

C-Programmierung für eingebettete  
Systeme Pointer, Peripherie, PIO, Interrupt,  
Timer (WAVE-Mode)



MIKROPROZESSORPRAKTIKUM

WS2013

**Termin 3**

C-Programmierung für eingebettete Systeme Pointer, Peripherie, PIO,  
Interrupt, Timer (WAVE-Mode)

Name, Vorname	Matrikelnummer	Anmerkungen
Datum	Raster (z.B. Mi3x)	Testat/Datum

Legende: V:Vorbereitung, D: Durchführung, P: Protokoll/Dokumentation, T: Testat

## Termin 3

### Lernziele:

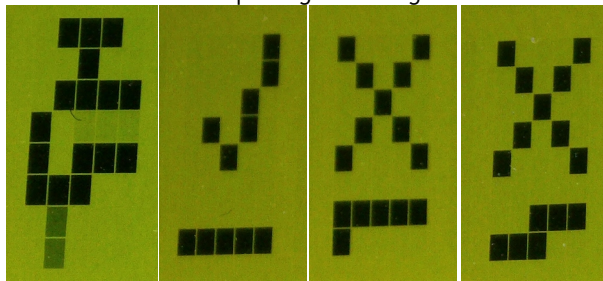
Mit den folgenden Versuchen sollen Sie lernen, wie Sie aus der Sprache "C" Peripherie (z. B. Timer) von modernen Mikrocontrollern nutzen.

### Arbeitsverzeichnis:

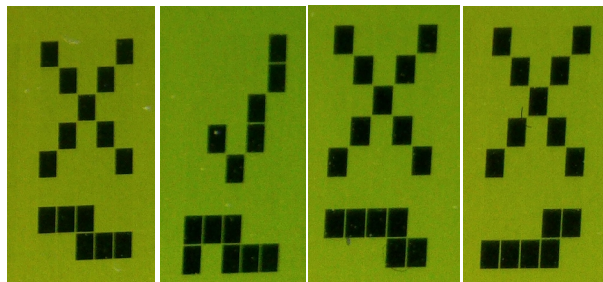
Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsWS2013. Dort finden Sie zu jedem Termin vorgegebene Dateien.

### Weitere Infos:

Ab dem Sommersemester 2011 stehen an jedem Laborarbeitsplatz WaSim (Waagensimulatoren) zur Verfügung. Mit diesen angeschlossenen WaSim kann auch das Pumpensignal überprüft werden. Im Display werden mit folgenden Symbolen die verschiedenen Pumpensignale dargestellt:



*es pumpt*   *kein Signal*   *dauer high*   *Frequenz zu hoch*  
*Gewicht nimmt zu*



*Frequenz zu niedrig*   *Frequenz richtig*   *Highpegel zu lang*   *Lowpegel zu lang*



### Aufgabe 1:

Es soll eine Kolbenhubpumpe, welche über PA1/TIOA3 angesteuert wird, betrieben werden. Die Pumpe benötigt ein symmetrisches Rechtecksignal mit einer Frequenz von ca. 50Hz. Sie könnten eine Zeitschleife programmieren. Hiermit würden Sie aber den Prozessor blockieren (Erinnerung an Termin 2). Besser ist es, Sie initialisieren einen Timer (Timer3) so, dass dieser selbstständig das Signal für die Pumpe erzeugt.

**ACHTUNG: Die Pumpe darf kein Dauerhighsignal erhalten.**

Vervollständigen Sie die das gegebene Programm *Termin3Aufgabe1.c* entsprechend. Ergänzen Sie auch die noch fehlenden Kommentare.

## Aufgabe 2:

Entwickeln Sie eine Funktion, mit der Sie die Tasten SW1 – SW3 der auf dem Board befindlichen Tasten abfragen können. Mit der Taste SW2 soll die Pumpe eingeschaltet und mit der Taste SW1 soll die Pumpe abgeschaltet werden können.

## Aufgabe 3:

Isolieren Sie die Routinen, welche für den Termin6, noch benötigt werden.

## Aufgabe 4:

Erstellen Sie zu diesem Termin ein Protokoll mit den Lösungen zu den Aufgaben und Ihren Erkenntnissen. Das Protokoll sollen Sie zum nächsten Termin vorlegen können.

## Für Fortgeschrittene:

Entwickeln Sie eine Funktion, mit der Sie die Tasten, die an einigen Boards angeschlossenen Tastaturen, abfragen können. Mit der Taste „1“ soll die Pumpe eingeschaltet und mit der Taste „0“ soll die Pumpe abgeschaltet werden können.

ACHTUNG: Die Pumpe darf kein Dauerhighsignal erhalten.

Der Anschluss der Tastatur an das Board ist in der Datei **TastaturAnAT91EB63mitBilder.pdf** dokumentiert.

ACHTUNG: Damit die Tastatur verwendet werden kann, ist es nötig den Jumper E4 auf der Entwicklungsplatine (AT91EB63) auf 2 und 3 zu stecken. Siehe hierzu auch im "AT91EB63 Evaluation Board User Guide" auf Seite 6-5.

Bitte am Ende Ihrer Sitzung Jumper E4 auf der Entwicklungsplatine (AT91EB63) wieder auf 1 und 2 stecken.

```
// Lösung zu Termin 3
// Aufgabe 1
// von:
// vom:
//
#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
// *****

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

// Timer3 initialisieren
void Timer3_init( void )
{
    StructTC* timerbase3 = TCB3_BASE;          // Basisadresse 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
        TC_ACPA_SET_OUTPUT |  //ACPA : Register A set TIOA
        TC_WAVE |              //WAVE : Waveform mode
        TC_CPCTRG |            //CPCTRG : Register C compare trigger enable
        TC_CLKS_MCK1024;       //TCCLKS : MCKI / 1024

// Initialize the counter:
    timerbase3->TC_RA = 300;    // hier sind noch die richtigen Werte zu ermitteln
    timerbase3->TC_RC = 600;    //__

// Start the timer :
    timerbase3->TC_CCR = TC_CLKEN;          //__
    timerbase3->TC_CCR = TC_SWTRG;          //__
    piobaseA->PIO_PER = (1<<PIOTIOA3);    //__
    piobaseA->PIO_OER = (1<<PIOTIOA3);    //__
    piobaseA->PIO_CODR = (1<<PIOTIOA3);    //__
}

int main(void)
{
    StructPMC* pmcbase = PMC_BASE;          // Basisadresse des PMC
    StructPIO* piobaseA = PIOA_BASE;          // Basisadresse PIO A
    StructPIO* piobaseB = PIOB_BASE;          // Basisadresse PIO B

    pmcbase->PMC_PCER = 0x4000; // Peripheral Clocks einschalten für PIOB, ____

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

    while(1)
    {

    }

    return 0;
}
```