Grundlagen der Informatik 2
Modul „Systemnahe Programmierung in C (SPiC)“

SS 2008

Dr.-Ing. Jürgen Kleinöder
Friedrich-Alexander-Universität Erlangen-Nürnberg
Informatik 4 (Verteilte Systeme und Betriebssysteme)
Martensstr. 1, 91058 Erlangen

Klausur am 25. Juli 2008

Angaben zur Person:

Nachname	
Vorname	
Matrikelnummer	
Studiengang	
Fachsemester	

(X)	Ich fühle mich gesund und bin in der Lage die Klausur zu schreiben
(X)	Die wichtigen Hinweise auf der nächsten Seite habe ich zur Kenntnis genommen
(X)	Ich habe die Klausur alleine und ohne fremde Hilfe bearbeitet

Unterschrift	

Aufgabe 1	**Aufgabe 2**			**Aufgabe 3**			**Summe**
	I	**L**		**a:**	**b:**	**c:**	

Wichtige Hinweise

Die folgenden Informationen bitte aufmerksam lesen und die Erklärung auf Seite 1 unterschreiben.

- Es sind **keine** eigenen Hilfsmittel zugelassen.
- Die Heftung der Klausur darf nicht aufgelöst werden.
- Die Bearbeitungszeit für diesen Teil der Klausur beträgt 45 Minuten.
- Prüfen Sie Ihr Exemplar der Klausur auf Vollständigkeit!
  Dieser Teil der Klausur hat insgesamt 10 Seiten (inkl. Deckblatt).
- Die Lösung einer Aufgabe muss auf das Aufgabenblatt in den dafür vorgesehenen Raum geschrieben werden. Sollte der Platz nicht ausreichen, können Sie die Rückseiten der Aufgabenblätter mitverwenden. Kennzeichnen Sie dabei die Zugehörigkeit Ihrer Lösung zu einer Aufgabe deutlich. Zusätzliche Blätter gehen nicht in die Bewertung ein!
- Die Lösungen müssen dokumentenecht in blau oder schwarz geschrieben werden. Als falsch Erkanntes muss deutlich durchgestrichen werden. Tintenkiller darf nicht verwendet werden. Keinen Bleistift verwenden!
- Fragen zu den Prüfungsaufgaben können grundsätzlich nicht beantwortet werden.
- Tragen Sie Ihren Namen und Vornamen, Ihre Matrikelnummer, Studiengang und Fachsemesterzahl auf dem Deckblatt der Klausur ein.
- Bitte legen Sie Ihren Studenten- und einen Lichtbildausweis zur Kontrolle bereit.
- Sie dürfen den Raum nicht verlassen bevor Ihre Personalien überprüft wurden und Sie die Klausurunterlagen der Aufsicht zurückgegeben haben.
- In den letzten 15 Minuten der Bearbeitungszeit können Sie den Raum nicht mehr verlassen. Bleiben Sie an Ihrem Platz sitzen, bis am Ende alle Klausurunterlagen eingesammelt und nachgezählt sind und die Aufsicht das Zeichen zum Gehen gibt.

Die **Klausurergebnisse** werden in ca. zwei Wochen im WWW unter der Seite der Vorlesung im SS 2008, Unterpunkt "Ergebnisse" veröffentlicht.

Aufgabe 1: (10 Punkte)

Bei den Multiple-Choice-Fragen ist jeweils nur **eine** richtige Antwort eindeutig anzukreuzen. Auf die richtige Antwort gibt es die angegebene Punktzahl.

Wollen Sie eine Multiple-Choice-Antwort korrigieren, kreisen Sie bitte die falsche Antwort ein und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten.

a) Welche Aussage zu Zeigern ist **richtig**? 2 Punkte

- ❏ Die Übergabesemantik für Zeiger als Funktionsparameter ist call-by-value.
- ❏ Die Übergabesemantik für Zeiger als Funktionsparameter ist call-by-reference.
- ❏ Ein Zeiger kann zur Manipulation von schreibgeschützten Datenbereichen verwendet werden.
- ❏ Zeiger vom Typ `void*` sind am besten für Zeigerarithmetik geeignet, da Sie kompatibel zu jedem Zeigertyp sind.

b) Welche der folgenden Aussagen über den C-Präprozessor ist **richtig**? 2 Punkte

- ❏ Der Präprozessor ist eine Softwarekomponente, welche Java-Klassen durch C-Funktionen ersetzt, die dann von einem C-Compiler übersetzt werden.
- ❏ Der Präprozessor optimiert Makros durch Zeigerarithmetik.
- ❏ Nach dem Übersetzen und dem Binden müssen C-Programme durch den Präprozessor nachbearbeitet werden, um Makros aufzulösen.
- ❏ Die Syntax von Präprozessoranweisungen ist unabhängig vom Rest der Sprache C.

c) In Betriebssystemen wie Linux oder Windows unterscheidet man die Begriffe Programm und Prozess. Welche Aussage ist **richtig**? 2 Punkte

- ❏ Programme sind Anwendungen der Benutzer, während Prozesse Aktivitäten des Betriebssystems sind.
- ❏ Programme sind C-Quellcode-Dateien, die durch einen C-Compiler in einen lauffähigen Prozess übersetzt werden können.
- ❏ Ein Prozess hat einen eigenen virtuellen Adressraum. Daten des Prozesses sind vor direktem Zugriff durch andere Prozesse geschützt.
- ❏ Ein Programm ist ein Prozess in Ausführung.

