MIKROPROZESSORPRAKTIKUM

SS2020

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

Lernziele:

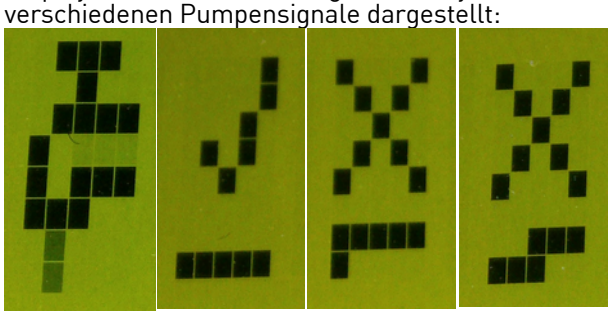
Mit den folgenden Versuchen sollen Sie lernen, wie Sie aus der Sprache "C" Peripherie (z. B. Timer) von modernen Mikrocontrollern nutzen.

Arbeitsverzeichnis:

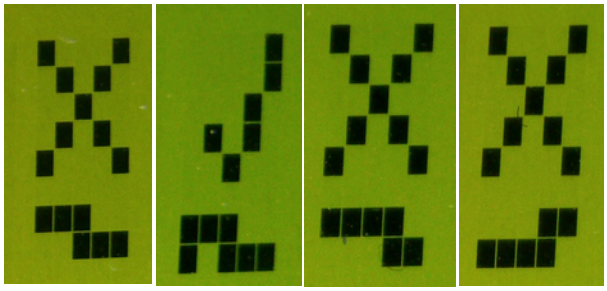
Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsSS2020. Dort finden Sie zu jedem Termin vorgegebene Dateien.

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 Gewicht nimmt zu | *kein Signal* | *dauer high* | *Frequenz zu hoch*



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 und berichtigen Sie auch die Kommentare.

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

Zeigen Sie die Berechnung der benötigten Werte um die Frequenz und das Pulsweitenverhältnis richtig einzustellen.

..

Aufgabe 2:

Erweitern Sie Ihr Programm so, dass die Pumpe durch Betätigung von Tasten eingeschaltet und abgeschaltet werden kann.

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

Für welche Lösung entscheiden Sie sich und warum?

..

Aufgabe 3:

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. Isolieren Sie die Routinen/ Funktionen, welche für die nächsten Termine (siehe speziell Termin6), noch benötigt werden.

Für Fleißige:

Entwickeln Sie eine Funktion, mit der Sie die Tasten, der an einigen Boards angeschlossenen Tastaturen, abfragen können. Mit einer Taste soll die Pumpe eingeschaltet und mit einer anderen Taste 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.
Infos gibt's auch in der Datei „TastaturAnAT91EB63mitBilder.pdf".
Bitte am Ende Ihrer Sitzung Jumper E4 auf der Entwicklungsplatine (AT91EB63) wieder auf 1 und 2 stecken.

```
// Loesung 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 )
{
        StructPMC* pmcbase = PMC_BASE;                   // Basisadresse des PMC
        StructTC* timerbase3  = TCB3_BASE;               // Basisadressse TC Block 1
        StructPIO* piobaseA   = PIOA_BASE;               // Basisadresse PIO B

        pmcbase→PMC_PCER = 0x2200;                       // Peripheral Clocks einschalten für PIOA und TC3

        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 = 400;     // hier sind noch die richtigen Werte zu ermitteln
        timerbase3->TC_RC = 800;     // Die Pumpe soll mit einem 50Hz Signal betrieben werden

 // Starten des Timer und Ausgabe eines Low-Pegel an die Pumpe
        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


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



        return 0;
}
```