a 256 byte sized toroidal core store. The data which is received first is transmitted again first.

If the toroidal core store is full and further data is received, this data will be lost. Therefore, the transmission rates of the CAN and the serial interfaces have to be synchronized.

2.1 Data Transfer CAN -> Serial Interface

The number of CAN data which is to be stored in the toroidal core store within an interval has to be smaller than the number of data transmitted via the serial interface.

The number of data received per interval depends on the number of data bytes transmitted, the frequency of transmissions, the bit rate of the CAN, and the CAN bus enabling assigned.

If data is lost during operation, breaks between transmissions have to be extended and/or the number of bytes transmitted has to be reduced.

The data is transmitted from CAN via identifier RxId1 or COB 1 to the module. Any number of bytes to be transmitted can be selected.

CAN-Identifier	length	Data							
		Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
RxId 1	0...8	data from CAN to serial interface							

Table 2.1.1: Receiving data to be transmitted via Rx-identifier

When the module is run with esd protocol, the data output of the serial interface can be stopped by means of the supervisor command ‘Suspend Module’. The data transfer is resumed by the command ‘Continue’.

If the module is in ‘Suspended’ status, received data is stored in the toroidal core store by both CAN and serial interface.

The data transfer from CAN to serial interface has the following chronological course:

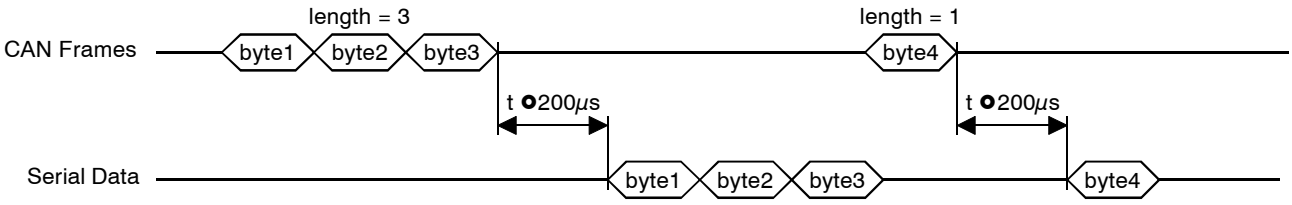


Fig. 2.1.1: Data transfer CAN -> serial interface (parameter *TxMinCharStart* = 0)

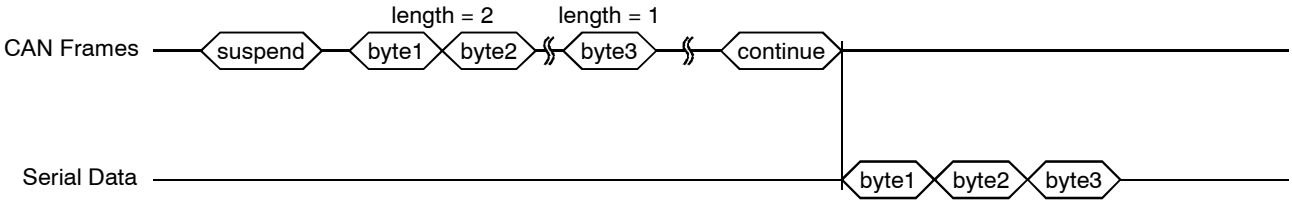


Fig. 2.1.2: Data transfer CAN -> serial interface, controlled by 'Suspend' and 'Continue' (parameter *TxMinCharStart* = 0)

Parameter *MinChar* determines the minimum number of data bytes which are to be transmitted on the CAN within a CAN frame (see page 3-4).

2.2 Data Transfer Serial Interface -> CAN

The number of data of the serial interface to be stored in the toroidal core store per interval has to be lower than the number of data to be transmitted via the CAN.

The CAN limits the data flow via bus capacity (priority), CAN-bit rate, the frequency of transmissions and the number of bytes transmitted.

Via parameters *Inhibit-Time*, *MaxChar* and *MinChar* the module offers possibilities to influence the last two factors. These parameters are combined in the user parameter *CAN-Tx-Mode* (see page 3-3).

By *Inhibit-Time* the delay between two transmissions on the CAN is determined.

MinChar and *MaxChar* determine the minimum and maximum number of data bytes which are to be transmitted on the CAN within a CAN frame.

