NOTICE

BMC Communications Corp. reserves the right to change the product described in this document as well as the document itself at any time and without notice.

DISCLAIMER

BMC COMMUNICATIONS CORP. MAKES NO WARRANTIES, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS DOCUMENT OR WITH RESPECT TO THE PRODUCT DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PRSULAR PURPOSE. IN NO EVENT SHALL BMC COMMUNICATIONS CORP. BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE PRODUCT.

Rev. 3.3 February 10, 2007

Table of Contents

General Board RS232 Functions (GSR):

UART MEMORY CONTROL SEGMENT

X7DD0	GSR_232	X7DD0	Com_Param [2]	WORD
		X7DD4	INT_Status_Tx_Rx	WORD
		X7DD6	Tx_Count [2]	WORD
		X7DDA	Rx_Count [2]	WORD
		X7DDE	Delay	WORD
X7FE8	Tx_ctl_232 [2]	X7DE0	Command	WORD
		X7DE2	Start_Addr	WORD
	CHANNEL 0	X7DE4	Word_Count	WORD
		X7DE6	Status	INT.
		X7DE8	Command	WORD
	CHANNEL1	X7DEA	Start_Addr	WORD
		X7DEC	Word_Count	WORD
		X7DEE	Status	INT.
X7FF0	Rx_ctl_232 [2]	X7DF0	Command	WORD
		X7DF2	Start_Addr	WORD
	CHANNEL 0	X7DF4	Buffer_Size	WORD
		X7DF6	Status	INT.
		X7DF8	Command	WORD
	CHANNEL1	X7DFA	Start_Addr	WORD
		X7DFC	Buffer_Size	WORD
		X7DFE	Status	INT.

rs232_act_tx_chan

DESCRIPTION

Active Transmitter channel number

USAGE

```
#include <uadi32.h>
int rs232_act_tx_chan (channel);
int channel; channel number  1 or 2
```

REMARKS

This routine checks if the specified number represents an active Transmitter. Channel on the currently selectedseiected board by reading the board's Channel Active Register.

RETURN VALUE

YES Valid channel
NO Not valid channel

EXAMPLE

```
/* count # of valid channels on the board */

for (chan=1; chan<3; chan++) { // channels 1 or 2

if (rs232_act_tx_chan(chan)) {

++fl;

}
}
printf (%d Transmitter Channels foundn",n);
```

SEE ALSO

rs232_act_rx_chan

rs232_act_rx_chan

DESCRIPTION

Active Receiver channel number

USAGE

```
#include <uadi32.h>

int rs232_act_rx_chan (channel);

int channel; channel number – 1 or 2
```

REMARKS

This routine checks if the specified number represents an active Receiver Channel on the currently selected board by reading the boards Channel Active Register.

RETURN VALUE

YES Active channel

NO Not Active channel

EXAMPLE

```
/* Try to receive 10 words on every available channel l

RS232_WORD buffer[4][10]; 1* 4 buffers of 10 word each */

for (chan=0; chan<2; chan++) {
if (rs232_act_rx_chan(chan)==NO) {
rs232_rx_cmd(chan, RX_NORMAL, buffer[channel][0); }
}
```

SEE ALSO

rs232_act_tx_chan

rs232_datarcvd

DESCRIPTION

Check if new data received

USAGE

#include <uadi32.h>

int rs232_datarcvd (channel)

int channel; receive channel number

REMARKS

This function checks the appropriate bit in the boards Status Register. This bit is set by the RS board in all receive modes when any new data (at least one word) has been received since the last time this bit was polled. After reading the bit, the function clears it for the next operation. This function should be used in a polled (non-interrupt) fassion to check if new data was received. In interrupt mode, it should be called from the Interrupt Service Routine to identify the source of the interrupt.

