h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

MIKROPROZESSORPRAKTIKUM

SS2013

Einführung

Einfuehrung zum Mikroprozessorpraktikum

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

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

Inhaltsverzeichnis

Laborordnung

Sinn dieser Laborordnung ist die Festlegung von Regeln für die Benutzung der Labore in D10

Jeder ordentliche Student des Fachbereich wird in den Grundlagenveranstaltungen auf die gültige Laborordnung hingewiesen.

1. Der Notausschalter des Labors D10/0.32 befindet sich über dem Lichtschalter. Der Notausschalter ist im Notfall zu betätigen, um sämtliche elektrische Geräte stromlos zu schalten. Achten Sie bei der Einführung auf die entsprechenden Hinweise.
2. Im Labor darf maximal Paarweise an den Geräten gearbeitet werden. Das Labor ist für 8 Gruppen ausgelegt.
3. Es ist nicht gestattet sich alleine im Labor aufzuhalten.
4. Die Ersthelfer sind Herr Michael Müller (Raum 0.33 / Tel.: 8430), Herr Manfred Pester (Raum 0.33 / Tel.: 8428) und Herr Sergio Vergata (Raum 0,37 / Tel.; 8491). Wenden Sie sich bitte im Falle einer Verletzung direkt an diese. Das Erste-Hilfe-Material befindet sich im Raum D10/0.41.
5. Es ist nicht gestattet Kabel zu entfernen, Gehäuse zu öffnen und Hardware (außer USB-Sticks) zu installieren oder Änderungen an der Laborinfrastruktur vorzunehmen. Sollte etwas nicht funktionieren, oder es wird etwas benötigt, welches die vorhandene Infrastruktur nicht abdeckt, so wenden Sie sich an den Betreuer des Labors oder direkt an den zuständigen Laboringenieur Manfred Pester (Raum 0.33 / Tel.: 8428).
6. Fahren Sie die von Ihnen benutzten Geräte am Ende Ihres Praktikum/Ihrer Übung herunter und schalten diese aus, es sei denn Sie bekommen vom zuständigen Betreuer andere Anweisungen.
7. Die Einnahme von Speisen und Getränken ist an den Arbeitsplätzen im Labor nicht erlaubt. Offene Speisen und offene Getränke sind im Labor nicht gestattet.
8. Bei der Benutzung des Labordrucker ist Sorgfalt und Sparsamkeit oberstes Gebot.
9. Evtl. ausgestellte Dokumentationen dienen der Laborarbeit und müssen im Raum verbleiben.
10. Die Benutzung von Mobiltelefonen ist untersagt. Schalten Sie vor dem Betreten des Raumes die Geräte ab. In dringenden Fällen können Sie sich über das Labortelefon mit der Nummer 06151 168433 anrufen lassen.
11. Hängen Sie Ihre Kleidung (Mäntel, Jacken, ..) an die dafür vorgesehenen Kleiderständer und nicht über die Stühle.
12. Deponieren Sie Taschen, Laptops u.s.w. nicht in den Gängen, sondern möglichst an den Seiten des Labors oder unter den Tischen.
13. Verlassen Sie Ihren Arbeitsplatz aufgeräumt! Müll gehört in die mehrfach vorhandenen Mülleimer, Altpapier in die dafür vorgesehene blaue Altpapierwanne.
14. Die Fluchtwege sind frei zu halten.

Bei Verstößen gegen die Laborordnung kann die Benutzungsberechtigung versagt werden.

Einleitung

Dieses Dokument soll Ihnen eine kurze Starthilfe in das Mikroprozessorpraktikum geben. Es richtet sich an jene, die noch keinerlei Erfahrung mit hardwarenaher Programmierung haben.

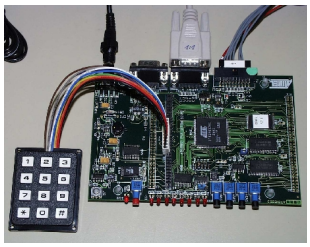
Zuerst werden wir kurz die Zielplattform betrachten, mit der Sie arbeiten werden, sowie die dazu notwendigen Entwicklungswerkzeuge in ihren wichtigsten Grundfunktionen vorstellen.

