Aufgabe 1: (12 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, streichen Sie bitte die falsche Antwort mit drei waagrechten Strichen durch (☒)und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten.

a) Welche Aussage zum Thema Polling und Interrupts ist richtig? 2 Punkte

- ❏ Beim Pollen eines Gerätes muss man selektiv dessen Interrupts sperren, um explizit Daten zu synchronisieren.
- ❏ Interrupts haben den Nachteil, dass sie nicht mit Präprozessormakros funktionieren.
- ❏ Geräte, die flankengesteuerte Interrupts auslösen können, lassen sich grundsätzlich nicht pollen.
- ❏ Bei hochfrequenten Geräteereignissen erzeugt Polling eine hohe CPU-Last.

b) Gegeben sei folgender Ausschnitt eines C-Programms, welches eine Variable foo vom Typ `uint8_t` verwendet: 2 Punkte

```
foo &= ~0xaa;
foo ^=  0xaa;
```

Welche Aussage über foo ist nach der Ausführung der Anweisungen richtig?

- ❏ Das höchstwertige Bit in foo ist 1.
- ❏ Die ersten beiden Zeichen in der Zeichenkette foo sind ’aa’.
- ❏ Das niederwertigste Bit in foo ist 1.
- ❏ Über den Zustand des höchstwertigen Bits von foo kann keine Aussage getroffen werden.

c) Wozu dient das `#ifdef`-Konstrukt in C? 2 Punkte

- ❏ Es kann alternativ zu if-Abfragen eingesetzt werden.
- ❏ Man kann damit Programmteile bei der Übersetzung ausblenden.
- ❏ Es kann eingesetzt werden, um sicherzustellen, dass ein Programmteil ein definiertes Ergebnis liefert.
- ❏ Es überprüft, ob die danach angegebenen Variablen definiert wurden.

d) Was versteht man unter den formalen und tatsächlichen Parametern einer Funktion? 2 Punkte

- ❏ Der tatsächliche Parameter enthält eine Kopie des Aufrufparameters, der formale Parameter ist ein Zeiger auf den Aufrufparameter.
- ❏ Die formalen Parameter sind die Namen, unter denen auf die Aufrufparameter innerhalb der Funktion zugegegriffen werden kann.
- ❏ Die tatsächlichen Parameter sind die Rückgabewerte eines Funktionsaufrufs.
- ❏ Die tatsächlichen Parameter sind die Namen, unter denen innerhalb einer Funktion auf die Aufrufparameter zugegriffen wird.

e) Was versteht man unter Nebenläufigkeit?

- ❏ Wenn ein Programm abwechselnd auf zwei verschiedene Speicherbereiche zugreift.
- ❏ Wenn ein Programmabschnitt in einer Schleife mehrfach durchlaufen wird.
- ❏ Die Programmabschnitte im if- und else-Teil einer bedingten Anweisung.
- ❏ Wenn für zwei Befehle aus zwei Programmabläufen nicht feststeht, welcher von beiden tatsächlich zuerst ausgeführt werden wird.

f) Was versteht man beim Zugriff auf I/O-Register unter dem Begriff "Memory-mapped"?

- ❏ Der Zugriff auf die Register erfolgt mit speziellen mmap-Instruktionen des Prozessors.
- ❏ Die Register sind in den normalen Adressraum des Prozessors eingeblendet und der Zugriff erfolgt mit den normalen Speicherzugriffsinstruktionen.
- ❏ Beim Zugriff auf spezielle Speicherbereiche des Hauptspeichers werden die Inhalte der Hauptspeicherzellen automatisch in Geräteregister umkopiert.
- ❏ Die Register sind nicht real, sondern nur virtuell im Hauptspeicher vorhanden (sog. "virtual devices").

Aufgabe 2: dimmer (30 Punkte)

Sie dürfen diese Seite zur besseren Übersicht bei der Programmierung heraustrennen!

Schreiben Sie eine Steuerung für dimmbare LEDs für ein AVR-Mikrocontroller-Board. Zu jedem Zeitpunkt ist genau eine von acht LEDs aktiv und leuchtet mit einer Helligkeitsstufe zwischen 0 und 10. Die Stufe wird vom Benutzer mit einem Plus-Taster (Taster 0) und einem Minus-Taster (Taster 1) reguliert. Wenn die Stufe 0 erreicht wird, wird die LED ausgeschaltet. Beim nächsten Druck auf den Plus-Taster wird dann gleichzeitig auf die nächste der acht LEDs umgeschaltet und diese mit Stufe 1 eingeschaltet. Nach der achten LED wird wieder auf die erste gewechselt.