The data is transmitted by the module via Tx-identifier TxId1 or COB 5 on the CAN.

CAN identifier	length	Data							
		Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
TxId 1	0...8	data from serial interface to CAN							

Table 2.2.1: Transmission of data received by serial interface via Tx-identifier

This page is intentionally left blank.

3. User Parameters of CAN-CBM-COM1

Via the user parameters settings for the serial interfaces and the CAN interface are specified for the module.

If the module is run with the esd protocol, the user parameters are specified by means of command 'Set User Parameters' ($86) on bytes 5 and 6 of the INIT-Id ($700).
All user parameters must always be specified as 16-bit value with byte 5 as MSB!

If the CMS protocol is implemented, the user parameters are set via a configuration download (NMT).

The following table gives an overview of the user parameters of the CAN-CBM-COM1 module.

User-parameter No.	Parameter	Value range	Default setting
$00	*First Tx-activate Delay*	$0000...$FFFF (0...65535 ms)	10.000 ms
$01	*CAN-Tx-Mode*	$0011...$FF88	$1411
$02	*Serial-Mode*	$0000...$88FF	$2273 (9600 baud serial)
$03	*Tx_Start/Protocol*	$0000...$FF83	$0080
$04	*Handshake_On/Off*	$0000...$FFFF	$0AF6
$05	*Transfer_at_Rx_Timeout*	$0000...$00FF	$0000
$06	*End_Character*	$0000...$FFFF	$0A0D
$07	*Special_Baudrates*	$0001...07FF	$0034

Table 3.1.1: User parameters of the module

3.1 *First Tx-activate Delay* (Parameter 0)

Parameter 0 specifies the delay between a RESET and the start of the transmission of data on the CAN and the serial interfaces.

This delay is used to guarantee that all modules run steadily on the CAN before the CAN-CBM-COM1 module starts its transmissions.

User-parameter No. (= sub-command No.)	Parameter	Value range	Default setting
$00	*First Tx-activate Delay*	$0000...$FFFF (0...65535 ms)	10.000 ms

Table 3.1.2: User parameter 0

3.2 CAN-Tx-Mode (Parameter 1)

By means of this user parameter the parameters *Inhibit Time, MaxChar* and *MinChar* are specified.

Inhi*bit Time dete*rmines the delay between two transmissions on the CAN.
MinChar and *MaxChar* determine the minimum and maximum number of data bytes which is to be transmitted on the CAN within a CAN frame. Alternatively, an abort command can be specified via parameter *MinChar*.

User-parameter No. (sub-command No.)	Parameter	Value range	Default setting
$01	*CAN-Tx-Mode*	$0011..$FF88	$1411

Table 3.2.1: User parameter *CAN-Tx-Mode*

The two bytes of parameter *CAN-Tx-Mode* are structured as follows:

	CAN-Tx-Mode		
	Byte 5 of INIT-Id $700	Byte 6 of INIT-Id $700	
Parameter	*Inhibit Time*	*MaxChar*	*MinChar*
Value range	$00...$FF	$1...$8	$1...$8, $9...$E

Table 3.2.2: Structure of parameter *CAN-Tx-Mode*

Inhibit-Time... Parameter *Inhibit-Time* determines the time the CAN controller waits after the last successful transmission of CAN data, before it initiates another transmission. The delay is specified in [ms].
The default setting is $14 (= 20 ms).

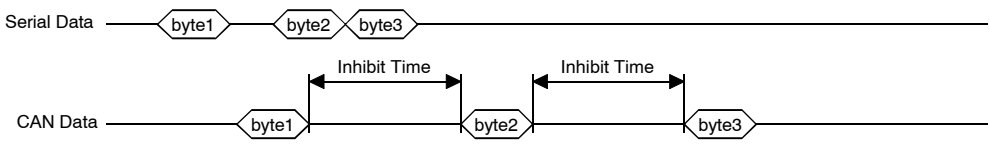


Fig. 3.2.1: Function of parameter *Inhibit Time* (Parameter *MaxChar* = 1)

MinChar, MaxChar... These two parameters determine the number of bytes from which a transmission to the CAN is to be started, and the maximum number of data bytes to be transmitted in a CAN frame.