Bedenken Sie bitte, dass dieses Dokument keinesfalls ausreichend ist, um die Übungsbeispiele zu lösen. Datenblätter sind nun einmal primäre Quellen im Bereich der hardwarenahen Programmierung. Im Mikroprozessorpraktikum kann und soll Ihnen die Beschäftigung damit nicht erspart bleiben. Wir wissen aus eigener Erfahrung, dass es oft mühsam ist, aus den mitunter nicht optimal aufbereiteten einschlägigen Datenblättern die wichtigsten Informationen herauszuholen. Dieses heraus filtern der wesentlichen Informationen in möglichst kurzer Zeit ist aber definitiv eine Fertigkeit, die beim Einstieg ins Berufsleben von Ihnen verlangt werden wird. Sie werden in Ihrer späteren Arbeitsumgebung auch nicht allmonatlich auf Firmenkosten nach Singapur verschifft, um eine persönliche Einführung zum Bauteil des Monats zu bekommen; auch werden Ihre Kollegen mit Ihnen keine Freude haben, wenn Sie die Hardware ständig im Detail erklärt haben wollen.

Andererseits handelt es sich hier um eine Lehrveranstaltung, daher sind Sie natürlich nicht auf sich alleine gestellt. Sie sollen nicht tagelang an ein und demselben Problem sitzen und letztlich frustriert unsere Geräte zum Fenster hinaus werfen. Als erste Anlaufstelle gibt es in den betreuten Übungszeiten Personal, welches Ihnen entsprechende Hinweise geben kann. Bedenken Sie aber bitte, dass es sich bei diesem Personal auch um keine Übermenschen handelt.

Hardware

Um die Programmieraufgaben möglichst vielfältig und abwechslungsreich gestalten zu können, haben wir uns für das Evaluationsboard AT91EB63 der Firma Atmel entschieden. Durch die Möglichkeit Peripherie an das Board anzuschließen, werden Sie in verschiedenen Versuchen, die Funktionalität der einzelnen Komponenten eines modernen Mikrokontroller besser kennenlernen.



Kontrollerplatine (Controllerboard)

Das Herzstück der Übungshardware ist das Evaluationboard AT91EB63. Die relevanten Komponenten darauf sind,

- der Kontroller AT91M63200
- zwei serielle Schnittstellen
- ein Resettaster
- vier 4 Taster
- acht Leuchtdioden
- 256KByte statischer Speicher (RAM)
- 2MByte Flash
- 2MByte serielles DatenFlash
- 64KBytes serielles EEPROM
- 2*32-pin EBI Erweiterungsverbinder
- 2*32-pin MPI Erweiterungsverbinder
- 2*32-pin Ein/Ausgabe Erweiterungsverbinder
- 20-pin JTAG Schnittstellenverbinder

Für weitere Informationen schauen Sie im Benutzer Handbuch „AT91EB63 Evaluation Board“ nach.

Der Mikrokontroller AT91M63200

Der AT91M63200 ist ein Mitglied der Atmel AT91 16/32-Bit Mikrokontrollerfamilie auf Basis des ARM7TDMI Prozessorkern. Der Prozessor hat eine hochperformante 32-Bit RISC-Architektur, unterstützt den hocheffizienten 16-Bit Befehlssatz (THUMB) und hat eine sehr geringe Verlustleistung. Um nicht alle Besonderheiten des Kontroller hier aufzählen zu müssen, verweisen wir an dieser Stelle auf die technische Dokumentation der Firma

ATMEL
AT91
ARM® Thumb®
Microcontrollers
AT91M63200

Rev. 1028A-11/99

Entwicklungsumgebung ARM-Toolchain und Aufgaben

Im Praktikum zur Rechnerarchitektur lernten wir die ARM-Toolchain kennen. Mit dieser werden wir auch in diesem Praktikum arbeiten. Zur Wiederholung und Auffrischung beschäftigen Sie sich mit den Aufgaben von Termin 1 zum Praktikum Mikroprozessorsysteme SS2013. Diese Aufgaben können im ARM-Simulator des GDB (arm-eb63-elf-insight) getestet werden. Die Labore D10/0.32 und D10/0.35 stehen Ihnen während der freien Übungszeiten hierfür zur Verfügung.

Sie möchten sich Ihre eigene Entwicklungsumgebung installieren? Dann schauen Sie z.B. mal hier vorbei: http://www.alphapogo.de/
Oder schauen Sie im Wiki (https://wiki.h-da.de/fbi/technische-systeme/index.php/Hauptseite) des Fachbereich Informatik nach.

Zu jedem Termin für das Praktikum Mikroprozessorsysteme SS2013 gibt es einen Ordner mit Informationen und Beispielen. Sie müssen Ihre Praktikumstermine wahrnehmen. Nur im Labor für Mikroprozessorsysteme D10/0.32 steht Ihnen zur Lösung der Aufgaben Termin 2 bis Termin 6 die notwendige Hardware zur Verfügung.

