SER1B – serial interfaces (Update)

Blank Page

Index

SER1B – serial interfaces (Update)

User function codes of SER1B_xx.TDD

User-Function-Codes (UFC) for the input instruction GET:

No	Symbol Prefix UFCI_	Description
1	UFCI_IBU_FILL	No. of bytes in input buffer (Byte)
2	UFCI_IBU_FREE	Free space in input buffer (Byte)
3	UFCI_IBU_VOL	Size of input buffer (Byte)
33	UFCI_OBU_FILL	Number of bytes in output buffer (Byte)
34	UFCI_OBU_FREE	Free space in output buffer (Byte)
35	UFCI_OBU_VOL	Size of output buffer (Byte)
65	UFCI_LAST_ERRC	Last error code
99	UFCI_DEV_VERS	Driver version
139	SER1_SET_OUTPUT	read out status of TxD Bit 1: TxD 0 Bit 5: TxD 1 0: TxD 1: Output
144	UFCI_SER_STAT	returns in WORD: low byte: number of receive errors high byte: number of buffer overflows
145	UFCI_SER_9STS	status when running 9-bit: 0: waiting for address 1: receiving data
146	UFCI_SER_9ADR	most recent received address
192	UFCI_SER_TX_ACT	Read out current transmit state YES (0): transmit active No (255): transmit inactive
193	UFCI_SER_TX_LOCK	Read out lock state 0: buffer is unlocked 1: buffer locked

User-Function-Codes for output (instruction PUT):

No	Symbol Prefix: UFCO_	Description
1	UFCO_IBU_ERASE	erase input buffer
33	UFCO_OBU_ERASE	erase output buffer
65	UFCO_ERRC_RESET	reset most recent OK-/WARNING-/ERROR-Code
94	UFCO_SET_SERIAL	set serial parameter
128	UFCO_SET_ISEP	set limiter characters for instruction INPUT
129	UFCO_RES_ISEP	delete limiter characters for INPUT
130	UFCO_SER_ECHO	generate echo chars into the output buffer (YES/NO)
131	UFCO_SER_9BIT	9-bit mode only: set 9th bit to ‘0’ or ‘1’
132	UFCO_SER_9ADR	9-bit mode only: set address of this module value 0...0FFh. Setting 100h clears the address.
133	UFCO_SER_9RTS	set RTS to ’0’ = not ready ’1’ = ready
136	UFCO_SER_XONXOFF	turn software handshake XON/XOFF for channel 0 or 1 on or off 0 = XON/XOFF active 255 = XON/XOFF inactive
137	UFCO_SER_XSEND	send ‹XON› or ‹XOFF› char immediately

No	Symbol Prefix: UFCO_	Description
138	UFCO_SER_XLIMITS	set thresholds for XON/XOFF ‹WORD› = Buffer Threshold: exactly at this amount of "free-buffer-space" the receive buffer is closed (send one ‹XOFF›) ‹WORD› = Buffer-Hysteresis: Area for the re-OPEN and "forced" close modes 1. Buffer OPEN (Free space decreases) a) Exactly at FREE = "THRESHOLD" one XOFF is sent, Buffer = CLOSE b) If less than "THRESHOLD" – "HYST", answer with 1 x XOFF on every char c) When buffer is full, characters are lost and every char is answered with XOFF 2. Buffer CLOSE (Free spaces increases) When exceeding "THRESHOLD" + "HYST", one XON is sent, Buffer = OPEN
139	SER1_SET_OUTPUT	set TxD to Output 0: LOW Else: High
140	SER1_SET_TXD	set Output pin to TxD, standard serial interface can be used again.
141	SER1B_CTS	0: disables CTS 1: enables CTS
142	UFCO_SET_RTS	sets Port & Pin for RTS
193	UFCO_SER_TX_LOCK	Locks transmit buffer until 9. bit is set 0: unlock buffer 1: lock buffer

Baudrates:

Nr.	Symbol	Meaning	BASIC-Tiger TINY-Tiger Econo-Tiger	TINY-Tiger 2
0	BD_50	50 Bd		
1	BD_75	75 Bd		
2	BD_110	110 Bd		
3	BD_150	150 Bd		
4	BD_200	200 Bd		
5	BD_300	300 Bd	available	available
6	BD_600	600 Bd	available	available
7	BD_900	900 Bd		available
8	BD_1_200	1,200 Bd	available	available
9	BD_1_800	1,800 Bd		available
10	BD_2_400	2,400 Bd	available	available
11	BD_3_600	3,600 Bd		available
12	BD_4_800	4,800 Bd	available	available
13	BD_7_200	7,200 Bd		available
14	BD_9_600	9,600 Bd	available	available
15	BD_14_400	14,400 Bd		available
16	BD_19_200	19,200 Bd	available	available
17	BD_28_800	28,800 Bd		available
18	BD_38_400	38,400 Bd	available	available
19	BD_57_600	57,600 Bd		available
20	BD_76_800	76,800 Bd	available	available
21	BD_115_200	115,200 Bd		available
22	BD_153_600	153,600 Bd	available	available
23	BD_230_400	230,400 Bd		
24	BD_307_200	307,200 Bd		available
25	BD_460_800	460,800 Bd		
26	BD_614_400	614,400 Bd		available
32	BD_31_250	31,250 Bd	available	available
33	BD_62_500	62,500 Bd	available	available

Nr.	Symbol	Meaning	BASIC-Tiger TINY-Tiger Econo-Tiger	TINY-Tiger 2
34	BD_EXT	external Oscillator / 16 Connect to CTS pin		available
35	BD_10_400	10,400 Bd		available
36	BD_41_600	41,600 Bd		available
37	BD_100_000	100,000 Bd		available
38	BD_26_000	26,000 Bd		available

Disable CTS

You can disable the function of the CTS Pin with the User Function Code *SER1B_CTS*!

PUT #D, #0, # SER1B_CTS, a

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

a is a constant, a variable or expression of the data type BYTE, WORD or LONG and determines functionality of the CTS pin.
0: disables the CTS pin
1: enables the CTS pin

Program sample:

```
user_var_strict

#INCLUDE DEFINE_A.INC              ' Definitions
#INCLUDE UFUNC4.INC                ' User Function Codes

TASK MAIN
  BYTE EVER                                      ' variable for endless loop

  INSTALL_DEVICE #SER, "SER1B_K1.TDD", &         ' install SER-driver
    BD_9_600, DP_8N, YES, &                      ' settings SER0
    BD_9_600, DP_8N, YES                         ' settings SER1

  PUT   #SER, #0, #SER1B_CTS, 0                  ' <=== disable CTS on SER-0
'  PUT  #SER, #0, #SER1B_CTS, 1                  ' <=== enable CTS on SER-0


  FOR EVER = 0 TO 0 STEP 0                       ' endless loop
    PUT #SER, "abcd"                             ' send data on SER-0
  NEXT

END
```

Use TxD as Output

While the TxD Pin is used as an output pin, no data can be PUT to serial channel!

TxD can ONLY be set to output, if the output buffer of this channel is EMPTY!

PUT #D, #ch, #SER1_SET_OUTPUT, a

D	is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.
ch	is a constant, a variable or expression of the data type BYTE, WORD, LONG and determines the channel of the serial interface (0 or 1).
a	is a constant, a variable or expression of the data type BYTE, WORD, LONG or STRING and determines the output state. 0: Low Else: High.

The output buffer of the selected channel has to be empty, before this operation is started. After this command, TxD is not available.

PUT #D, #ch, #SER1_SET_TXD, dummy

D	is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.
ch	is a constant, a variable or expression of the data type BYTE, WORD, LONG and determines the channel of the serial interface (0 or 1).
dummy	is a dummy ☺

This command activates the transmit pin of the serial interface again. TxD is available now.