Default setting is $11, i.e. each received byte is transmitted individually in a frame.

Transmission command:

MinChar is only evaluated as described above, if values between $1...$8 are specified.

If values between $9 and $F are specified, the functionality of the parameter changes: In this case the local software evaluates the entry of characters which have been specified via user parameter *End-Character* as transmission command.

In default setting of user parameter *End_Character* these are the flags 'Carriage Return' ($0D) or 'Line Feed' ($0A). As soon as one or both flags are detected, the data which has been received by the serial interface is transmitted to the CAN. If 8 bytes have been received before the flags have been received, the transmission starts automatically.

You can also specify for the transmission, whether the flags are to be transmitted on the CAN together with the data or not. The following table shows the various options:

MinChar [HEX]	Transmission command	Flag transmission to CAN	Comments
1...8	-	-	At least 1 to 8 bytes are always transmitted. Parameter *MaxChar* is also evaluated.
9	<*Char_1*>	with <*Char_1*>	Data is transmitted after <*Char_1*> has been received. <*Char_1*> is also transmitted.
A	<*Char_2*>	with <*Char_2*>	As under '9', but with <*Char_2*>.
B	<*Char_1*>	without <*Char_1*>	As under '9', but with <*Char_1*> is not transmitted as well.
C	<*Char_2*>	without <*Char_2*>	As under 'B', but with <*Char_2*>.
D	<*Char_1*>	without <*Char_1*> and without <*Char_2*>	The end of the message can have <*Char_1*> and <*Char_2*>. The data is transmitted after <*Char_1*> has been received. Neither <*Char_1*> nor <*Char_2*> are transmitted as well.
E	<*Char_2*>	without <*Char_1*> and without <*Char_2*>	As under 'D', but with <*Char_2*>.
F	-	-	reserved

Table 3.2.3: Selection of transmission command via parameter *MinChar*

Examples:

The examples below are representative for the CAN-CBM-COM1 being operated by parameters in default setting, i.e. *Char_1* = <Cr>, *Char_2* = <Lf>.

1. For *MinChar* value $B has been selected. Via the serial interface the data 'abcd<Cr>' is received. The data 'abcd' would be transmitted on the CAN after <Cr> had been received.

2. For *MinChar* value $E has been selected. Via the serial interface the data 'abcd<Cr><Lf>' is received. The data 'abcd' would be transmitted on the CAN after <Lf> had been received.

3.3 *Serial-Mode* (Parameter 2)

User parameter *Serial-Mode* sets the baud rate (in 14 steps), the stop bits, the number of bits/character and the CTS lock, and determines the parity evaluation of the serial interface.

User-parameter No. (sub-command No.)	Parameter	Value range	Default setting
$02	*Serial-Mode*	$0000..$88FF	$2273

Table 3.3.1: User parameter *Serial-Mode*

The two bytes of parameter *Serial-Mode* are structured as follows:

	Serial-Mode		
	Byte 5 of INIT-Id $700		Byte 6 of INIT-Id $700
Parameter	*Rx-Baudrate*	*Tx-Baudrate*	*mode*
Value range	$0...$8	$0...$8	$00...$FF

Table 3.3.2: Structure of parameter *Serial-Mode*

Rx-Baudrate, Tx-Baudrate...

In 4 bits the baud rate is specified with which data is to be transmitted (Tx) or received (Rx) on the serial interfaces. The default setting is 9600 kbit/s for Rx- and Tx-baud rates.

The physically attainable baud rate is limited by the hardware: RS-422, RS-485: max. ca. 125 kbit/s, RS-232: max. ca. 38.4 kbit/s, TTY: typ. 1200 bit/s.

The same value has to be selected for Rx- and Tx-baud rate!

Via user parameter *Special_Baudrates* further baud rates can be set! If you want to do this, the value '$E' has to be specified for parameters *Rx-Baudrate* and *Tx-Baudrate.*

Parameter *Rx-(Tx) Baudrate* [HEX]	**Baud rate** (set value) [bit/s]	Baud rate (realized values) [bit/s]
0	3.8400	3.8462
1	19200	19231
2	**9600**	9615
3	4800	4808
4	2400	2404
5	1200	1199
6	600	600
7, 8	300	300
9	7200	7246
A	14400	14286
B	28800	29412
C	(57600)	55556
D	(115200)	(125000)
E	variable baud rate	variable baud rate
F	-76800	-71429

Table 3.3.3: Setting of baud rate of the serial interfaces in 14 steps

Mode...