RETURN `UE

YES New data received

NO No new data since last time checked

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

```
        /* polled receive operation: wait for 10 RS words on channel 1 */

/* and display them when done: */
RS232 rcv_buff[10];
int i;


rs232_rx_cmd( 1, RX_NORMAL, Ox0000, 10); /* start receiving */
while (rs232_datarcvd(1 )=NO); 1* wait for incomming word */
rs_readbuf(rcv_buff,Ox0000, 10*4); /*read data from board into our buffer*/ *
display data on screen: *1
for (i=0; i<10; i++)
prinff("Data word #%d = %0IXn, i+1, rcv_buff[i]);
```

SEE ALSO

rs232_ei_datarcvd

rs232_rcvbuf_full

rs232_halfbuf_full

rs232_hallbuff_full

DESCRIPTION

Check if new data received

USAGE

#include <uadi32.h>

int rs232_halfbuff_full (channel)

int channel; receive channel number 1 or 2

REMARKS

This function checks the appropriate bit in the boards Status Register. This bit is set by the RS board in all receive modes when half of a predefined buffer size has been received since the last time this bit was polled. After reading the bit, the function clears it for the next operation. This function should be used in a polled (non-interrupt) fassion to check if new data was received. In interrupt mode, it should be called from the Interrupt Service Routine to identify the source of the interrupt.

RETURN `UE

YES Half buffer full

NO Half buffer not full

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

SEE ALSO

rs232_ei_datarcvd

rs232_rcvbuf_full

rs232_halfbuf_full

rs232_halfbuff

DESCRIPTION

Disable Interrupt on New Data Received

USAGE

#include <uadi32.h>

int rs232_di_datarcvd (channel)

int channel; the receive channel number

REMARKS

This function clears the appropriate bit in the appropriate interrupt condition register which will prevent the board from issuing an IRQ (interrupt request) when the following condition is met: New data (at least one new word) received on the channel.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

rs232_ei_datarcvd

ei_rs232_rcvbuf_full

di_rs232_rcvbuf_full

rs232_di_halfbuf_full

DESCRIPTION

Disable Interrupt on Receive Buffer half full

USAGE

#include <uadi32.h>

int rs232_di_halfbuf_full (channel)

int channel; the receive channel number

REMARKS

This function clears the appropriate bit in the appropriate interrupt condition register which will prevent the board from issuing an IRQ (interrupt request) when the following condition is met: the flow of incoming data just passed the half-length mark of the Receive Buffer.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

ei_rs232_halfbuf_full

rs232_di_datarcvd rs232_ei_datarcvd

rs232_ei_rcvbuf_full di_rs232_rcvbuf_full

rs232_rcvbuf_full

DESCRIPTION

Disable Interrupt on Receive Buffer Full

USAGE

#include <uadi32.h>

int rs232_rcvbuf_full (channel)

int channel; the receive channel number

REMARKS

This function clears the appropriate bit in the appropriate interrupt condition register which will prevent the board from issuing an IRQ (interrupt request) when the following condition is met: the last location of the Receive Buffer has been written with new data.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

ei_ei_rcvbuf_full **rs232_di_datarcvd**

rs232_ei_datarcvd

ei_rs232_halfbuf_full **rs232_di_halfbuf_full**

rs232_di_txcomplete

DESCRIPTION

Disable Interrupt on Transmit complete

USAGE

#include <uadi32.h>

int rs232_di_txcomplete (channel)

int channel; the transmit channel number

REMARKS

This function clears the appropriate bit in the appropriate interrupt condition register which will prevent the board from issuing an IRQ (interrupt request) when the following condition is met: the entire contents of the buffer specified in a Transmit Command has been transmitted to the RS channel.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

rs232_di_datarcvd

ei_rs232_rcvbuf_full

di_rs232_rcvbuf_full

ei_rs232_rcvbuf_full

rs232_ei_datarcvd

DESCRIPTION

Enable an Interrupt on new data received

USAGE

#include <uadi32.h>

int rs232_ei_datarcvd (channel)

int channel; the receive channel number

REMARKS

This function sets the appropriate bit in the appropriate interrupt condition register which will cause the board to issue an IRQ (interrupt request) when the following condition is met: One or more new data word(s) have been received on the channel.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

rs232_di_datarcvd

rs232_ei_rcvbuf_full

di_rs232_rcvbuf_full

rs232_ei_rs_halfbuff

DESCRIPTION

Enable Interrupt on Receive Buffer half full

USAGE

#include <uadi32.h>

int ei_rs232_halfbuf_full (channel)

