Mikroprozessorpraktikum WS 12/13

Carlos Martín Nieto, Simon Hohberg, Tu Tran

February 13, 2013

7 USART

7.1 UART Senden

7.1.1 Folgende Vorgaben für die Initialisierung der UART zur Lösung der Aufgabe:

- **UART1 an den Portleitungen P3.6 (TxD) und P3.7 (RxD) wird genutzt**
- **Controller arbeitet mit 7,3728MHz DCOCLK-Takt**
- **SMCLK-Taktquelle für Baudratengenerator**
- **Datenübertragungsrate 115200 Bit/sek**
- **Datenformat ist 8,N,1**
- **Initialisierung der UART erfolgt mit der init_UART1() aus init.c**

Die Funktion init_UART1() aus init.c ist entsprechend den Vorgaben zu vervollständigen. Testen Sie die Richtigkeit der Initialisierung, indem Sie ein Zeichen in das U1TXBUF-Register schreiben.

Siehe 7.1.2!

7.1.2 Entwickeln Sie ein kleines Programm, welches aus Tastendruck P1.0 das Zeichen "?" über die UART zum PC schickt. Auf dem PC muß zum Empfang des Zeichens ein Terminalprogramm mit den entsprechenden Parametern gestartet werden.

```
1  // init.c
2  void init_UART1(void)
3    {
4      //  P3SEL .....;              // USART RX und TX dem Modul zuweisen
5      P3SEL = BIT6 | BIT7;
6      //  U1CTL .....;              // Reset
7      U1CTL = SWRST;
8      //  U1CTL .....;              // Format 8N1
9      U1CTL |= CHAR; // 8 Bits Daten, keine Paritaet und 1-Bit Stop ist default
10     //  U1TCTL.....;              // Taktquelle SMCLK
11     U1TCTL |= SSEL0 + SSEL1; // Taktquelle SMCLK
12     //  U1BR0 .....;              // Teiler Low-Teil, da 7372800/64 ca. 115200
13     U1BR0 = 64; //7372800/64 ca. 115200
14     U1BR1 = 0;
15     //  U1BR1 .....;              // Teiler High-Teil
16     //  U1MCTL ....;              // Modulationskontrolle
17     U1MCTL = 0;
18     //  ME2 .......;              // Enable USART1 TXD/RXD
19     ME2 = BIT5 + BIT4;
20     //  U1CTL .....;              // Reset
21     U1CTL &= ~SWRST;
```

```
22      //  IE2 .......;              // Enable Interrupt
23      //IE2 = BIT5 + BIT4;
24      }
25
26    //main.c
27    BIT_CLR(P1DIR, TASTE_LINKS);
28    BIT_SET(P1IES, TASTE_LINKS); // HL an linker taste loest interrupt aus
29    BIT_SET(P1IE, TASTE_LINKS); // interrupts fuer linke taste erlauben
30    BIT_CLR(P1IFG, TASTE_LINKS); // interrupt flag loeschen
31  _bis_SR_register(GIE);
32  //===Hier die Endlosschleife quasi das Betriebssystem==================
33    while(1){
34      }    // Ende der Endlosschleife
35  }        // Ende Main
36  //====Ende des Hauptprogramms ========================================
37
38  #pragma vector = PORT1_VECTOR
39  __interrupt void PORT1(void)
40  {
41      if (P1IFG & TASTE_LINKS) {
42          BIT_CLR(P1IFG, TASTE_LINKS);
43          U1TXBUF = '?';
44      }
45  }
```

7.1.3 Schicken Sie bei jedem Tastendruck eine Zeichenkette "Hallo World" zum PC.

Initialisierung wie oben.

```
1   void print_buf(const char *str)
2   {
3     do {
4           U1TXBUF = *str;
5           /*
6            * (115200 / 8)**-1 = 70us pro Zeichen
7            */
8           wait(7);
9       } while(*str++);
10  }
11
12  #pragma vector = PORT1_VECTOR
13  __interrupt void PORT1(void)
14  {
15      if (P1IFG & TASTE_LINKS) {
16          BIT_CLR(P1IFG, TASTE_LINKS);
17          print_buf("Hallo World\r\n");
18      }
19  }
```

7.2 UART Empfang

7.2.1 Initialisieren Sie die USART1 so, daß ein empfangenes Zeichen einen Interrupt auslöst. In der ISR der USART1 soll folgendes realisiert werden:

- **wird das Zeichen E empfangen, schaltet LED P4.1 ein**
- **wird das Zeichen A empfangen, schaltet LED P4.1 aus**
- **wird das /CR Zeichen empfangen, toggelt LED P4.2**

In der interrupts.c sollte die entsprechende ISR für die USART1RX eingefügt werden.

