Mikroprozessorsysteme Praktikum 3

```
#include "../h/pmc.h"
#include "../h/tc.h"
#include "../h/pio.h"
#include "../h/aic.h"

void taste_irq_handler (void) __attribute__ ((interrupt));

// Interruptserviceroutine für die Tasten SW1 und SW2
void taste_irq_handler (void)
{
  StructPIO* piobaseB   = PIOB_BASE;        // Basisadresse PIO B
  StructAIC* aicbase  = AIC_BASE;       //__

// ab hier entsprechend der Aufgabestellung ergänzen
// ************************************************
   StructPIO* piobaseA   = PIOA_BASE;
   if(!(piobaseB->PIO_PDSR & KEY1))
 {
       piobaseA->PIO_PDR = (1 << PIOTIOA3);
       piobaseB->PIO_CODR = ALL_LEDS;
 }

 if(!(piobaseB->PIO_PDSR & KEY2))
 {
       piobaseB->PIO_SODR = ALL_LEDS;
       piobaseA->PIO_PER = (1<<PIOTIOA3);
 }

    aicbase->AIC_EOICR = piobaseB->PIO_ISR; //__
}

// Timer3 initialisieren
void Timer3_init( void )
{
  StructTC* timerbase3  = TCB3_BASE;    // Basisadressse TC Block 1
  StructPIO* piobaseA   = PIOA_BASE;    // Basisadresse PIO B

    timerbase3->TC_CCR = TC_CLKDIS; // Disable Clock

  // Initialize the mode of the timer 3
  timerbase3->TC_CMR =
    TC_ACPC_CLEAR_OUTPUT  |    //ACPC    : Register C clear TIOA    bei 8000 (Periodendauer) clear
    TC_ACPA_SET_OUTPUT    |    //ACPA    : Register A set TIOA      von 0 --> 400 A (Hälfte der Periodendauer) setzt
    TC_WAVE               |    //WAVE    : Waveform mode
    TC_CPCTRG             |    //CPCTRG  : Register C compare trigger enable    wieder auf 0 setzen
    TC_CLKS_MCK8;          //TCCLKS  : MCKI / 1024  --> MCKI 8 = 31250 für 20 ms

  // Initialize the counter:
  timerbase3->TC_RA = 31250;    //__
  timerbase3->TC_RC = 62500;    //__

  // Start the timer :
  timerbase3->TC_CCR = TC_CLKEN ;   //__
  timerbase3->TC_CCR = TC_SWTRG ;   //__
  piobaseA->PIO_PER  = (1<<PIOTIOA3) ;  //PER Port Enable Register  CPU hat Kontrolle
  piobaseA->PIO_OER  = (1<<PIOTIOA3) ;  //OER Output Enable Register    Lege Timer als Output fest, während er deaktiviert ist
  piobaseA->PIO_CODR = (1<<PIOTIOA3) ;  //CODR Clear Output Data Register   Reset den aktuellen Output
}
```

```
int main(void)
{

    StructPMC* pmcbase  = PMC_BASE; // Basisadresse des PMC (Power Management Controller)
    StructPIO* piobaseA     = PIOA_BASE;    // Basisadresse PIO A
    StructPIO* piobaseB     = PIOB_BASE;    // Basisadresse PIO B

    pmcbase->PMC_PCER   = 0x6200;   // Peripheral Clocks einschalten für PIOB, (PCER = Periphal Clock Enable Register)

// ab hier entsprechend der Aufgabestellung ergänzen
//**************************************************

    StructAIC* aicbase = AIC_BASE;      //AIC = Advanced Interrupt Controller
    aicbase->AIC_IDCR = 1 << 14;        //IDCR (Interrupt Disable Command Register) Interrupt PIOB deaktivieren
    aicbase->AIC_ICCR = 1 << 14;        //ICCR (Interrupt Clear Command Register) Interrupt PIOB löschen

    aicbase->AIC_SVR[PIOB_ID] = (unsigned int) taste_irq_handler;   //SVR = Source Vector Register --> Interrupt für PIOB registrieren
    aicbase->AIC_SMR[PIOB_ID] = 0x1;    //SMR (Source Mode Register) --> Priorität festlegen
    aicbase->AIC_IECR = 1 << 14;        //IECR (Interrupt Enable Command Register) --> Interrupt PIOB einschalten

    piobaseB->PIO_IER = KEY1 | KEY2;    //IER (Interrupt Enable Command Register) --> Aktiviere Interrupt für KEY1 und KEY2

    Timer3_init();                       //Timer initialisieren

   // piobaseA->PIO_PDR = (1<<PIOTIOA3);  //Gebe der Peripherie die Kontrolle

    piobaseB->PIO_PER = ALL_LEDS;
    piobaseB->PIO_OER = ALL_LEDS;
    while(1)
    {
    }
    return 0;
}
```

Aufgabe 1

Könnte der eingesetzte Timer bei dem benötigten symmetrischen Signal auch anders betrieben werden?

Timer können auch zur Erzeugung von analogen Signalen benutzt werden.

Aufgabe 2

Welche Möglichkeiten haben Sie gefunden, um das Pumpensignal ein- bzw. auszuschalten?

Wir haben uns entschieden mit der PIO Disable Register den PIO in der Interrupt Service Routine auszuschalten bzw. Anzuschalten.

Für welche Lösung entscheiden Sie sich und warum?

Wir halten diese Lösung am besten, da sie die komplette Peripherie abkoppelt bzw. abschaltet und da weniger Peripherie für den Mikroprozessor immer(aus Leistungs- und Stromspargründen) besser ist.