int channel; the receive channel number

REMARKS

This function sets the appropriate bit in the appropriate interrupt condition register which will cause the board to issue an IRQ (interrupt request) when the following condition is met: the flow of incoming data just passed the half-length mark of the Receive Buffer. This interrupt condition is useful for handling large bursts of received data. Getting an interrupt at the half-buffer mark allows to read the first half while the second half is filling up. Upon receiving the Buffer Full interrupt, the second half of the buffer can be read while the first half is filling up, and so-on.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

di_rs232_di_halfbuf_full rs232_di_datarcvd

rs232_ei_datarcvd ei_rs232_rcvbuf_full

di_rs232_rcvbuf_full

rs232_ei_rcvbuf_full

DESCRIPTION

Enable Interrupt on Receive Buffer Full

USAGE

#include <uadi32.h>

int rs232_ei_rcvbuf_full (channel)

int channel; the receive channel number

REMARKS

This function sets the appropriate bit in the appropriate interrupt condition register which will cause the board to issue an IRQ (interrupt request) when the following condition is met: the last location of the Receive Buffer has been written with new data.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

rs232_di_rcvbuf_full **rs232_di_datarcvd**

rs232_ei_datarcvd **ei_rs232_halfbuf_full**

rs232_di_halfbuf_full

rs232_ei_tx_complete

DESCRIPTION

Enable Interrupt on Transmit complete

USAGE

#include <rsh>

int ei_rs232_tx_complete (channel)

int channel; the transmit channel number

REMARKS

This function sets the appropriate bit in the appropriate interrupt condition register which will cause the board to issue an IRQ (interrupt request) when the following condition is met: the entire contents of the buffer specified in a Transmit Command has been transmitted to the RS channel.

RETURN VALUE

RS_SUCCESS

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

RS232 tx_buff[500];

```
intr_handler 0
{
        if (rs txcomplete(channel)) {
        }
}

main()
{
rs_setup_intr( intr_handler); rs232_ei_txcomplete(channel);
```

```
rs232_writebuf(Ox0000,tx_buff,500); I transfer data to board*I
rs232_tx_cmd(channel,RS_TX_NORMAL, Ox0000, 500,0);
r when transmit buffer completed intr_handler will */
/* be activated from the IRQ. */
}
```

SEE ALSO

rs232_di_txcomplete

rs232_halfbuf_ full

DESCRIPTION

Check if Receive Buffer is half full

USAGE

#include <uadi32.h>

int rs232_halfbuf_full (channel)

int channel; receive channel number

REMARKS

This function checks the appropriate bit in the board's Status Register. This bit is set by the RS board in all receive modes when the flow of incoming data just passed the half-length mark of the Receive Buffer. After reading the bit, the function clears it for the next operation. This function should be used in a polled (non-interrupt) fassion to check if half-buffer is full was received. In interrupt mode, it should be called from the Interrupt Service Routine to identify the source of the interrupt.

RETURN VALUE

YES Half-buffer full

NO Half-buffer mark not reached since last checked

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

ei_rs232_halfbuf_full

rs232_rcvbuf_full

rs232_datarcvd

rs232_parity

DESCRIPTION

Set parity

USAGE

#include <uadi32.h>
rs232_parity (short channel, short ena, short even)

int channel; the receive channel number (1-2)
int ena: enable parity – default ODD
int even: set even parity

REMARKS

This routine seiects the parity for the specified channel.

RETURN VALUE

RS_SUCCESS Successful

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

SEE ALSO

rs232_rcvbuf_full

DESCRIPTION

Checks if Receive Buffer if full

USAGE

#include <uadi32.h>

int rs232_rcvbuf_full (channel)

int channel; receive channel number

REMARKS

This function checks the appropriate bit in the boards Status Register. This bit is set by the RS board in both Normal and Continuous modes when the last location in the receive buffer was written by RS with a new data word. After reading the bit, the function clears it for the next operation. This function should be used in a polled (non-interrupt) fassion to check if the buffer is full. In interrupt mode, it should be called from the Interrupt Service Routine to identify the source of the interrupt.