Termin 1

Lernziele:

Mit den folgenden Versuchen sollen Sie die Sprache "C" einmal aus einer anderen Sicht kennen lernen. An ganz einfachen Programmen sollen Sie ermitteln, welchen Code ein Compiler erzeugt, wo welche Variablen abgelegt werden, wie ein c*all by value / reference* in ARM Assembler umgesetzt wird.

Arbeitsverzeichnis:

Kopieren Sie sich das Verzeichnis /mpsSS2013. Dort stehen im Unterverzeichnis Termin 1 die Dateien *Termin1Aufgabe1.c als* Programmgerüstbeispiel und *makefile* zur Verfügung.

Aufgabe 1:

Es gibt folgende elementare Datentypen. Füllen Sie die Tabellen mit den fehlenden Informationen.

Datentyp	Bit	kleinste Zahl Hex / Dez	größte Zahl Hex / Dez	Bemerkungen
char	8	0x80 / -128	0x7F / 127	
short int	16			
int	(16) 32 (64)			hängt vom verwendeten Compiler und Betriebssystem ab
unsigned int	16			
long int	32 (64)			hängt vom verwendeten Compiler und Betriebssystem ab

Zusätzlich gibt es noch die Gleitpunktzahlendatentypen

Datentyp	Bit	kleinste darstellbare Zahl	größte darstellbare Zahl
float	32		
double	64	ca. 1,7E-308	ca. 1,7E308
long double	80	ca. 1,2E-4932	ca. 1,2E4932

Legen Sie im C Programm Variablen mit den in der Tabelle gezeigten Datentypen an. Weisen Sie den Variablen Werte zu und betrachten Sie sich die Ergebnisse mit dem Debugger im Speicher.

Aufgabe 2:

Geben Sie die folgenden Zahlen in der jeweils anderen Schreibweise des entsprechenden Datentypen an.

Dez	int	float
1	= _____	= _____
-1	= _____	= _____
65535	= 0x0000FFFF	= _____
$1{,}024*10^3$	= _____	= _____
	= _____	= 0x44FFFFFF

Aufgabe 3:

Legen Sie im C Programm zwei lokale integer Variablen an. Weisen Sie diesen im Code Werte zu, z.B. 0x1 und 0x2. Übersetzen Sie Ihr Programm und schauen Sie sich den erzeugten Code an. Führen Sie das Programm im Simulator aus. Ändern Sie im Makefile die Optimierung und beobachten Sie die Auswirkungen.

Aufgabe 4:

Verwenden Sie nun statt der lokalen Variablen globale Variablen. Was ändert sich? Warum?

Aufgabe 5:

Legen Sie nun zwei globale und zwei lokale integer Variablen an. Weisen Sie den Variablen Werte zu. Machen Sie nun Zuweisungen der Form "global=lokal". Was stellen Sie fest? Ändern Sie im Makefile die Optimierung und beobachten Sie die Auswirkungen.

Aufgabe 6:

Vereinbaren Sie einen Funktionsprototypen für eine Funktion "addition", die 3 integer Parameter erwartet. Rufen Sie diese Funktion im Anschluss an die Zuweisung nach Aufgabe 3 mit den beiden lokalen und einer globalen Variablen auf. Dokumentieren Sie Ihre Beobachtungen. Ändern Sie im Makefile die Optimierung und beobachten und dokumentieren Sie die Auswirkungen.

Aufgabe 7:

Ändern Sie den Funktionsprototypen so ab, dass er statt der lokalen Variablen Zeiger (Pointer) auf diese Variablen erwartet und rufen Sie die Funktion entsprechend auf. Was ändert sich und warum? Wo liegen die lokalen Variablen nun?

Aufgabe 8:

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.

Termin 2

Lernziele:

Erster Umgang mit Peripherie in eingebetteten Systemen.

Arbeitsverzeichnis:

Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsSS2013. Dort finden Sie zu jedem Termin vorgegebene Dateien.

Aufgabe 1:

In eingebetteten Systemen und für Kartentreiber muss man oft auf Register zugreifen, die auf festen Adressen liegen. Wir werden uns zuerst mit den Registern der PIO des hier eingesetzten Mikrocontroller (AT91M63200) beschäftigen.

