



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

MIKROPROZESSORPRAKTIKUM

SS2014

Termin 5

C-Programmierung für eingebettete SystemePointer, Peripherie, USART,
SWI

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

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

Arbeitsverzeichnis:

Kopieren Sie sich aus dem Ordner /mnt/Originale das Verzeichnis mpsSS2014. Dort finden Sie zu jedem Termin vorgegebene Dateien.

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.

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 Initialisierungs-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 im CPSR.

Aufgabe 4:

Entwickeln und **testen** Sie eine Routine mit der Sie sich eine vorzeichenbehaftete Integerzahl 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. Denken Sie daran, dass zum letzten (sechsten Termin) eine Dokumentation erstellt werden muss.

```
// 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))    // 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 putc,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 getc,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) & 0x0FFFFFFF)    // Maskieren der unteren 24 Bits  
  
        // und Verzweigen in Abh. der SWI Nummer  
    {  
        case 0x100:  
            init_ser();  
            break;  
        case 0x200:  
            *((char *)reg_r0) = putchar(*((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 benoetigten 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