The bits of parameter *Mode* have got the following functions:

Bit	7	6	5	4	3	2	1	0
Assign-ment	*RTS-Mode*	*CTS-enable*	*Stop-Bit*	*Parity-Mode*		*Parity-Type*	*Bits per Character*	
Default	0	1	1	1	0	0	1	1

Table 3.3.4: Assignment of parameter *Mode*

The module can only handle 8 bits per character. Furthermore the maximum number of serial bits is limited to 11 by the microcontroller. Therefore, the following permissible combinations result for bits 4...0:

Bit						Function
7...5	4	3	2	1	0	
RTS, CTS, Stop: see following tables	*Parity-Mode*		*Parity-Type*	*Bits per Character*		
	1	0	x	1	1	No Parity, evaluate 1 or 2 Stop-Bits
	1	1	a	1	1	Force Tx-Parity to value of 'a', 1 Stop-Bit
	0	1	O/E*	1	1	Tx Parity acc. to Bit 'Parity Type', no Rx-Parity, 1 Stop-Bit
	0	0	O/E*	1	1	Tx-Parity and Rx-Parity acc. to Bit 'Parity Type', 1 Stop-Bit

Table 3.3.5: Permissible combinations of bits 4...0

The coding of individual bits of parameter Mode will be described in detail below:

Explanation of bits of parameter Mode

RTS-Mode... Via this bit the RTS-modem mode for RS-485 interfaces can be selected.

RTS-Mode	Evaluation
0	RTS on Rx (default setting) hardware-handshake signal
1	RTS-modem (RS-485)

Table 3.3.6: Evaluation of *RTS-Mode* bit

CTS enable... This bit is only important, if the RTS-mode is set to '0'!

Via this bit the CTS-function of the serial controller is enabled.

If the bit is '0', the CTS-signal is not evaluated and the controller transmits the available data immediately (if the transmitter is not blocked by other commands; such as 'suspended', <Xoff> received, number data < *MinChar*).

CTS enable bit	CTS -evaluation
0	CTS -input is ignored
1	CTS -input is evaluated: hardware handshake (default setting)

Table 3.3.7: Evaluation of *CTS enable* bit

Stop Bit... Here the number of stop bits of the serial interface is determined:

Stop Bit	Number of Stop Bits
0	1 Stop Bit
1	2 Stop Bits (default setting)

Table 3.3.8: Number of Stop Bits

Parity Mode... By means of these two bits the evaluation of the parity bit is specified:

Parity Mode		Evaluation
Bit 4	Bit 3	
0	0	parity evaluation when receiving data and transmitting the parity bit
0	1	parity bit is only transmitted
1	0	no parity evaluation, no parity transmission (default setting)
1	1	value of the parity bit to be transmitted is specified in bit *Parity Type* (‘forced parity’)

Table 3.3.9: Parity evaluation

Parity Type... The polarity of the parity bit is specified by parameter bit *Parity Type*.

Parity Type	Polarity
0	‘even’ (default setting)
1	‘odd’

Table 3.3.10: Setting the polarity

Bits per Character... The number of bits per character is set to ‘8’ and cannot be changed. Therefore, these two bits have always to be set to ‘1’!

3.4 Tx_Start/Protocol (Parameter 3)

User-Parameter No. (sub-command No.)	Parameter	Value range	Default setting
$03	*Tx_Start/Protocol*	$0000...$FF83	$0080

Table 3.4.1: User parameter *Tx_Start/Protocol*

User parameter *Tx_Start/Protocol* is divided into two parameters with one-byte length each:

Tx_Start/Protocol	
Byte 5 of INIT-Id $700	Byte 6 of INIT-Id $700
TxMinCharStart	*Protocol*

Table 3.4.2: Structure of parameter *Tx_Start/Protocol*

TxMinCharStart... In this parameter the minimum number of data bytes is specified that has to be stored in the serial Tx-buffer before the transmission of data on the serial interface is started (data CAN -> serial).

This function is required, if strings of more than 8 bytes are to be transmitted together on an RS-485 interface (such as with Master/Slave protocols).

