



BLOCKVERANSTALTUNG MIKROPROZESSORPRAKTIKUM

WS1415Block

Termin 2

C-Programmierung für eingebettete Systeme Pointer, Peripherie, PIO,
Interrupt

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

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

Lernziele:

Erster Umgang mit Peripherie in eingebetteten Systemen.

Arbeitsverzeichnis:

Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsWS1415Block. 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 ~/mpsWS1415Block/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 Tastendrucke 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 Tastendrucke 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
    StructPIO* piobaseB = PIOB_BASE; // Basisadresse PIO B
    pmcbase->PMC_PCER = 0x4200; // 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;
}
```