d) Welche der folgenden Aussagen bzgl. der Interruptsteuerung **trifft zu**? 2 Punkte

- ❏ Pegel-gesteuerte Interrupts werden beim Wechsel des Pegels ausgelöst, daher der Name.
- ❏ Interrupts sind eine Besonderheit von AVR-Mikroprozessoren. Auf anderen Architekturen kommen POSIX-Signale zum Einsatz.
- ❏ Pegelgesteuerte Interrupts müssen durch Polling des Pegels abgefragt werden.
- ❏ Wurde gerade ein Flanken-gesteuerter Interrupt ausgelöst, so muss erst ein Pegelwechsel der Interruptleitung stattfinden, damit erneut ein Interrupt ausgelöst werden kann.

e) Gegeben ist folgendes Makro: 2 Punkte

```
#define ADD(a,b) a+b
#define MUL(a,b) a*b
```

Was ist das Ergebnis von folgendem Ausdruck:

```
4 * MUL( ADD(1,2), 3)
```

- ❏ 10
- ❏ 18
- ❏ 28
- ❏ 36

Aufgabe 2: (22 Punkte)

Schreiben Sie ein Lauflicht-Programm für einen AVR-Mikrokontroller mit 8 LEDs an Port A. Es soll immer *genau eine LED aktiv* sein, im Ausgangszustand die LED an Pin 0. Das Programm stoppt nach Einschalten der LED an Pin 7. Mit einem Taster kann ein neuer Durchlauf gestartet werden. Wird der Taster bereits während des Durchlaufs gedrückt, so stoppt das Programm nicht, sondern fängt unmittelbar nach LED 7 wieder mit einem neuen Durchlauf bei LED 0 an.

Im Detail soll sich das Programm wie folgt verhalten:

- Die Initialisierung der Hardware soll in einer eigenen Funktion `void init()` erfolgen, die zu Beginn des Programms aufgerufen wird. Es können hierbei keine Annahmen über den initialen Zustand der Hardware gemacht werden.
- Anschließend erfolgt ein kompletter Lauflichtdurchlauf.
- Zum Warten zwischen dem Umschalten zweier LEDs wird die Funktion `void wait(unsigned int interval)` aufgerufen. In dieser Funktion wird aktiv in einer Warteschleife mit `interval` Durchläufen gewartet. Für `interval` wird konstant der Wert 5000 übergeben.
- Nach einem Lauflichtdurchlauf wird der Mikrokontroller in den Standard-Stromsparmodus versetzt falls zwischenzeitlich die Taste nicht gedrückt wurde. Durch einen Tastendruck soll der Mikrokontroller aus dem Stromsparmodus geweckt werden können und einen neuen Lauflichtdurchlauf beginnen.

Information über die Hardware

LEDs: **PORTA**, Pins 0-7, Start bei LED an Pin 0
- Pin als Ausgang konfigurieren: entspr. Bit in **DDRA**-Reg. auf 1

Taster: **PORTD**, Pin 2
- Pin als Eingang konfigurieren: entspr. Bit in **DDRD**-Reg. auf 0
- externe Interruptquelle **INT0**, ISR-Vektor-Makro: **INT0_vect**
- Aktivierung der Interruptquelle erfolgt durch Setzen des **INT0**-Bits im Register **GICR**.
- Taster verbindet den Pin mit Masse, es muss der interne Pullup-Widerstand verwendet werden (entspr. Bit in **PORTD**-Reg. auf 1 setzen).
- Konfiguration der externen Interruptquelle 0 (Bits in Register **MCUCR**)

ISC01	ISC00	Beschreibung
0	0	Interrupt bei low Pegel
0	1	Interrupt bei beliebiger Flanke
1	0	Interrupt bei fallender Flanke
1	1	Interrupt bei steigender Flanke

Ergänzen Sie das folgende Codegerüst so, dass ein vollständig übersetzbares Programm entsteht.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

/* Funktiondeklarationen, globale Variablen, etc. */

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
/* Unterbrechungsbehandlungsfunktion */

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------




/* Funktion main */

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
  /* Initialisierung */

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
```

```
  /* Hauptschleife */

.....

.....

.....

.....

.....

    /* ein Lauflicht-Durchlauf */

.....

.....

.....

.....

.....

    /* Vorbereitung des naechsten Durchlaufs bzw. Schlafen */

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

/* Ende der Funktion main */
```

L:

```
/* Funktion init */

------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------

/* Ende der Funktion init */
/* Funktion wait() */

------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------

/* Ende der Funktion wait() */
```

I:

Aufgabe 3: (13 Punkte)

Die folgenden Beschreibungen sollen kurz und prägnant erfolgen (Stichworte, kurze Sätze)

a) Beschreiben Sie die Unterschiede bei der Ausführung eines Programms auf einem Mikrokontroller ohne Betriebssystem und auf einem Betriebssystem wie z. B. Linux.

b) Beschreiben Sie den Unterschied zwischen Pegel- und Flanken-gesteuerten Interrupts.

c) Was passiert, wenn während der Bearbeitung eines Interrupts weitere Interrupts der gleichen Quelle eintreffen? Worauf muss man deshalb bei der Programmierung von Interrupt-Funktionen achten?