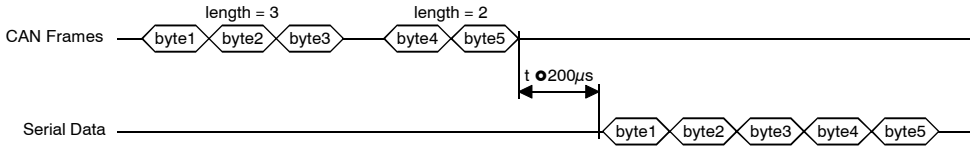


Fig. 3.4.1: Function of parameter *TxMinCharStart*

Protocol...

By means of parameter *Protocol* handshake functions of the CAN and the serial interface can be selected.

Bit	Name	Meaning
0	*RTR_HS*	0 - no RTR-handshake (default setting) 1 - remote mode (RTR-handshake)
1	*XON/XOFF_EN*	0 - no XON/XOFF 1 - XON/XOFF-handshake enabled
2 : 6	-	reserved (always set to '0')
7	*CANlink*	0 - module transmits and receives CANlink frames 1 - no CANlink frames

Table 3.4.3: Bits of parameter *Protocol*

Bit 0: *RTR_HS* (RTR-Handshake)

In this handshake mode the CAN-CBM-COM1 module responds with an RTR-frame to the Rx-data received by a CAN master (data CAN -> serial), if capacity for at least one more CAN frame is available in the buffer. The RTR-frame is transmitted on the identifier on which the data has been received (RxId1).

If the CAN-CBM-COM1 receives an RTR on TxId1, the data is transmitted from the Rx-buffer, even if the number of data in the buffer is smaller than the number defined in *MinChar*.

Bit 1: *XON/XOFF_EN*

Alternatively to the hardware handshake, this mode can be selected for the serial interface, if the handshake is activated by parameter *Handshake_On/Off*.

Bit 7: *CANlink*

CANlink is a handshake protocol for the CAN. It transmits up to 6 bytes effective information and two bytes handshake information in a CAN frame.

If CANlink is activated, the CAN-CBM-COM1 module utilizes another Tx-identifier (TxId2) and another Rx-identifier (RxId2) on which acknowledge messages are transmitted or received.

In default setting of the jumpers on the module the least significant identifier bits are assigned as follows:

RxId2: id3 = 0, id2 = 1, id0 = 0

TxId2: id3 = 0, id2 = 1, id0 = 1

CANlink will not be explained further in this manual. Please do not hesitate to contact our support to get more information.

3.5 Handshake_On/Off (Parameter 4)

User-parameter No. (sub-command No.)	Parameter	Value range	Default setting
$04	*Handshake_On/Off*	$0001...$FFFF	$0AF6

Table 3.5.1: User parameter *Handshake_On/Off*

User parameter *Handshake_On/Off* is divided into two parameters with one-byte length each:

Handshake_On/Off	
byte 5 of INIT-Id $700	byte 6 of INIT-Id $700
Handshake_Off	*Handshake_On*

Table 3.5.2: Structure of parameter *Handshake_On/Off*

Handshake_On,
Handshake_Off...

By means of these parameters the handshake function of the serial interface can be influenced for the data direction serial -> CAN (*this function is not supported in modem mode!*).

The buffer for the serial data received has a capacity of 256 bytes. In order to prevent the buffer from overflowing and data to be lost, the transmitter of the serial data can be told via the handshake line RTS or via protocol byte <Xoff> to suspend transmission until the buffer has capacity for new data again.

In parameter *Handshake_On* the number of bytes received in the Rx-buffer is entered from which the serial data flow is to be suspended. In default setting this value is $F6, i.e. if 250 bytes are in the Rx-buffer, the RTS-signal of the serial interface will be activated. If *XON/XOFF_EN* is enabled by user parameter *Tx_Start/Protocol,* the software handshake <Xoff> also signalizes that the buffer is not ready to receive.

In parameter *Handshake_Off* the number of bytes in the Rx-buffer is specified from which the reception of serial data will be enabled again. In default setting this value is $0A, i.e. if only 10 bytes are left in the Rx-buffer, the RTS-signal of the serial interface will be deactivated. If *XON/XOFF_EN* is enabled via user parameter *Tx_Start/Protocol*, the software handshake <Xon> also signalizes that the buffer is ready to receive again.

