Aufgabe 1: (14 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) Gegeben sei folgende Deklaration in einem C-Programm: 2 Punkte

```
typedef void (*func)(int, int);
```

Welche Aussage ist **richtig**?

- ❑ **`func`** ist eine Funktion, die hier durch forward-Deklaration dem Compiler bekannt gegeben wird.
- ❑ **`func`** ist ein Funktionszeiger auf eine **`void`**-Funktion, die zwei **`int`**-Parameter erwartet.
- ❑ **`func`** ist Datentyp für Funktionszeiger auf **`void`**-Funktionen, die zwei **`int`**-Parameter erwarten.
- ❑ Der Compiler meldet einen Fehler, weil bei der Deklaration von **`func`** die Namen der formalen Parameter fehlen.

b) Was ist ein Stack-Frame? 2 Punkte

- ❑ Ein Bereich des Speichers, in dem u.a. lokale automatic-Variablen einer Funktion abgelegt sind.
- ❑ Ein Fehler, der bei unberechtigten Zugriffen auf den Stack-Speicher entsteht.
- ❑ Der Speicherbereich, in dem der Programmcode einer Funktion abgelegt ist.
- ❑ Ein spezieller Registersatz des Prozessors zur Bearbeitung von Funktionen.

c) Was versteht man unter Polling? 2 Punkte

- ❑ Ein Konzept zur Abarbeitung von Interrupts.
- ❑ Wenn ein Programm zum Zugriff auf kritische Daten Interrupts sperrt.
- ❑ Das regelmäßige Anheben eines Pegels, um einem Gerät einen bestimmten Zustand zu signalisieren.
- ❑ Wenn ein Programm regelmäßig eine Peripherie-Schnittstelle überprüft, ob Daten oder Zustandsänderungen vorliegen.

d) Welche Aussage zu *volatile* ist richtig? 2 Punkte

- ❏ Das Schlüsselwort *volatile* unterbindet Optimierungen an einer damit definierten Variable.
- ❏ Das Schlüsselwort *volatile* unterbindet alle Nebenläufigkeitsprobleme.
- ❏ Das Schlüsselwort *volatile* hat nur noch eine historische Bedeutung und ist in heutigen Programmen nicht mehr nötig.
- ❏ Das Schlüsselwort *volatile* erlaubt dem Compiler bessere Optimierungen durchzuführen.

e) 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.

f) Gegeben sei folgender Programmcode: 2 Punkte

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

Was ist das Ergebnis von folgendem Ausdruck:

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

- ❏ 20
- ❏ 16
- ❏ 48
- ❏ Der Compiler warnt, weil das Ergebnis undefiniert ist.

g) Gegeben sei folgende Funktionsdeklaration: 2 Punkte

```
int f();
```

Welche Aussage ist richtig:

- ❏ **`f`** ist eine parameterlose Funktion vom Typ **`int`**
- ❏ **`f`** ist eine rekursive Funktion
- ❏ Der Aufruf **`f("Hallo Welt");`** wird vom Compiler als Fehler moniert
- ❏ Der Rückgabewert von **`f`** kann ignoriert werden

Aufgabe 2a: remote_control (30 Punkte)

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

Programmieren Sie eine Steuerung für eine Fernbedienung mit einem AVR-Mikrocontroller. Die Fernbedienung hat 8 Tasten, eine Infrarot-LED und eine rote LED zur Anzeige von Fehlern. Wird eine der Tasten gedrückt, wird durch Ein- und Ausschalten der Infrarot-LED in einem bestimmten Muster ein Signal gesendet. Wird mehr als eine Taste gleichzeitig gedrückt, leuchtet die rote LED zur Anzeige einer fehlerhaften Eingabe.
Im Detail soll Ihr Programm wie folgt funktionieren:

- Initialisieren Sie die Hardware in der Funktion `void init(void);`. Es sollen keine Annahmen über den initialen Zustand der Hardware-Register getroffen werden. Das Programm startet mit ausgeschalteten LEDs.
- Während des Wartens auf eine Eingabe soll der Mikrocontroller in den Schlafmodus wechseln.
- Wird eine der Tasten gedrückt, wird dadurch ein Interrupt ausgelöst. Welche Taste jeweils gedrückt wurde, kann in der Unterbrechungsbehandlung im entsprechenden Hardware-Register abgefragt werden. Wurde mehr als eine Taste gedrückt, soll die rote Fehler-LED für eine Sekunde leuchten.
- Implementieren Sie das Senden in der Funktion `void send(uint8_t button);`. Hier wird zuerst die Infrarot-LED für eine variable Dauer abhängig von der gedrückten Taste eingeschaltet, danach wird die Infrarot-LED für 500ms ausgeschaltet. Dieser Vorgang wird so oft wiederholt, wie es durch die Präprozessorkonstante `NBLINK` vorgegeben ist. Die Einschaltdauer der Infrarot-LED wird pro Taste in einem vorgegebenen Array `muster` festgelegt. Die Werte sind in Millisekunden angegeben.
- Implementieren Sie die Pausen unter Verwendung einer aktiven Wartefunktion `void wait(uint16_t ms);`, die `ms` Millisekunden wartet. Ihnen steht eine Präprozessorkonstante `LOOPS_PER_MS` zur Verfügung, die angibt, wieviele Schleifendurchläufe einer Millisekunde entsprechen.