Name	Adresse	Bedeutung
PIOB_PER	0xFFFF0000	PIOB Port Enable Register
PIOB_OER	0xFFFF0010	PIOB Output Enable Register
PIOB_SODR	0xFFFF0030	PIOB Set Output Data Register
PIOB_CODR	0xFFFF0034	PIOB Clear Output Data Register

Legen Sie zunächst Zeigervariablen (PIOB_PER, PIOB_OER, ..) an und initialisieren Sie diese im Code auf die Adresse des Registers. Danach können Sie über diese Zeiger auf die Register zugreifen. Schreiben Sie zunächst den Wert 0x100 ins PIOB_PER und dann ins PIOB_OER. Danach schreiben Sie nacheinander den Wert 0x100 einige Male alternierend ins PIOB_SODR und PIOB_CODR. Dadurch sollte die LED DS1 auf dem Board AT91EB63 an und aus gehen. Testen Sie dieses mit Einzelschritten aus.

Aufgabe 2:

Gut, wir können jetzt eine LED (DS1) kontrollieren. Schauen Sie sich die Dateien im Verzeichnis ~/mpsSS2013/h an. Sie können diese Header-Dateien in Ihren nächsten Programmen benutzen.

Ändern und erweitern Sie Ihr Programm so, dass die LED DS1 durch drücken der Taste SW1 eingeschaltet und durch drücken der Taste SW2 ausgeschaltet wird. Sollte die Taste nicht funktionieren, so überprüfen Sie ob im Power Management Controller (PMC) der Clock für PIOB eingeschaltet ist. In welchem Register muss welches Bit gesetzt sein?

Aufgabe 3:

Lassen Sie im nächsten Programm zusätzlich die LED DS2 mit ca. 0,5Hz (Zeitschleife programmieren) blinken. Wie reagieren Ihre Tastendrücke an SW1 und SW2?

Aufgabe 4:

Schreiben Sie für die Tasten SW1 und SW2 eine passende Interruptserviceroutine. Erklären Sie die Wechsel der ARM-Betriebsmodi. Wie reagieren nun Ihre Tastendrücke an SW1 und SW2? Welchen Einfluss hat das Bedienen der Tasten auf die Blinkfrequenz von DS2?

Aufgabe 5:

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. Denken Sie daran, dass einige Programmteile nochmals benötigt werden.

```
// Lösung zu Termin 2
// Aufgabe 1
// Namen: _____; _____
// Matr.: _____; _____
// vom :  _____
//

// Beispiel einer Zeigervariablen
#define PIOB_PER      ((volatile unsigned int *) 0xFFFF0000)


int main(void)
{
// Beispiel des Beschreibens eines Register über Zeigervariable
        *PIOB_PER = 0x100;    // aktiviere Register LED's und Taster an PortB

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

        return 0;
}
```

```
// Lösung zu Termin 2
// Aufgabe 2
// Namen: _____; _____
// Matr.: _____; _____
// vom :  _____
//
#include "../h/pmc.h"
#include "../h/pio.h"
int main(void)
{
  StructPMC* pmcbase = PMC_BASE;          // Basisadresse des PMC
  pmcbase->PMC_PCER = 0x4000;             // Peripheral Clocks aktiv für PIOB

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


        return 0;
}
```

```
// Lösung zu Termin2
// Aufgabe 4
// Namen: _____; _____
// Matr.: _____; _____
// vom :  _____
//

#include "../h/pmc.h"
#include "../h/pio.h"
#...

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

void taste_irq_handler (void)
{

}

int main(void)
{

        return 0;
}
```

Termin 3

Lernziele:

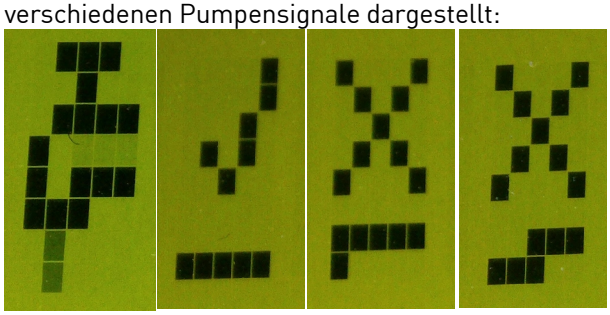
Mit den folgenden Versuchen sollen Sie lernen, wie Sie aus der Sprache "C" Peripherie (z. B. Timer) von modernen Mikrocontrollern nutzen.