The value of parameter *Handshake_Off* must always be smaller than the value of parameter *Handshake_On*!

If the bit *RTS_Mode* = '0', the RTS-signal always has the current handshake, regardless of the bit *XON/XOFF_EN*.

Note: User parameter *Handshake_On/Off* does not have an influence on the handshake performance in the opposite data direction, i.e. CAN -> serial:

The transmitter of the serial interface is being enabled,

- if the bit *CTS-Mode* = '1' and the CTS-signal is inactive, or
- if *XON/XOFF_EN* = '1' and <Xoff> has been received.

3.6 Transfer_at_Rx_Timeout (Parameter 5)

User-parameter No. (sub-command No.)	Parameter	Value range	Default setting
$05	*Transfer_at_Rx_Timeout*	$0000...$00FF	$0000

Table 3.6.1: User parameter *Transfer_at_Rx_Timeout*

Only byte 6 of user parameter *Transfer_at_Rx_Timeout* is relevant. Byte 5 must always be set to '00':

Transfer_at_Rx_Timeout	
byte 5 of INIT-Id $700	byte 6 of INIT-Id $700
always set to '00'	*TxByRxTimeout*

Table 3.6.2: Bytes of parameter *Transfer_at_Rx_Timeout*

TxByRxTimeout...

The serial data received on the CAN is usually only transmitted after the number of bytes specified in *MinChar* in user parameter *CAN-Tx-Mode* has been received by the serial interface.

For large data rates it is useful to set parameter *Minchar* = *MaxChar* = '8' in order to get as much data into a CAN frame as possible. It is possible, however, that less bytes than specified in *MinChar* are stored in the buffer of the CAN-CBM-COM1 with the last transmission. This data would then only be transmitted when transmissions will be resumed again. In order to prevent these data from remaining in the buffer for an unspecified period, parameter *Transfer_at_Rx-Timeout* has been introduced.

If at least one data byte is in the serial Rx-buffer and no further data is received by the serial interface within a specified timeout, the data of the Rx-buffer will be transmitted on the CAN, even if the number of bytes is still smaller than specified in *MinChar.*

In default setting *TxByRxTimeout* = '0' and therefore the transmission function is blocked. The value for *TxByRxTimeout* is specified in [ms]. Values up to 255 ms are permissible.

3.7 End_Character (Parameter 6)

User-parameter No. (sub-command No.)	Parameter	Value range	Default setting
$06	*End_Character*	$0000...$FFFF	$0A0D

Table 3.7.1: User parameter *End_Character*

User parameter *End_Character* is divided into two parameters with one-byte length each:

End_Character	
byte 5 of INIT-Id $700	byte 6 of INIT-Id $700
Char_2	*Char_1*

Table 3.7.2: Structure of parameter *End_Character*

Char_1, Char_2... By means of user parameter *CAN-Tx-Mode,* described above, and selection of parameter *MinChar* the transmission of serial data received on the CAN can be initiated after the end command Carriage Return <Cr> or Line Feed <Lf> have been received.

The characters whose reception is evaluated as end command can be defined by means of user parameter *End_Character,* i.e. other characters than <Cr> or <Lf> can be selected. In default setting *Char_1* is assigned by <Cr>, that is $0D, and *Char_2* is assigned by <Lf>, that is $0A.

3.8 Special_Baudrates (Parameter 7)

User-parameter No. (sub-command No.)	Parameter	Value range	Default setting
$07	*Special_Baudrates*	$0000...$07FF	$0034

Table 3.8.1: User parameter *Special_Baudrates*

The baud rate of the serial interface can be set in 14 steps by means of user parameter *Serial-Mode* (see page 3-6). Further baud rates can be set by means of user parameter *Special_Baudrates.*

In *Special_Baudrates* an integer *'N*' is specified whose value range is between:

N = 1....$07FF, i.e. $1....2047_{decimal}$.

The baud rates that can be set are determined by means of the following equation:

$$\text{baud rate [Baud]} = \frac{8 \cdot 10^6}{16 \cdot N}$$

Because N has to be an integer, it is not possible to attain all standard baud rates exactly!

This page is intentionally left blank.