RETURN VALUE

YES Receive fuffer is full

NO Recive buffer not full

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

```
        /* polled receive operation: wait for 10 RS words on channel 1 */

l and display them when done: */
RS232 rcv_buffll 0];
int i;

rs232_rx_cmd(1,RX_NORMAL,Ox0000,1O); /* start receiving */
while (rs232_rcvbuf_full(1 )== NO); /* wait for receive complete */
/* receive complete: */
/* read data from board into our buffer: */
rs_readbuf(rcv_buff,Ox0000, I O4);
/* display data on screen: */
for (1=0; i<10; i++)
printf("Data word #%d = %0lXn", i+1, rcv_buff]);
```

SEE ALSO

ei_rs232_rcvbuf_full

rs232_halfbuf_full

rs232_datarcvd

rs_readbuf

DESCRIPTION

Read a section of RS memory

USAGE

#include <uadi32.h>

int rs_readbuf (dst,src,size)

void *dest; destination address

void far *src; address of data to read from

int size; number of bytes to be read starting from src

REMARKS

This function is used to transfer a block of data from the RS board Dual-Ported memory into program memory. The most typical use is to transfer Receive Data from the board after receiving data. This function should be used rather than direct memory operations in order to maintain compatibility with drivers for protected-memory operating systems.

RETURN VALUE

RS_SUCCESS Transfer complete

EXAMPLE

SEE rs232_rx_cmd example

SEE ALSO

rs232_writebuf

DESCRIPTION

Uninstall interrupt service routine - DOS

USAGE

#include <rsh>

int rs_restore_intr( )

DESCRIPTION

This function uninstalls the RS interrupt service routine and restores the vector which was originally hooked to the RS IRQ.

RETURN VALUE

RS_SUCCESS

SEE ALSO

rs_setup_intr

rs_set_irq

rs232_rx_cmd

DESCRIPTION

Send a Receive Command to an RS board

USAGE

#include <uadi32.h>

int rs232_rx_cmd (channel, command, start, bufsize)

int channel; The receive channel number (1-2)

WORD command; Command code: RS_RX_NORMAL,
RS_RX_CONTNS, or
RS_RX_STOP.
WORD start Location (offset from board base addresss)
WORD bufsize; /* Size of receive data buffer in bytes

REMARKS

This routine sets-up a Receive Command in the board's Receive Channel Control Block and waits for the board's response. If command is equal to RS_RX_NORMAL, RS will start receiving and will fill the supplied buffer (count words starting at offset start) with the incoming data. When the buffer is full, i.e. count words were received, the board will stop receiving and set the channel's bit in the tx_rx_complete interrupt status register. (this bit can be polled and cleared via the rs232_rx_buff_full. If interrupt on receive complete is enabled (done via ei_rsrcvfull), an Interrupt Request (IRQ) will be sent. If command is equal to RS_RX_CONTNS, the board will receive in the continuous mode, i.e. it will operate like in NORMAL mode, but it will not stop receiving when the buffer is full, it will continue receiving and filling the buffer from the beginning, overwriting the oldest data. In this mode, the board will send an IRQ (if enabled) also when the buffer is half-full, in addition to when the buffer is full, to allow the application software to start transfering data before the buffer is full.
Each time a new word is received, the RS processor will set the channel's bit in the board's Data Received interrupt status byte (can be read via rs_data_rcvd). If interrupt on Data Received is enabled (via rs232_ei_datarcvd), the board will also send an IRQ.

If command is equal to RS_RX_STOP, the board will stop receiving (if it was receiving at all) on the channel.

RETURN VALUE

RS_SUCCESS command was accepted: execution started.

RS_INV_CHANNEL invalid or unconfigured channel specified.

RS_INV_ADDRESS start address does not fall in the O-fOO range.

RS_INV_SIZE count is such that the last data word would fall outside of the range.

RS_CHNL_BUSY

command=RS_RX_NORMAL, RS_RX_CONTNS or

RS_ TX_LOOP and the channel is already receiving.