Im Detail soll Ihr Programm wie folgt funktionieren:

- Initialisieren Sie die Hardware in der Funktion **`void init(void)`**.
- Das Programm startet mit LED 1 in Stufe 0 (ausgeschaltet); bei Stufe 0 soll der Mikrocontroller jeweils in den Schlafmodus gehen.
- In den Stufen 1 bis 10 leuchtet die gerade aktive LED und ist auf die entsprechende Stufe gedimmt. Das Dimmen wird durch schnellen Wechsel des An- und Aus-Zustands der aktiven LED implementiert. Dazu soll in einer Periode von 512 Mikrosekunden die LED einen zur Stufe proportionalen Teil eingeschaltet sein. Beispiel Stufe 7: 70% von 512 Mikrosekunden eingeschaltet, 30% ausgeschaltet.
- Implementieren Sie das Warten zwischen den Ein-/Aus-Anteilen einer Periode in einer aktiven Wartefunktion **`void wait(uint16_t us)`**, die **`us`** Mikrosekunden wartet. Ihnen steht eine Präprozessorkonstante **`LOOPS_PER_US`** zur Verfügung, die angibt, wieviele Schleifendurchläufe gewartet werden muss, um eine Mikrosekunde verstreichen zu lassen.

Information über die Hardware

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

Taster: **PORTD**, Plus-Taster (Taster 0) = Pin 2, Minus-Taster (Taster 1) = Pin3
- Pins als Eingang konfigurieren: entspr. Bit in **DDRD**-Reg. auf 0
- externe Interruptquellen **INT0** und **INT1**, ISR-Vektor-Makros **INT0_vect** und **INT1_vect**
- Aktivierung der Interruptquellen erfolgt durch Setzen des **INT0**- bzw. **INT1**-Bits im Register **GICR**.
- die Taster verbinden den Pin mit Masse, es müssen die internen Pullup-Widerstände verwendet werden (entspr. Bits in **PORTD**-Reg. auf 1 setzen).
- Konfiguration der externen Interruptquellen 0 und 1 (Bits in Register **MCUCR**)

Interrupt 0		Beschreibung	Interrupt 1	
ISC01	**ISC00**		**ISC11**	**ISC10**
0	0	Interrupt bei low-Pegel	0	0
0	1	Interrupt bei beliebiger Flanke	0	1
1	0	Interrupt bei fallender Flanke	1	0
1	1	Interrupt bei steigender Flanke	1	1

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

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

#define LOOPS_PER_US 42


/* Funktionsdeklarationen, globale Variablen, etc. */

.....
.....
.....
.....
.....


/* Funktion main */

.....
.....
.....
.....

   /* Initialisierung und lokale Variablen */

.....
.....
.....
.....
.....
.....
.....
.....
```

A:

```
   /* Hauptschleife */
.....
      /* Schlafen */
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
      /* LED ansteuern */
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
} /* Ende Funktion main */
```

L:


```
/* Initialisierungsfunktion */
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
/* Wartefunktion */
.....
.....
.....
.....
.....
.....
.....
```

IW:

```
/* Unterbrechungsbehandlung */
```

U:

Aufgabe 3: (18 Punkte)

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

a) Was versteht man in der Informatik unter einem Modul und wozu dient es?
(4 Punkte)

b) Wie werden Module in C realisiert?
(4 Punkte)

c) Ein Mikrocontroller-Programm "main.c" verwende die Funktionen:

```
void send(char);
char receive(void);
```

eines Netzwerk-Treibers. Der Treiber implementiert diese Funktionen und verwaltet zudem einen Puffer für eingehende Zeichen und eine Unterbrechungsbehandlungsfunktion, die diesen befüllt:

```
char buf[32];
void receive_irq(void){
  ...
}
```

Skizzieren Sie, welche Dateien (netzwerk.*) erforderlich sind, um den Treiber in Form eines C-Moduls bereit zu stellen und im Hauptprogramm verwenden zu können. Was ist zu beachten (d.h. "was muss in welcher Datei wie stehen"), um eine korrekte Kapselung der Modulinterna sowie die korrekte Sichtbarkeit und Verwendung der Modulschnittstellen zu gewährleisten? (7 Punkte)

d) Aus Java kennen Sie zum Zwecke der Informationsverbergung (information hiding) die Sichtbarkeitsattribute "public", "protected" und "private". Nennen Sie - sofern vorhanden - die Entsprechungen in C! (3 Punkte)