Information über die Hardware

Infrarot-LED:**PORTC**, Pin 0, eingeschaltet bei low-Pegel
- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRC**-Register auf 1

Fehler-LED:**PORTB**, Pin 4, eingeschaltet bei low-Pegel
- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRB**-Register auf 1

Tasten: **PORTA**, alle Pins
- Pin als Eingang konfigurieren: entsprechendes Bit in **DDRA**-Register auf 0
- die Leitungen verbinden den Pin jeweils mit Masse, es muss der interne Pullup-Widerstand verwendet werden (entsprechendes Bit in **PORTA**-Register auf 1 setzen).
- Zustand kann durch Lesen des **PINA**-Registers abgefragt werden.
- Durch eine entsprechende Gatterschaltung wird durch einen Tastendruck auch die Interrupt-Leitung mit Masse verbunden, so dass jede Taste den Interrupt auslösen kann.

Interrupt-Leitung:**PORTD**, Interrupt-Leitung an Pin 2
- externe Interruptquelle **INT0**, ISR-Vektor-Makros: **INT0_vect**
- die Leitung verbindet den Pin mit Masse, es muss der interne Pullup-Widerstand verwendet werden (entsprechendes Bit in **PORTD**-Register auf 1 setzen).
- Aktivierung der Interruptquelle erfolgt durch Setzen des **INT0**-Bits im Register **GICR**.

Konfiguration der externen Interruptquelle 0 (Bits in Register **MCUCR**)

Interrupt 0		Beschreibung
ISC01	ISC00	
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>
#include <stdint.h>

#define LOOPS_PER_MS 50
#define NBLINK 5

static uint16_t muster[8] = {420,680,210,160,370,110,920,750};


/* Funktionsdeklarationen, globale Variablen, etc. */

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

/* Unterbrechungsbehandlungsfunktion */

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

A:

```
/* Funktion main */

   /* Initialisierung und lokale Variablen */

   /* Hauptschleife */

    /* auf Tastendruck warten */
```

M:

```
    /* Betaetigung von Tasten auswerten */
```

```
/* Ende main */
/* Funktion send
   zu Taste gehoerendes Muster mit Infrarot-LED senden */
```

S:

```
/* Initialisierungsfunktion */
```

```
/* Wartefunktion */
```

IW:

Aufgabe 3: Microcontroller-Programmierung (12 Punkte)

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

Gegeben sei folgende ISR auf einem AVR-Microcontroller (SPiCBoard), ausgelöst durch mechanischen Taster (Schließer, active-low, hardwareseitig entprellt):

```
ISR( INT1_vect ) {
   static uint8_t counter = 100;
   sb_7seg_showNumber( --counter ); // Wert anzeigen
   if( counter == 0 )
     counter = 100;
 }
```

a) Was ist der angezeigte Wert nach einem kurzen Tastendruck (Schließen und Öffnen), wenn der Interrupt `INT1` konfiguriert wurde mit:
   (1) Pegelsteuerung,
   (2) Flankensteuerung mit fallender Flanke,
   (3) Flankensteuerung mit steigender und fallender Flanke.
   Geben Sie jeweils eine kurze Begründung an (6 Punkte)

b) Wie ändert sich die Ausgabe in den Fällen (1) - (3), wenn der Taster nicht hardwareseitig entprellt wird (3 Punkte)

c) Wie ändert sich die Ausgabe in den Fällen (1) - (3), wenn man das **`static`** in der Definition von **`counter`** entfernt. Bitte geben Sie eine kurze Begründung an. (3 Punkte)

Aufgabe 4: Programmierfehler und C (4 Punkte)

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

C gilt als sehr fehleranfällig. Nennen Sie vier Beispiele für Programmierfehler, die in C zu undefiniertem Programmverhalten führen (4 Punkte).