RS_BOARD_RESP board did not respond to the command within the response interval defined by _RS_RESP_ TIMEOUT.

EXAMPLE

SEE RXNORMAL.C example program.

SEE ALSO

rs232_datarcvd
rs232_rx_count **ei_rs_rxcomplete**
ei_rs_data_rcvd

rs_rxcount

DESCRIPTION

Read Receive Data Counter

USAGE

```
#include <uadi32.h>

int rs232_rx_count (channel);

int channel; channel number
```

REMARKS

This function returns the number of data bytes received on the channel since the last Receive Command was given modulo 32,768.

RETURN VALUE

Data count (0 to 32,767) Received Data Counter.

RS_INV_CHANNEL Invalid or unconfigured channel.

EXAMPLE

```
/* CONTINUOUS mode receive operation interrupt handler: */ /* In this
mode the board sends an IRQ each time the buffer */
/* is full and each time the buffer is half-full. The only */
/* way to sense if we have a buffer full or half-full */
/* condition is the Receive Count: */
/* Let's assume the main program sent an RS_RX_CONTNS */
/* command with buffer size 200 words starting at Ox0000 */
/* and enabled interrupt on Receive Complete.
RS232 rx_buff {BUFFSIZE];/* can be as big as you want */
int I = 0;
void intr_handler()
{
```

```
if (rs232_rx_buf_full (channel) {
        if ((rs232_tx_count(channel)%200)> 100) {
        /* half buffer full - read 1st half: */
                rs_readbuf(&rx_buff[i], 0x0000, 100*4);
                i= i+100;
        }
        eise{
                /* 2nd half full, read it */
                rs_readbuf(&rx_buff], Ox0000+ 100,100*4);
                i= i+100;
        }
}
}
```

SEE ALSO

rs232_tx_count

rs232_clr_rxcount

DESCRIPTION

Reset Receive Data Counter

USAGE

#include <uadi32.h>

int rs232_rx_count (channel);

int channel; channel number

REMARKS

This function clear registers receive cpount.

RETURN VALUE

RS_INV_CHANNEL Invalid or unconfigured channel.

SEE ALSO

rs232_clr_tx_count

RS232_BR

DESCRIPTION

Set channel Receive-transmit Data Rate

USAGE

#include <uadi32.h>

rs232_br (short channel , short baud_rate);
int channel; the receive channel number (1-2)

REMARKS

This routine sets the Data Rate for the specified channel.
Baud rate value table:

Value	Baud Rate
0	100 KHZ
1	1.2 KHZ
2	2.4 KHZ
3	4.8 KHZ
4	9.6 KHZ
5	19.2KHZ
6	38.4KHZ
7	RESERVED

RETURN VALUE

RS_SUCCESS Successful

RS_INV_CHN Invalid or unconfigured channel

SEE ALSO

rs_set_board

DESCRIPTION

Select active board's address

USAGE

#include <uadi32.h>

int rs_set_board (address);

WORD address; the base segment of the board address

REMARKS

This routine tries to find an RS-429 board at the specified address. If so, it seiects it as the currently seiected board (global variable _rs_429_board). All subsequent functions and commands will refer to the board at this address.

RETURN VALUE

RS_SUCCESS Operation successful

RS_NO_BOARD No board found at specified address.

EXAMPLE

if (arlic_set_board(OxD000) '=RS_SUCCESS) {

```
prinff("Board not present\n");
}
eise {
rs_sys_id(&min, &maj);
prinif ("RS-429 Version %d.%d at 0000:0000",min,maj);
}
```

SEE ALSO

rs_sys_id

rs_set_irq

DESCRIPTION

DOS

Set IRQ (interupt request) number

USAGE

#define <uadi32.h>

int rs_set_irq (irq)

int irq; the IRQ number (3-7)

REMARKS

This function is used to teil the API which IRQ (interrupt request line) is tied to the board (via Jumper Block JP5). The number is loaded into the API global variable _RS_IRQ. When a user interrupt handler is install (rs_setup_intr), the API will tie the RS interrupt handler to the vector associated with the IRQ number.