Arbeitsverzeichnis:

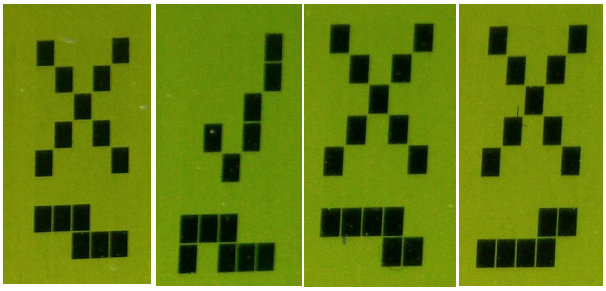
Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsSS2013. Dort finden Sie zu jedem Termin vorgegebene Dateien.

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 Sie auch die noch fehlenden Kommentare.

Aufgabe 2:

Erweitern Sie das Programm so, dass Sie die Tasten SW1, SW2 und SW3 abfragen können. Mit der Taste SW2 soll die Pumpe eingeschaltet, mit der Taste SW1 soll die Pumpe abgeschaltet und mit der Taste SW3 soll das Programm beendet werden.

Aufgabe 3:

Isolieren Sie die Routinen , welche für den Termin6, 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, der 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;              // 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
                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* piobase  = PIOB_BASE;        // Basisadresse PIO B

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

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

        while(1)
        {

        }
                return 0;
}
```

Termin 4

Lernziele:

Sie sollen heute lernen wie mit Zählern im Capture-Mode Zeitmessungen realisiert werden können.

Arbeitsverzeichnis:

Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsSS2013. Dort finden Sie zu jedem Termin vorgegebene Dateien. Ein Schaltplan der Anbindung von Waage und Pumpe finden sie in der Datei **interface.pdf.**

Weitere Infos:

Ab dem Sommersemester 2011 stehen an jedem Laborarbeitsplatz WaSim (Waagensimulatoren) zur Verfügung. Die WaSim liefern zwei, entsprechend des zu simulierenden Gewichtes, sich verändernde Frequenzen im Bereich von ca. 500Hz. Diese beiden Signale können direkt auf die Timereingänge (PA7 und PA4) der Evaluierungsplatine AT91EB63 gegeben werden.

Aufgabe 1:

Machen Sie sich mit dem Programm aus ***Termin4-Aufgabe1.c*** vertraut. Übersetzen und testen Sie das Programm. Wie groß ist die gemessene Periodendauer an T_{PA4}? Verändert sich die Periodendauer bei Belastung der Waage?

ACHTUNG: Die Waage darf nur mit max. 1000g (1kg) belastet werden!

Aufgabe 2:

Erweitern Sie das Programm so, damit auch die Periodendauer an T_{PA7} erfasst wird. Testen Sie auch hier ob sich die Periodendauer um die 2ms bewegt.

Aufgabe 3:

Aus dem Verhältnis der beiden Frequenzen f_{PA7}, f_{PA4} wird die Masse **m** nach der Gleichung

$$m = C1 * ((f_{PA7} / f_{PA4}) - 1) - C2$$

oder

$$m = C1 * ((T_{PA4} / T_{PA7}) - 1) - C2$$

errechnet. Die Schwingfrequenzen f_{PA7}, f_{PA4} der Saiten liegen bei etwa
16kHz / 32 = 500Hz.
Die beiden Größen C1 und C2 sind wägezellenspezifische Konstanten.

Die Frequenz an PA4 wird mit zunehmendem Gewicht kleiner. Die Frequenz an PA7 wird mit zunehmendem Gewicht größer.

ACHTUNG: Die Waage darf nur mit max. 1000g belastet werden!

Berechnen Sie nun noch die Masse **m** welche auf der Waage liegt. Setzen Sie für C1 und für C2 die für die Waage angegebenen Werte ein. Überprüfen Sie anhand der gegebenen Gewichte ob Ihre Messwerte stimmen.

Aufgabe 4:

Schreiben Sie für die Messung der Masse eine Funktion

int MessungderMasse(void)

welche die ermittelte Masse als integer Wert in g (Gramm) liefert.

Denken Sie auch diesmal daran, dass Programmteile zur Lösung des Gesamtprojektes am Ende wieder benötigt werden.

Aufgabe 5:

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

// Lösung zur Aufgabe Termin 4

```
// Aufgabe 1
//****************************
// Messen der Periodendauer einer angelegten Frequenz
// von: Manfred Pester
// vom: 06. August 2003
// Achtung: Programm hat noch Fehler

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

// für die Initialisierung des Zähler TC4

#define TC4_INIT  TC_CLKS_MCK2 | TC_LDBSTOP | TC_CAPT | TC_LDRA_RISING_EDGE | TC_LDRB_RISING_EDGE

int main(void)
{
        volatile int captureRA1;
        volatile int captureRB1;
        volatile int capturediff1;
        volatile float Periodendauer1;

