Microcontroller
Selbststudium Semesterwoche 7

1. Timersystem ohne Interrupt in C
Timersystem für 1sec-Timer realisieren: Aus dem Hauptprogramm (main.c) soll 50 mal die Funktion delay20ms in der Datei delay.c/delay.h aufgerufen welche eine 20ms Verzögerung bewirkt. Dadurch ergibt sich insgesamt eine Verzögerung von 1sec (= 50 x 20ms). Erzeugen Sie dabei die Zeitverzögerung in delay.c mittels des Timersystems, indem Sie dort das Flag eines Timer Output Compare Ereignisses abfragen (Polling). Verwenden Sie TPM1CH0.

```
//main.c
#include <hidef.h>      // for EnableInterrupts macro
#include "derivative.h" // include peripheral declarations
#include "delay.h"        // include for delay function

void initClock(void)
{
    SOPT = 0x73;            // Watchdog=disabled, Stop-Mode=enabled
    ICGC1 = 0x28;           // ICG-Mode: FEI
    ICGC2 = 0x72;           // N=10, R04
    while(!ICGS1_LOCK);     // Warten bis Clock stabil (locked) ist.
}

void initPorts(void)
{
    PTFDD = 0xff;           // Port F = Output
    PTADD = 0x00;           // Port A = Input
    PTAPE = 0xff;           // Port A = Pull-Up enable
}

void initTimer(void)
{
      // select bus clock
      TPM1SC_CLKSB = 0;
      TPM1SC_CLKSA = 1;

      // now, that we're not able to count to 100'000 with a 16bit value
      // we set the prescalar to 2 (5MHz/2 = 2'500'000cycles/sec)
      // alternative declaration could be: TPM1SC_PS = 1;
      TPM1SC_PS0 = 1;
      TPM1SC_PS1 = 0;
      TPM1SC_PS2 = 0;

      // reset to start timer
      TPM1CNT = 0;
}


void initChannel(void)
{
      // set channel as output compare channel
      TPM1C0SC_MS0B = 0;
      TPM1C0SC_MS0A = 1;

      // set port PTD0
      // this port can be used to switch an output device (e.g. LED)
      TPM1C0SC_ELS0B = 0;
      TPM1C0SC_ELS0A = 1;

      // 2'500'000cycles/s * 0.02s = 50'000cycles
      // we reach 2 milliseconds in ~50'000cycles (hex: 0C350)
      TPM1C0V = 0xC350;
}
```

```
void main(void)
{
       int i = 0;

       initClock();            // Clock und Watchdog konfigurieren

       initPorts();            // Ports konfigurieren

       EnableInterrupts;       // Interrupts aktivieren

       initTimer();
       initChannel();


    for(;;)
    {
       // 50 * 20ms = 1sec
       for(i = 0; i<50; i++)
       {
              delay20ms();
       }

       //stepping the LEDs
       PTFD++;
    }
}
```

//delay.c

```
#include "derivative.h"

void delay20ms(void)
{
       while(TPM1C0SC_CH0F != 1)
       {
       }

       TPM1C0SC_CH0F = 0;
       return;
}
```

//delay.h

```
void delay20ms(void);
```

2. Timersystem mit Interrupt in C

Wie unter Aufgabe 1, aber diesmal erzeugen Sie die Zeitverzögerung mittels einem Timer Interrupt (Timer Output Compare, TPM1Ch0 an Port D:0) so, dass Zeit nicht mit 50 mal 20ms, sondern direkt mit einem 1s Interrupt realisiert wird. Ihr Unterprogramm heisst jetzt delayISR.

```
//main.c
#include <hidef.h>      // for EnableInterrupts macro
#include "derivative.h" // include peripheral declarations

void initClock(void)
{
    SOPT = 0x73;            // Watchdog=disabled, Stop-Mode=enabled

    ICGC1 = 0x28;           // ICG-Mode: FEI
    ICGC2 = 0x72;           // N=10, R04
    while(!ICGS1_LOCK);     // Warten bis Clock stabil (locked) ist.
}

void initPorts(void)
{
    PTCDD = 0xff;
    PTDDD = 0xff;
}

void initTimer(void)
{
       // select bus clock
       TPM2SC_CLKSB = 0;
       TPM2SC_CLKSA = 1;

       // no PWM
       TPM2SC_CPWMS = 0;

       // set prescalar to 2 (5MHz/128 = 39062.5 cycles/sec)
       TPM2SC_PS0 = 1;
       TPM2SC_PS1 = 1;
       TPM2SC_PS2 = 1;

       // 0 - 2^16 // free running counter
       TPM2MOD = 0;

       // reset to start timer
       TPM2CNT = 0;
}

void initChannel(void)
{
       // set channel as output compare channel
       TPM2C0SC_MS0B = 0;
       TPM2C0SC_MS0A = 1;

       // set port D
       // toggle output on compare
       TPM2C0SC_ELS0B = 0;
       TPM2C0SC_ELS0A = 1;

       // 39062.5cycles/s
       // take the rounded value and subtract 1
       // because the counter is zero-indexed
       TPM2C0V = 0x9895;

       // enable interrupt
       TPM2C0SC_CH0IE = 1;

       // reset compare flag of channel 0
       TPM2C0SC_CH0F = 0;
}
```

```
// this function (ISR=Interrupt Service Routine)
// is called in case a interrupt is raised
interrupt void delayISR(void)
{
        // increment port C
        PTCD++;

        // set the offset value for the next count cycle
        // we use this method instead of resetting the whole timer
        TPM2C0V = TPM2C0V + 0x9895;

        // acknowledge interrupt
        // if you don't do this, your interrupt keeps hanging in an endless loop
        // (remember: an interrupt is raised if CH0F && CH0IE == 1, thus we set CH0F = 0)
        TPM2C0SC_CH0F = 0;
}

/**
 * main program
 */
void main(void)
{
    initClock();            // Clock und Watchdog konfigurieren

    initPorts();            // Ports konfigurieren

    EnableInterrupts;       // Interrupts aktivieren

       initTimer();
       initChannel();

    for(;;)
    {
       // don't do anything here
    }
}
```

//project.prm

```
VECTOR ADDRESS 0xFFE2 errISR_TPM2O     // TPM2 overflow
VECTOR ADDRESS 0xFFE4 errISR_TPM2CH4   // TPM2 channel 4
VECTOR ADDRESS 0xFFE6 errISR_TPM2CH3   // TPM2 channel 3
VECTOR ADDRESS 0xFFE8 errISR_TPM2CH2   // TPM2 channel 2
VECTOR ADDRESS 0xFFEA errISR_TPM2CH1   // TPM2 channel 1
VECTOR ADDRESS 0xFFEC delayISR         // TPM2 channel 0
VECTOR ADDRESS 0xFFEE errISR_TPM1O     // TPM1 overflow
VECTOR ADDRESS 0xFFF0 errISR_TPM1CH2   // TPM1 channel 2
VECTOR ADDRESS 0xFFF2 errISR_TPM1CH1   // TPM1 channel 1
VECTOR ADDRESS 0xFFF4 errISR_TPM1CH0   // TPM1 channel 0
```