RETURN VALUE

RS_SUCCESS

SEE ALSO

rs_setup_intr

rs_restore_intr

rs_setup_intr

DESCRIPTION

DOS

Installs user function called on Interrupt condition

USAGE

```
#include <uadi32.h>

int rs_setup_intr (userfunc)

void (far *userfunc)();    the user-supplied intr service routine
```

REMARKS

This function installs a user-supplied function to be activated when the RS board sends an interrupt. The user function can be any function (no parameters). rs_install_intr saves first the current vector hooked to the seiected IRQ, then hooks the API interrupt handler to it. The API interrupt handler essantially calls the user-supplied interrupt service routines, in addition to performing some actions that are reiated to the interrupt mechanism.

RETURN VALUE

RS_SUCCESS

SEE ALSO

rs_restore_intr

rs_set_irq

rs_sys_id

DESCRIPTION

Get system ID and version number

USAGE

#include <uadi32.h>

int rs_sysjd (major, minor);

int * major, major version number

int * minor minor version number

REMARKS

This routine reads the system ID string from the GSR section of the common memory. If it recognizes the RS-429 signature, it reads the version number and loads it into the supplied integer pointers major and minor.

RETURN VALUE

RS SUCCESS Operation successful

RS_NO_BOARD No board signature found

EXAMPLE

if (rs_set_board(OxD000)!=RS_SUCCESS)

printf("Board not present /n");
eise{
rs_sys_id(&min, &maj);
printf ("RS-429 Version %d.%d at D000:0000",min,maj); }

SEE ALSO

rs_set_board

rs232_tx_complete

DESCRIPTION

Check if entire buffer was transmitted

USAGE

#include <uadi32.h>

int rs232_tx_complete (channel)

int channel; transmit channel number

REMARKS

This function checks the appropriate bit in the board's Status Register. This bit is set by the RS board in both TX_NORMAL and TX_LOOP modes when the entire data in the buffer specified in the Transmit Command was sent out to the RS. After reading the bit, the function clears it for the next operation. This function should be used in a polled (non-interrupt) fassion to check if the buffer was transmitted. In interrupt mode it should be called from the Interrupt Service Routine to identify the source of the interrupt.

RETURN VALUES

YES Transmit fuffer is complete

NO Transmit buffer not complete

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

/* polled transmit operation: transmit 3 RS words on channel 1 */

```
RS232 tx_buff[3] = {Oxl 1111111, 0x22222222, 0x33333333 };
/*transfer data into the board's memory starting at offset Ox0000: */
rs232_writebuf(Ox0000,tx_buff, 3*4);
rs232_tx_cmd(1,TX_NORMAL,Ox0000,3,O); /* start transmission */
while (rs232_tx_complete(1 )== N0); /* wait for transmit complete */
printf ("3 words transmitted.\n"); I complete */
```

SEE ALSO

rs232_tx_count

rs232_tx_count

DESCRIPTION

Read Transmit Counter value

USAGE

#include <uadi32.h>

int rs232_tx_count (channel)

int channel; the transmit channel number (1-2)

REMARKS

This function returns the number of transmitted data words modulo the buffer size since the last transmit command, i.e. reiative to the beginning of the buffer. For example, if the buffer size specified in an TX_LOOP command is 100 and 230 words were transmitted since the command was accepted, this function will return 30.

RETURN VALUE

Transmit data counter value

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

rs_txcmd(channel,TX_NORNAL,Ox0000, 100,0);

```
while (!rs232_tx_completeO)
printf("\r %d words transmitted", rs232_tx_count(channel));
/* tx complete, display final transmit count (should be 100)*/
printf("\rTransmit complete, total %d words transmitted";
        rs232_tx_count(channel));
```

SEE ALSO

rs232_rx_count **rs232_tx_complete**

rs232_clr_tx_count

DESCRIPTION

Reset Transmit Counter value

USAGE

#include <uadi32.h>

int rs232_tx_count (channel)

int channel; the transmit channel number (1-2)

REMARKS

This function clears transmit counter register.