        StructPMC* pmcbase = PMC_BASE;
        StructPIO* piobaseA = PIOA_BASE;
        StructTC* tcbase4 = TCB4_BASE;
        StructTC* tcbase5 = TCB5_BASE;

        pmcbase->PMC_PCER = 0x06f80;          // Clock PIOA, PIOB, Timer5, Timer4, Timer1 einschalten

// Periodendauer der Waagensignale messen
// Signal an TIOA4 ca. 16kHz entspricht ca. einer Periodendauer von 62,5us
// durch den Teiler von 32 ergeben sich ca. 2ms
// Zähler mit positiver Flanke starten

        piobaseA->PIO_PDR = 0x090;
        tcbase4->TC_CCR = TC_CLKDIS;
        tcbase4->TC_CMR = TC4_INIT;
        tcbase4->TC_CCR = TC_CLKEN;
        tcbase4->TC_CCR = TC_SWTRG;

        while(!(piobaseB->PIO_PDSR & KEY3))
        {
                tcbase4->TC_CCR = TC_SWTRG;
                while (!( tcbase4->TC_SR & 0x40)); // Capture Register B wurde geladen Messung abgeschlossen
                        captureRA1 = tcbase4->TC_RA;              //
                        captureRB1 = tcbase4->TC_RB;
                        capturediff1 = abs(captureRB1) - abs(captureRA1);
                        Periodendauer1 = abs(capturediff1) / 12.5;              // Zeit in us
        }

        return 0;
}
```

Termin 5

Lernziele:

Die Programmierung von Funktionen die wiederum andere Funktionen aufrufen. Die Kenntnis der Basistechnologie zur Implementierung einer Schnittstelle zwischen Anwendungsprogrammen und Betriebssystemen. Die Bedeutung und Anwendung von Supervisor- und User-Mode.

Arbeitsverzeichnis:

Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsSS2013. Dort finden Sie zu jedem Termin vorgegebene Dateien.

Aufgabenstellung:

Der SWI Befehl führt zu einer Ausnahmebehandlung im Prozessor die mit einem Wechsel in den Supervisor Mode verbunden ist. Dem SWI Befehl kann beim Aufruf eine bis zu 23 Bit große Zahl übergeben werden, die der SWI Handler dazu benutzen kann die gewünschte Funktion auszuwählen.

In der Aufgabe soll ein SWI Handler genutzt werden, um eine Trennung des Low Level IOs vom Anwendungsprogramm zu erreichen. Dazu sind die Funktionen die den SW-Interrupt aufrufen in Assembler zu implementieren und mit dem Debugger ist die Funktionsweise des Interrupthandlers zu untersuchen.

Aufgabe 1:

Nehmen Sie die zur Verfügung gestellten Dateien in ein neues Projekt auf, testen und dokumentieren Sie dieses.

Die erzeugten Ausgaben von CR und LF sollten auf einem Terminal an der seriellen Schnittstelle zu sehen sein. Verwenden Sie in einer separaten Shell das Programm ***„minicom“*** als Terminalersatz.

Aufgabe 2:

Sie sollen die Funktion puts (siehe ***ser_io.S***) in Assembler so ergänzen, dass auch der String auf die serielle Schnittstelle ausgegeben wird. Die Intialisierungs-Funktion init_ser und die IO-Funktionen putch und getch (siehe ***seriell.c***) werden dabei über einen SWI (siehe ***swi.c***) aufgerufen.

void puts(char *) Ausgabe eines nullterminierten Strings und Ersetzung von Newline durch Carriage Return (0x0d) und Linefeed (0x0a).

Aufgabe 3:

Erklären Sie den Code des SWI-Handlers (siehe swi.c/swi.S). Debuggen Sie Ihr Programm und lokalisieren Sie die Umschaltung vom User Mode in den Superuser Mode und zurück. Dokumentieren Sie sich die Vorgänge.

Aufgabe 4:

Entwickeln Sie eine Routine mit der Sie sich eine vorzeichenbehaftete Integerzahl (z.B. die aus dem vorigen Termin ermittelte Masse) dezimal auf das Terminal ausgeben lassen können.

Aufgabe 5:

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.