GET #D, #ch, #SER1_SET_OUTPUT, Number, Variable

D is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

ch is a constant, a variable or expression of the data type BYTE, WORD, LONG and determines the channel of the serial interface (0 or 1).

Number is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output.

Variable is a variable of the data type BYTE, WORD, LONG or STRING which contains the status of the TxD pins.

Statusbyte:

Bit No.	Description
0	---
1	TxD0 (1: output 0: ser)
2	---
3	---
4	---
5	TxD1 (1: output 0: ser)
6	---
7	---

Program example:

```
USER_VAR_STRICT                          '
#INCLUDE DEFINE_A.INC                    '
#INCLUDE UFUNC4.INC                      ' User Function Codes
#define channel 1


TASK MAIN
  LONG A                                 '
  BYTE B                                 '
  STRING S$(64)                          '

  INSTALL_DEVICE #SER, "SER1B_K1.TD2",&
  BD_38_400, DP_8N, YES, &
  BD_38_400, DP_8N, YES

  install_device #0, "lcd1.tdd"


  while 1=1
    FOR A = 0 to 100
      PRINT   #SER,#channel, "Loop start"
      GET       #SER, #channel, #SER1_SET_OUTPUT, 0, B' read out status
                                                    ' BIT1: TX0
                                                    ' 0: TxD0   1: Output
                                                    ' BIT5: TX1
                                                    ' 0: TxD0   1: Output

      PRINT   #SER,#channel, "Status TxD: " ; B          ' Print to SER

      wait_duration 1000
      PUT       #SER, #channel, #SER1_SET_OUTPUT, 0     ' LOW
      PUT       #0, "LOW"
      wait_duration 1000
      PUT       #SER, #channel, #SER1_SET_OUTPUT, 1     ' HIGH
      PUT       #0, ",HIGH"
      wait_duration 1000
      PUT       #SER, #channel, #SER1_SET_OUTPUT, 0     ' LOW
      PUT       #0, ",LOW"
      wait_duration 1000
      PUT       #SER, #channel, #SER1_SET_OUTPUT, 1     ' HIGH
      PUT       #0, ",HIGH"
      wait_duration 1000

      GET       #SER, #channel, #SER1_SET_OUTPUT, 0, B  ' read status
      PUT       #SER, #channel, #SER1_SET_TXD, 0        ' set TxD

      PRINT     #SER,#channel, "Status TxD: " ; B    ' show status (in Loop)
      PRINT     #SER,#channel, "End of Loop "        '
    NEXT
  endwhile

END
```

Enabling pins as I/O pins

It is possible not to use all pins of the serial ports completely, but to use some as digital I/O's, if, e.g. no handshake is needed, one wishes to only receive/send, or when only one port is needed. The parameters can be found in the chart further down.

INSTALL DEVICE #*D*, "SER1B_K1.TD2" [, *P1, ..., P12*]

D is a variable, a constant, or an expression of the data type BYTE, WORD, LONG in the range between 0...63 and stands for the device number of the driver.

	Keep default	Description of the parameter
P1	0EEH	is a parameter for setting the Baud rate, channel 0
P2	0EEH	is a parameter for setting the number of data bits and the parity, channel 0
P3	0EEH	= NO: characters are suppressed, that are recognized as faulty by the hardware = YES: passes on probably incorrectly received characters to the receive buffer, channel 0
P4	0EEH	is a parameter for setting the Baud rate, channel 1
P5	0EEH	Is a parameter for setting the number of data bits and the parity, channel 1
P6	0EEH	= NO: characters are suppressed, that are recognized as faulty by the hardware = YES: passes on probably incorrectly received characters to the receive buffer, channel 1
P7a P7b P7c	- - 0EEH	= 0AAH (fixed value) = Bit mask for pins of the serial interface (see table further down) = Bit mask for transmit-enable pin, channel 0 (only RS-485)
P8	0	Logical port address for transmit-enable, channel 0 (only RS-485)
P9	0EEH	0: Transmit-enable is high-level 1: Transmit-enable is low-level (only RS-485)
P10	0EEH	Bit mask for transmit-enable-pin, channel 1(only RS-485)
P11	0	Logical port address for transmit-enable, channel 1 (only RS-485)
P12	0EEH	0: Transmit-enable is high-level 1: Transmit-enable is low-level (only RS-485)