RETURN VALUE

Transmit data counter value

RS_INV_CHNL Invalid or unconfigured channel

SEE ALSO

rs232_rx_count **rs232_tx_complete**

rs232_tx_cmd

DESCRIPTION

Send Transmit Command to an RS board

USAGE

#include <uadi32.h>

int rs232_tx_cmd (channel, command, start, count, delay);

int channel; The transmit channel number (1-2)

WORD command;
Command code: RS_TX_NORMAL,RS_TX_LOOP or RS_TX_STOP
WORD start,; Location (offset from board base addresss) of the first transmit data word.
WORD count; Data Count: number of RS data
WORD delay, Pre-transmission delay in milioseconds

REMARKS

This routine sets-up a Transmit Command in the board's Transmit Control Block and waits for the board's response. Before calling this function the data to be transmitted (not applicable for STOP command) should be placed in RS's memory, normally by using the rs232_writebuf function, at the address offset start. If command is equal to RS_ TX_NORMAL, RS will initiate a time delay of delay microseconds after which it will transmit count words starting at address start. If command is equal to RS_TX_LOOP, RS will perfom the same operations as in NORMAL mode but after the entire buffer is transmitted, it will re-initiate the entire sequence (delay+transmission) and will keep cycling until it receives an RS_ TX_STOP command). If command is equal to RS_TX_STOP, all the other parameters are ignored and transmission (if any) is stopped on the channel.

RETURN VALUE

RS_SUCCESS command was accepted: execution started.

RS_INV_CHANNEL invalid or unconfigured channel specified

RS_INV_ADDRESS start address does not fall in the 0-F00 range

RS_INV_SIZE count is such that the last data word would fall outside of the 0400 range (start + count> F00).

RS_CHN L_BUSY command=RS_TX_NORMAL or RS_ TX_LOOP and the channel or its twin channel is already transmitting.

RS_BOARD_RESP board did not respond to the command within the response interval defined by _RS_RESP_ TIMEOUT

EXAMPLE

See TXCMD.C example program

SEE ALSO

rs232_tx_complete

rs232_tx_count

rs232_ei_txcomplete

rs232_val_chan

DESCRIPTION

Validate Transmitter channel number

USAGE

```
#include <uadi32.h>

int rs232_val_chan (channel);

int channel; channel number – 1or 2
```

REMARKS

This routine checks if the specified number represents a valid configured Transmitter Channel on the currently seiected board by reading the boards Channel Configuration Register.

RETURN VALUE

YES Valid channel

NO Not valid channel

RS_INV_CHNL Invalid or unconfigured channel

EXAMPLE

```
/* count # of valid channels on the board */

for (chan=1; chan<=2; chan++) {
if (rs232_val_chan(chan)) {
)
}
printf ("%d Transmitter Channels found\n",n);
```

SEE ALSO

rs_val_rx_chan

rs232_writebuf

DESCRIPTION

Write a section of ARTIC memory

USAGE

#include <uadi32.h>

int rs232_writebuf (offset, src, byte_count) WORD offset

VOID *src;

WORD byte_count;

REMARKS

This function is used to transfer a block of data into the RS board Dual-Ported memory. The most typical use is to transfer Transmit Data to the board before issuing a Transmit Command. This function should be used rather than direct memory operations in order to maintain compatibility with drivers for protected-memory operating systems.

RETURN VALUE

RS_SUCCESS Transfer complete

EXAMPLE

SEE rs232_tx_cmd example

SEE ALSO

rs_readbuf

rs232_readbuf

DESCRIPTION

Read a section of ARTIC memory

USAGE

#include <uadi32k.h>

int rs232_readbuf (dst,src,size)

void *dest; destination address

void far *src; address of data to read from

int size; number of bytes to be read starting from src

REMARKS

This function is used to transfer a block of data from the board Dual-Ported memory into program memory. The most typical use is to transfer Receive Data from the board after receiving data. This function should be used rather than direct memory operations in order to maintain compatibility with drivers for protected-memory operating systems.

RETURN VALUE

ARTIC_SUCCESS Transfer complete

EXAMPLE

SEE rs232_rx_cmd example

SEE ALSO

Rs232_writebuf