```
// FileName:  Termin5Aufgabe1.c
// Programmrahmen zur Aufgabe Termin5
// Aufgabe 1
//************************************
//
// von: Manfred Pester
// vom: 06. August 2003
// letzte Änderung:   30. November 2004
// von: Manfred Pester

int main(void)
{
        char i;
// Serielle initialisieren
        inits();
// CR und LF auf das Terminal ausgeben
        putc (0xd);
        putc (0xa);
// ein Zeichen von der seriellen abholen
        i=getc();
        putc(i);
// String ausgeben
        puts("Hallo! \n");

        return 0;
}
```

```
/*-----------------------------------------------------------------------------
@ File Name:          seriell.c
@ Object:             Grundfunktionen der seriellen Schnittstelle
@          int init_ser(); char putch(char); char getch();
@
@ Autor:   M.Pester
@ Datum: 04.12.2007
@-----------------------------------------------------------------------------*/
#include "../h/pmc.h"
#include "../h/pio.h"
#include "../h/usart.h"
#include <stdio.h>

int init_ser(void);
char putch(char);
char getch(void);

#define    DEFAULT_BAUD = 38400
#define    CLOCK_SPEED = 25000000
//US_BAUD =  (CLOCK_SPEED / (16*(DEFAULT_BAUD))+0.5    // 25MHz / ( 16 * 38400) = 40.69  -> 41 -> 0x29
#define    US_BAUD = 0x29

// Initialisiert die serielle Schnittstelle USART0
int init_ser()
{
          StructPIO* piobaseA = PIOA_BASE;
          StructPMC* pmcbase  = PMC_BASE;
          StructUSART* usartbase0 = USART0;

          pmcbase->PMC_PCER = 0x4;  // Clock für US0 einschalten
          piobaseA->PIO_PDR = 0x18000;          // US0 TxD und RxD
          usartbase0->US_CR = 0xa0;    // TxD und RxD disable
          usartbase0->US_BRGR = 0x29;          // Baud Rate Generator Register
          usartbase0->US_MR = 0x8c0;  // Keine Parität, 8 Bit, MCKI
          usartbase0->US_CR = 0x50;    // TxD und RxD enable

          return 0;
}

// Gibt wenn möglich ein Zeichen auf die serielle Schnittstelle aus
// und liefert das Zeichen wieder zurück
// wenn eine Ausgabe nicht möglich war wird eine 0 zurück geliefert
char putch(char Zeichen)
{
          StructUSART* usartbase0 = USART0;

          if( usartbase0->US_CSR & US_TXRDY )
          {
                    usartbase0->US_THR = Zeichen;
                    return Zeichen;
          }
          else
          {

                    return 0;
          }
}

// Gibt entweder ein empfangenes Zeichen oder eine 0 zurück
char getch(void)
{
          StructUSART* usartbase0 = USART0;
          char Zeichen;

          if( usartbase0->US_CSR & US_RXRDY )
                    return usartbase0->US_RHR;
          else
                    return 0;

}
```

```
@-------------------------------------------------------------------------------
@ File Name:        ser_io.S
@ Object:           Ein- Ausgabe-Funktionen der seriellen Schnittstelle
@                   welche ueber den Supervisor-Mode gehen
@
@ Namen :                   Matr.-Nr.:
@
@-------------------------------------------------------------------------------
@ Debuginformationen
        .file   "ser_io.S"
@ Funktion
        .text
        .align  2
        .global inits
        .type   inits,function
inits:
        swi     0x100
        bx      lr          @ Rücksprung
@ Funktion
        .text
        .align  2
        .global putc
        .type   inits,function
putc:
        mov     r1, r0              @ Zeichen nach r1
        ldr     r0, =Zeichen        @ Zeiger holen
        str     r1, [r0]    @ Zeichen unter Zeiger ablegen
        swi     0x200               @
        ldr     r1, =Zeichen        @ Zeiger holen
        ldr     r0, [r1]    @ Zeichen aus Zeiger holen
        bx      lr
@ Funktion
        .text
        .align  2
        .global getc
        .type   inits,function
getc:
        ldr     r0, =Zeichen        @ Zeiger holen
        swi     0x300
        ldr     r0, =Zeichen        @ Zeiger holen
        ldr     r0, [r0]    @ empfangenes Zeichen zurueck geben
        bx      lr
@ Funktion
        .text
        .align  2
        .global puts
        .type   puts,function
puts:
        stmfd sp!,{lr}      @ Retten der Register

// Hier muß Ihr Code eingefügt werden.

        ldmfd sp!,{pc}      @ Rücksprung
@ Funktion
        .text
        .align  2
        .global gets
        .type   gets,function
gets:
        stmfd   sp!,{lr}    @ Retten der Register

// Hier könnte Ihr Code eingefügt werden!