Bit mask for pins of the serial interface:

Bit No.	Description
0	RxD0 (1: enable, 0: disable)
1	TxD0 (1: enable, 0: disable)
2	RTS0 (1: enable, 0: disable)
3	CTS0 (1: enable, 0: disable)
4	RxD1 (1: enable, 0: disable)
5	TxD1 (1: enable, 0: disable)
6	RTS1 (1: enable, 0: disable) (Tiger 2 only)
7	CTS1 (1: enable, 0: disable) (Tiger 2 only)

Program sample:

```
TASK MAIN

  INSTALL DEVICE #1, "LCD1.TD2"
  INSTALL_DEVICE #2, "SER1B_K1.TD2",&    ' install SER1-driver
  BD_38_400, DP_8N, YES, &               ' setting SER0
  BD_38_400, DP_8N, YES, &               ' setting SER1
  0AAH, 00110000B                        ' <== new parameters (P7a + P7b)


  dir_pin 9,0,0          ' TxD0 output (==> used as I/O)
  dir_pin 9,1,0          ' RxD0 output (==> used as I/O)
  dir_pin 9,2,0          ' CTS0 output (==> used as I/O)
  dir_pin 9,5,0          ' RTS0 output (==> used as I/O)

  run_task blink_led     ' here the SER0 lines are used as output


  PRINT #1,"<1>SER1B.tig"              ' show program name

while 1=1
  put #2,#1, " PUT "                             ' use SER1
  print #2,#1, " PRINT "                         ' use SER1
endwhile

end



task blink_led

  while 1=1

    OUT 9, 00100111B, 255        ' Pin high
    wait_duration 1000           '


    OUT 9, 00100111B, 0          ' Pin low
    wait_duration 1000           '

  endwhile

end
```

Change RTS0 pin

You can change the RTS pin of the serial interface with User Function Code *UFCO_SET_RTS*! This can be helpful for the use of Econo-Tiger™.

PUT #D, #0, #UFCO_SET_RTS, Port, Pin

D	is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.
Port	is a constant, a variable or expression of the data type BYTE and determines the Port of the new RTS0 pin.
Pin	is a constant, a variable or expression of the data type BYTE and determines pin no. of the new RTS0 pin

SER1B expects exactly 2 Bytes !!!

Sets RTS0 to Port 8 Pin 1 (L81):

```
PUT     #SER, #0, #UFCO_SET_RTS, 8, 1
```

Use of RS-232 and RS-485 together

To use SER-0 as RS-232 and SER-1 as RS-485, please use the standard settings for SER-0. This is a little example to install SER1B for this purpose:

```
INSTALL_DEVICE #SER, "SER1B_R02.TDD",&
BD_19_200, DP_8N, JA, BD_19_200, DP_8N, JA,0EEH,0,0,00000001b,8,0
```

Using an external oscillator

You can generate your own special baud rates with an external oscillator. The frequency of the oscillator is divides by 16. Please connect the oscillator to the CTS Pin of the serial interface and use the baud rate *BD_EXT*.

```
INSTALL_DEVICE #SER, "SER1B_R02.TDD",&
BD_EXT, DP_8N, JA, BD_19_200, DP_8N, JA
```

Using XON/XOFF software handshake

XON/XOFF software handshake uses special codes, transmitted in-band, over the primary communications channel. These codes are generally called XOFF and XON (from "transmit off" and "on", respectively). This is in contrast to flow control via dedicated out-of-band signals (hardware handshake) such as RTS/CTS. To use XON/XOFF software handshaking on SER-0 or SER-1, there are three User-Function-Codes available.