4. Examples

In this chapter the operation and initialization of a module run by the ***esd-CAN Protocol*** will be explained by means of a few examples.

4.1 Operation by Default Parameters

4.1.1 Conditions, Target

A device whose parameters correspond to the default setting of the CAN-CBM-COM1 is to be connected to the CAN-CBM-COM1:

- 9600 baud
- 8 bits/character
- 2 stop bits
- no parity

The CAN-CBM-COM1 module has not yet been initialized.

The desired Tx-identifier for transmitting data to the serial interface is to be $54x. The Rx-identifier is to be $54y. The last three identifier bits are to be assigned to correspond to the default setting at X100.

4.1.2 Process

4.1.2.1 Setting Identifiers

The identifier is defined by means of jumper X100 and the coding switches.

The last three identifier bits, which define the Tx- and Rx-identifier, are set via X100. In default setting these bits are as follows:

Identifier	Identifier bits		
	id 3	id 2	id 1
RxId1	0	0	0
TxId1	0	0	1

Table 4.1.1: Setting identifier bits id1...id3

The coding switches can be accessed externally: Bits 11...8 are set via coding switch ‘HIGH’, bits 7...4

are set via coding switch ‘LOW’.
The position and function of jumpers and coding switches are described in detail in the hardware manual of the module.

4.1.2.2 Transmitting Data to the Serial Interface

Via Rx-identifier RxId1 the data is transmitted to the module. The number of data bytes transmitted can be between 0 and 8. In this example the following five bytes are to be transmitted:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
$11	$22	$33	$44	$55

On the CAN the bytes are transmitted via Rx-identifier RxId1 as follows:

RxId1	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
$540	$11	$22	$33	$44	$55	-	-	-

Table 4.1.2: Transmitting the data to be transmitted via Rx-identifier RxId1

Bytes 6...8 are not required.

4.1.2.3 Receiving Data from the Serial Interface

Via Tx-identifier TxId1 the data received by the serial interface is transmitted by the module to the CAN. The module is run by the default parameters and therefore each received byte is transmitted at once. The delay between the start of the individual transmissions by the CAN controller is 20 ms.

In this example the device connected is to transmit a block of eight bytes with the following contents:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
$FF	$EE	$DD	$CC	$BB	$AA	$99	$88

Now the module transmits the individual bytes of this block via the Tx-identifier with delays of about 20 ms. If the CAN is handling messages of a higher priority, the delays between the transmissions can be longer.

TxId1	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
$541	$FF	-	-	-	-	-	-	-
	t ≥ 20ms							
TxId1	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
$541	$EE	-	-	-	-	-	-	-
	t ≥ 20ms							
TxId1	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
$541	$DD	-	-	-	-	-	-	-
...								
	t ≥ 20ms							
TxId1	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
$541	$88	-	-	-	-	-	-	-

Table 4.1.3: Transmitting serial data received on the CAN

Bytes 2...8 of the individual Tx-transfers are not transmitted, because the value '1' is specified in default setting of user parameter *CAN-Tx-Mode* for *MaxChar.*

4.2 Changing the Baud Rate of the Serial Interface

The baud rates of the serial interface are to be changed from 9600 baud (default setting) to 19200 baud for receive and transmit data.

The module No. of the CAN-CBM-COM1 is to be $12 again.

The baud rate is changed in cells *Rx-Baudrate* and *Tx-Baudrate* of user parameter *Serial-Mode.* These two cells are in the first byte of the user parameter (in transmission = byte 5). The sub-command is $02.

Assigning baud rate 19200 results in value '$1' for each nibble of byte 5 of the user parameter.

Byte 6 of the parameter (*Serial-Mode*) is not to be changed, has to be transmitted as well, like in all user parameters! Therefore, the default value $73 is specified here.

CAN-Id	Byte 1 *(Command)*	Byte 2 *(Sub-Command)*	Byte 3 (always $00)	Byte 4 *(Module-No.)*	Byte 5 *(Baudrate)*	Byte 6 *(Serial Mode)*
$700 =INIT-Id	$86	$02	$00	$12	$11	$73

Table 4.2.1: Changing the baud rates of the serial interface

Bytes 7 to 8 are not required for this command.

The changed baud rate becomes active as soon as the user parameter has been received.