        ldmfd   sp!,{pc}    @ Rücksprung

        .data
Zeichen:  .word 0
.end
```

```
/*-------------------------------------------------------------------------
@ File Name:        swi.c
@ Object:           SoftwareInterruptHandler
@
@ Autor:            Horsch/Pester
@ Datum:            3.12.2007/Januar2011
@-------------------------------------------------------------------------*/
void SWIHandler () __attribute__ ((interrupt ("SWI")));

void SWIHandler()
{
        register int reg_r0 asm ("r0");
        register int *reg_14 asm ("r14");

        switch( *(reg_14 - 1) & 0x00FFFFFF)     // Maskieren der unteren 24 Bits

                                                // und Verzweigen in Abh. der SWI Nummer
        {
                case 0x100:
                        init_ser();
                        break;
                case 0x200:
                        *((char *)reg_r0) = putch(*((char *)reg_r0));
                        break;
                case 0x300:
                        *((char *)reg_r0) = (unsigned int) getch();
                        break;
        }
}

# Vorschlag eines Makefile zu Termin5 SS2011

FILE = Termin5Aufgabe1
Opti = 1

all:

# uebersetzen der Quelldatei
        arm-elf-gcc -c -g -O$(Opti) $(FILE).c -I ../h

# Erzeugen einer Assemblerdatei aus der Quelldatei
        arm-elf-gcc -S -O$(Opti) $(FILE).c -I ../h
        arm-elf-gcc -S -O$(Opti) seriell.c -I ../h
        arm-elf-gcc -S -O$(Opti) swi.c -I ../h

# Erzeugen der benoetitgen Objektdateien
# eigener SoftWareInterrupt-Handler
        arm-elf-gcc -c -g -O$(Opti) swi.c -o swi.o -I ../h
        arm-elf-gcc -c -g -O$(Opti) seriell.c -o seriell.o -I ../h
        arm-elf-gcc -c -g -O$(Opti) ser_io.S -o ser_io.o -I ../h
        arm-elf-gcc -c -g -O$(Opti) ../boot/boot_ice.S -o boot_ice.o -I ../h

# Binden fuer die RAM-Version
#       arm-elf-ld -Ttext 0x02000000 -O$(Opti) boot_ice.o swi.o seriell.o ser_io.o $(FILE).o -o $(FILE).elf /usr/local/arm-
elf/lib/gcc/arm-elf/4.3.1/libgcc.a
        arm-elf-ld -Ttext 0x02000000 -O$(Opti) boot_ice.o swi.o seriell.o ser_io.o $(FILE).o -o $(FILE).elf /usr/local/arm-
elf/lib/gcc/arm-elf/4.3.1/libgcc.a

clean:
        rm *.o
        rm *.s
        rm *.elf
        rm *.rom
```

Termin 6

Arbeitsverzeichnis:

Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsSS2013. Dort finden Sie zu jedem Termin vorgegebene Dateien. Die benötigten Dateien, Programme und Funktionen sollten Sie aus den Terminen 1-5 mitbringen.

Lernziele:

Sie sollen aus dem Gelernten und Gesammelten eine Ausschankstation realisieren.

Die Ausschankstation soll folgende Funktionen haben:

Aufgabe 1:

Initialisierung der benötigten Peripherie.

Aufgabe 2:

Begrüßung und Informationen.

Aufgabe 3:

Wiegen, kalibrieren und anzeigen des Gefäßgewichtes nach Aufstellen eines Bechers oder Glases und Drücken der Taste SW1.

Aufgabe 4:

Abfüllen von 50ml (Gramm) nach drücken der Taste SW2.
Achtung sollte das Gefäß zu früh weg genommen werden!

Aufgabe 5:

Meldung nach dem Abfüllvorgang welche Menge abgefüllt wurde und das der Becher weg genommen werden kann.

Aufgabe 6:

Wenn der Becher weg genommen ist soll von vorne (Aufgabe 2 evtl. Aufgabe1) begonnen werden.

Aufgabe 7:

- Dokumentieren Sie Ihre Lösung.
- Erstellen Sie ein Benutzerhandbuch.
- Verkaufen Sie Ihre Lösung dem zuständigen Laborbetreuer.
- Sind Sie auf einige Fragen des zuständigen Betreuers vorbereitet.
- Schauen Sie, dass Sie in der Lage sind auf kleine Änderungswünsche reagieren zu können.

Zusatzaufgabe:

Erweitern Sie die Lösung so, dass auch über die Konsole (minicom) die Ausschankstation bedient werden kann. Also die Tasten vom Board nicht mehr benötigt werden.