Initialisierung wie oben.

```
#pragma vector = USART1RX_VECTOR
__interrupt void USART1RX(void)
{
    char c = U1RXBUF;
    BIT_CLR(IFG2, URXIFG1);

    if (c == 'E')
        LED_ON(LED_GELB);
    else if (c == 'A')
        LED_OFF(LED_GELB);
    else if (c == '\r')
        LED_TOGGLE(LED_GRUEN);
}

...
void init_UART1(void)
{
    ...
    //  IE2 .......;              // Enable Interrupt
    IE2 = BIT4;
}
```

7.2.2 Über ein Terminalprogramm am PC soll eine Zeichenkette eingegeben und mit CR+LF abgeschlossen werden. Das Terminalprogramm überträgt die Zeichenkette zum Controller. Dieser soll die Zeichenkette (Zeichen für Zeichen) im Interrupt empfangen, die Länge der Zeichenkette bestimmen und zum PC zurückschicken. Die empfangene Zeichenkette soll auf dem LCD-Display angezeigt werden.

Initialisierung wie oben.

```
void print_buf(const char *str)
{
    do {
        U1TXBUF = *str;
        /*
        * (115200 / 8)**-1 = 70us pro Zeichen
        */
        wait(7);
    } while(*str++);
}

static int len = 0;

#pragma vector = USART1RX_VECTOR
__interrupt void USART1RX(void)
{
    char c = U1RXBUF;
```

```
18      char str[14] = {0};
19
20      if (c == '\r') {
21          snprintf(str, sizeof(str)-1, "%d\r\n", len);
22          print_buf(str);
23          lcd_clear(WHITE);
24          lcd_paint();
25          len = 0;
26      } else {
27          str[0] = c;
28          str[1] = '\0';
29          lcd_string(BLACK, (len % 21)*6, (len / 21)*9, str);
30          lcd_paint();
31          len++;
32
33      }
34  }
```

7.3 Sensordaten

7.3.1 Entwickeln Sie ein Programm das zyklisch die Uhrzeit und die Werte des SHT11- Sensors als ASCII-Zeichenkette auf der seriellen Schnittstelle ausgibt und in geeigneter Form parallel dazu auf dem LCD-Display darstellt.

Initialisierung wie oben.

```
1   static unsigned char hours = 0;
2   static unsigned char mins = 0;
3   static unsigned char secs = 0;
4
5   void print_buf(const char *str)
6   {
7       do {
8           U1TXBUF = *str;
9           /*
10          * (115200 / 8)**-1 = 70us pro Zeichen
11          */
12          wait(7);
13      } while(*str++);
14  }
15
16  #pragma vector = TIMERB0_VECTOR
17  __interrupt void TIMERB0(void)
18  {
19    char buffer[16];
20      secs++;
21      // Sekundenueberlauf
22      if (secs > 59) {
23          mins++;
24          secs %= 60;
25          // Minutenueberlauf
26          if (mins > 59) {
27              mins %= 60;
28              hours++;
29              // Stundenueberlauf
30              if (hours > 23) {
31                  hours %= 24;
32              }
33          }
34      }
```

```
35      SHT11_Read_Sensor();
36      print_buf(temp_char);
37      print_buf(humi_char);
38      sprintf(buffer, "  %02d:%02d:%02d\r\n", hours, mins, secs);
39      print_buf(buffer);
40  }
```

- Ausgabe:

```
1  T= 25.72C,F=32.7%  15:39:51
2  T= 25.72C,F=32.5%  15:39:52
3  T= 25.72C,F=31.7%  15:39:53
4  T= 25.74C,F=31.2%  15:39:54
5  T= 25.83C,F=31.8%  15:39:55
6  T= 26.12C,F=37.0%  15:39:56
7  T= 26.35C,F=45.0%  15:39:57
8  T= 26.84C,F=53.1%  15:39:58
9  T= 26.94C,F=60.2%  15:39:59
10 T= 26.83C,F=66.0%  15:40:00
11 T= 26.69C,F=69.4%  15:40:01
12 T= 26.62C,F=71.1%  15:40:02
13 T= 26.63C,F=72.2%  15:40:03
14 T= 26.60C,F=73.1%  15:40:04
15 T= 26.53C,F=73.3%  15:40:05
16 T= 26.45C,F=73.7%  15:40:06
17 T= 26.36C,F=72.3%  15:40:07
18 T= 26.29C,F=68.3%  15:40:08
19 T= 26.25C,F=64.2%  15:40:09
20 T= 26.22C,F=60.4%  15:40:10
21 T= 26.20C,F=57.7%  15:40:11
```