The first and most important is for activating/deactivating the XON/XOFF software handshake mode:

```
PUT #SER, #0, #UFCO_SER_XONXOFF, 0        ' activate XON/XOFF handshake
PUT #SER, #0, #UFCO_SER_XONXOFF, 255      ' deactivate XON/XOFF handshake
```

You can at any time send a XON or XOFF character immediately. Even if there are characters in the output buffer still to be sent, the XON (ASCII char 11h) or XOFF (ASCII char 13h) are sent out at once, in between the normal data stream:

```
PUT #SER, #0, #UFCO_SER_XSEND, "<11H>"    ' send XON character
PUT #SER, #0, #UFCO_SER_XSEND, "<13H>"    ' send XOFF character
```

The third User Function Code is for setting the limits when the input buffer is closed and re-opened (although physically data is still received in a closed buffer until buffer is full). Two word values are needed, the first is the initial value when the buffer is closed (Threshold), the second is the offset (Hysteresis) when the buffer is “forced” closed or re-opened:

```
                                '   Thres|Hyster
PUT #SER, #0, #UFCO_SER_XLIMITS, "80 00 20 00"%    ' set limits (128, 32)
```

The default values are 80 dec (50 hex) for Threshold and 60 dec (3C hex) for Hysteresis. Following is a sample of how the driver behaves when XON/XOFF handshake is active, using the default values for the limits:

Input:	Free Buffer:	Output:	Meaning for buffer & remote:
any char	82	–	Buffer open, no problem
any char	81	–	Buffer open, no problem
any char	80	‹13h›	Buffer is closed, please stop sending
any char	79	–	Buffer still closed
…			
any char	21	–	Buffer still closed
any char	20	‹13h›	Buffer forced closed, stop sending now!
any char	19	‹13h›	Buffer forced closed, stop sending now!
…			
any char	1	‹13h›	Buffer forced closed, stop sending now!
any char	0	‹13h›	Buffer full, stop sending now!
any char	0	‹13h›	Character is lost!
…			
e.g. 160 chars are read from the buffer			
–	160	‹11h›	Buffer is open again, you can send again
any char	159	–	Buffer open, no problem

Now the complete process restarts when free buffer space is down to 80 again.

Read out transmit state

Even if the output buffer is empty, it is possible that the serial interface sends the last Byte, because the buffer of the device driver is already empty, but the last Byte is still in the transmit buffer of the serial interface. To ensure, that there is no transmit activity, please use the User Function Code *UFCI_SER_TX_ACT*

GET #D, #ch, #UFCI_SER_TX_ACT, Number, Variable

D	is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.
ch	is a constant, a variable or expression of the data type BYTE, WORD, LONG and determines the channel of the serial interface (0 or 1).
Number	is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output.
Variable	is a variable of the data type BYTE, WORD, LONG or STRING which contains the transmit status YES (0): transmit active NO (255): transmit inactive

wait for empty buffer and completed transmission:

```
blIsAct = YES
llObuFill = 1
while blIsAct = YES or llObuFill > 0
  GET #SER, #0, #UFCI_OBU_FILL, 0, llObuFill
  GET #SER, #0, #UFCI_SER_TX_ACT, 0, blIsAct
endwhile
```

Multi Drop Bus / Vending

The Multi Drop Bus / Internal Communication Protocol is a registered trademark of the National Automatic Merchandising Association (NAMA®).

