FUJITSU

Application Note

Using UART0 with an
internal timer to generate the baudrate
(MB90670/75 series)
© Fujitsu Mikroelektronik GmbH

30.1.1997 Vers. 1.0 by M.Mierse

In some applications one will have to use the internal timer #0 to generate the baudrate for UART0, because the dedicated baudrate generator cannot be used (e.g. because of an odd frequency of the oscillator). This note shows how to it.

Using the dedicated baudrate generator

UART0 is a serial I/O port for communications with external devices. The easiest way to use the UART is to set a baudrate using the dedicated baudrate generator. The baudrate is calculated using the follwing equation :

$$b = \frac{\Phi}{8xnR} \qquad (1)$$

with : b : baudrate

Φ : machine clock (after PLL) in Hz
x : clock divider (3,4 or 5) bits BCH,BCH0
n : factor (1,2,4 or 8) bits RC3..RC2
R : factor (12 or 13) bit RC0

where the specified bits are included in the URD(rate and data-) register :

BCH	RC3	RC2	RC1	RC0	BCH0	P	D8

Table 1 : The URD-Register bit meaning

Please note, that the values in the manual on page 91 are NOT correct ! The right values for calculating the dedicated baudrate for an asyncronous communication are :

RC3	RC2	RC1	*n*		RC0	*R*		BCH	BCH0	*x*
0	0	0	*1*		0	*12*		0	1	*4*
0	1	0	*2*		1	*13*		1	0	*3*
0	1	1	*4*					1	1	*5*
1	0	0	*8*							

Table 2 : bit settings for calculating decidated baudrates

Example : for a 16Mhz clock (e.g. 4Mhz clock and PLL=4x) the appropriate values for 9600 baud would be x=4, n=4 and R=13 $\rightarrow$ URD=3C$_{\text{hex}}$

Using timer0 to generate the baudrate

If using the internal timer (recource-name : 16-bit reload timer #0), the RC3...RC0-bits in the control register have to be set to “1101”. Then the reload value is calculated from equation 2 :

$$r = \frac{\Phi}{16xb} - 1 \qquad (2)$$

with : r : reload value (Register TMR0)

Φ : machine clock (after PLL) in Hz
x : clock divider(2^1,2^3,2^5)bits CSL0,1 in TMCSR
b : baudrate

For the previous example (16Mhz clock rate, b=9600 baud) the appropriate value for r would be 51 (x=2 assumed). To set up the UART0 for use with the internal timer, the timer0 control register TMCSR0 should be initialized at least with the reload-mode-bit RELD and the enable bit CNTE set to 1. Interrupts can be selected, but are not useful because the speed of transmission would be lower depending on the interrupt service routine. Setting bit 0 of the TMCSR0 triggers the timer.

The clock signal output pin (TOT0) from timer 0 is internally connected to the UARTs, so external connections are not required.

UART1 works in a similar way, but the reload values for UART1 are calculated from the equation :

$$r = \frac{\Phi}{32xb} - 1 \qquad (3)$$

<u>example code :</u>

```
.
.
.
void main(void)
{
                              /* both function calls of initUART would */
                              /* initialize the UART0 with 9600,8,N,2  */
                              /* async. communication. 16Mhz clock assumed */

/*  initUART(0x059,0 ,0x03C);     /* init UART0 for Baudgenerator-mode */

  initUART(0x059,0 ,0x06C);     /* init UART0 for use with timer0 */


  while(1)                     /* main loop */
  {
    putS("Test \13\10");       /* put string from UART0 */
    wait(10000);               /* short delay */
  }
}

.
.
.
void wait(int i)
{
  for (; i ; i--);         /* very simple delay loop */
}

void initUART (BYTE modus, BYTE interr, BYTE baud)
{
      UMC_SOE = 0;             /* stop UART, clear flags */
      URD = baud;               /* set baud rate */
      USR = interr;             /* enable/disable UART interrupts */
      UMC = modus;              /* set various bits (start,stop,parity etc) */

      if (URD && 0x068)             /* If Timer0 should be used for Baudrate :
*/
      {
      TMCSR0 = 0x012;          /* Set as Reload timer; 0.125us speed(X=2) */
      TMCSR0_INTE = 0;          /* No INTs for Baud Generator Mode */
      TMR0 = 51;               /* 51 for 9600 Baud (with X=2 and 16MHz clock)
*/
      TMCSR0_TRG = 1;          /* Trigger for the timer0 */
      }
}

.
.
.

int  Putch (int ch)            /* sends a single character */
{
      while (!USR_TDRE);       /* wait for empty buffer */
    UIDR = (BYTE)ch;        /* fill Input/Output register */
    return ch;
}

void putS(char *buf)    /* puts a string */
{
    while (*buf != '\0') Putch(*buf++);      /* send every char */
}
```