There is a special operation mode to use the Multi Drop Bus (Vending) with the SER1B device driver. The Multi Drop Bus is a kind of RS-485 bus, but with a different use of the ninth bit and a specific timing. Also the hardware levels differ from the RS-485. To activate the Multi Drop Bus, please select DP_9MDB for data / parity. In the following example we will use SER-0 as Multi Drop Bus, e.g. at the TP1000:

```
install_device #SER, "SER1B_K4.TD2", &
                     BD_9_600, DP_9MDB, YES, &
                     BD_9_600, DP_8N, YES, &
                     00010000b, 1, 0
```

The ninth bit in this mode is no address indicator, but a so called mode bit. The mode bit is used for different purposes, so the ninth bit is written into the buffer, too. For every received data, 2 Bytes are reserved in the input buffer. **The first Byte indicates the status of the mode bit (0 or 1), the second byte is the 8-bit data word.** Ensure that the input buffer is filled at least with 2 Bytes and read out the buffer with a length multiple of 2.

```
get #SER, #SER_CHANNEL, #UFCI_IBU_FILL, 0, llIbuFill
if llIbuFill >= 2 then
  get #SER, #SER_CHANNEL, 2, slReceive$
  blModeBit = NFROMS(slReceive$, 0, 1)
  blData = NFROMS(slReceive$, 1, 1)
endif
```

The same applies to the transmission buffer. Each Databyte needs a prefixed mode bit, which is saved in an extra data byte. **It is necessary to pass 2 Bytes to the transmission buffer of the device driver to send 1 byte.** The first Byte indicates the status of the mode bit (0 or 1), the second byte is the 8-bit data word.

```
send$ =  "<0><055H>"            ' 1. Byte = Modebit, 2. Byte = Data
send2$ = "<1><055H>"            ' 1. Byte = Modebit, 2. Byte = Data
put #SER, #SER_CHANNEL, send$   ' send with Mode-Bit = 0
put #SER, #SER_CHANNEL, send2$  ' send with Mode-Bit = 1
```

The Multi Drop Bus specifies a fast acknowledge timing. After receiving a complete packet, which is determined with a set mode bit, this message must be acknowledged (or NAK) within 5ms. To ensure this timing, there is the User Function Code *UFCO_SER_TX_LOCK*. With *UFCO_SER_TX_LOCK* you can lock the output buffer

by passing a 1, which means the transmission will not be started. The bytes written to the output buffer just prepared to send. After receiving data with set mode bit, this lock will be disabled automatically and the transmission is started. **Please ensure that there is no active transmission before the buffer is locked.**

```
'------------------------------------------------------------------
' wait until transmission is completely finished
'------------------------------------------------------------------
tx_active = 0
while tx_active = 0
  GET #SER, #SER_CHANNEL, #UFCI_SER_TX_ACT, 0, tx_active
endwhile
'------------------------------------------------------------------

put #SER, #SER_CHANNEL, #UFCO_SER_TX_LOCK, 1        ' lock buffer
put #SER, #SER_CHANNEL, send$                       ' prepare output

'------------------------------------------------------------------
' wait for received 9. bit (transmission will start automatic)
'------------------------------------------------------------------
obu_fill = 1
while obu_fill > 0
  GET #SER, #SER_CHANNEL, #UFCI_OBU_FILL, 0, obu_fill
endwhile
'------------------------------------------------------------------
```

The buffer can be unlocked manually every time by passing a 0 to *UFCO_SER_TX_LOCK.* If the output is filled, the transmission will be started.

```
put #SER, #SER_CHANNEL, #UFCO_SER_TX_LOCK, 0    ' unlock buffer
```

You can read out the current lock state with the User Function Code *UFCI_SER_TX_LOCK*. Reading a 0 means that the buffer is not locked, otherwise the output buffer is still locked and no set mode bit was received yet.

```
GET #SER, #SER_CHANNEL, #UFCO_SER_TX_LOCK, 0, lock_state
```

Documentation History

Version of Documentation	Version of SER1B	Description / Changes
001	1.03a	- first version
002	1.03e	New baudrates: - 41.600 - 100.000
003	1.03e	Baud rates revised
004	1.03g	XON/XOFF software handshake added
005	1.03h	New User Function Code UFCI_SER_TX_ACT
006	1.03i	Multi Drop Bus / Vending
007	1.03j	New baudrate 26.000 Bd
008	1.03j	Telephone number changed