

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) Folgende Makrodefinition findet sich in der AVR-Bibliothek:

```
#define PINA (*(volatile uint8_t *)0x39)
```

Welche der folgenden Aussagen bezüglich der Verwendung des **volatile**-Schlüsselworts ist in diesem Fall richtig?

2 Punkte

- Das **volatile**-Schlüsselwort stellt hier sicher, dass der Zugriff auf **PINA** mit Interrupts synchronisiert wird.
- Das **volatile**-Schlüsselwort ermöglicht den sicheren Zugriff auf einzelne Bits des Registers.
- Auf der AVR-Plattform werden Hardware-Register (wie **PINA**) im RAM eingebunden. Das **volatile**-Schlüsselwort zeigt an, dass es sich dabei um flüchtigen Speicher handelt.
- Wird der Port A als Eingang konfiguriert, könnte sich der Wert von **PINA** jederzeit ändern. Durch **volatile** wird der Compiler angewiesen, stets den aktuellen Wert aus **PINA** zu lesen.

b) Welche Aussage zu folgender Funktion ist richtig?

```
int *foo() {
    static int bar = 0;
    bar++;
    return &bar;
}
```

2 Punkte

- Die Funktion liefert einen Zeiger auf die lokale Variable **bar** zurück. Dies ist in C nicht zulässig und führt zu einem Übersetzungsfehler.
- Die Variable **bar** ist über die Laufzeit der **foo()**-Funktion hinaus gültig. Daher kann der zurückgelieferte Zeiger sicher vom Aufrufer verwendet werden.
- Die Variable **bar** enthält beim Verlassen der **foo()**-Funktion immer den Wert 1, da **bar** bei jedem Aufruf von **foo()** mit 0 initialisiert wird.
- Beim Verlassen der Funktion **foo()** wird die automatic-Variablen **bar** vom Stack entfernt und der Zeiger verliert seine Gültigkeit. Ein Zugriff durch den Aufrufer führt zu zufälligen Ergebnissen.

c) Welche der folgenden Aussagen zum Begriff der Rücksprungadresse ist richtig?

2 Punkte

- Bei Aufruf einer Funktion über einen Funktionszeiger muss der Programmierer eine Rücksprungadresse angeben, an der das Programm später fortgesetzt werden soll.
- Die Rücksprungadresse ermöglicht die Rückkehr ins Betriebssystem. Auf einer Mikrocontroller-Plattform ist sie allerdings nicht vorhanden.
- Bei Aufruf einer Funktion sichert der Prozessor selbsttätig die Adresse der folgenden Instruktion. Dies ist die Rücksprungadresse.
- Bei rekursiven Funktionsaufrufen erstellt der Compiler eine Rücksprungadresse um sicher zu stellen, dass die Rekursion terminiert.

d) Gegeben sei folgendes Programmfragment für einen AVR-Microcontroller:

```
uint8_t a = 100;
uint8_t b;
```

```
b = a+a * 2-50;
```

Welche der folgenden Aussagen ist richtig?

2 Punkte

- b** hat nach Ausführung der Zuweisung den Wert 250.
- b** hat nach Ausführung der Zuweisung den Wert 350.
- Während der Ausführung kommt es zu einem Bereichsüberlauf; auf der AVR-Plattform bleibt dieser jedoch unentdeckt.
- Der Compiler warnt zur Übersetzungszeit vor einem Bereichsüberlauf.

e) Gegeben sei folgender Ausschnitt eines C-Programms, welches eine Variable **foo** vom Typ **uint8_t** verwendet:

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

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

2 Punkte

- Das höchwertige 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 höchwertigen Bits von **foo** kann keine Aussage getroffen werden.

f) Gegeben ist folgender Programmcode:

```
uint16_t x[] = {1, 2, 4, 8};
uint16_t *y = &x[1];
y += 2;
```

2 Punkte

Welchen Wert liefert die Dereferenzierung von **y**?

- 2
- 4
- 8
- Zur Laufzeit tritt ein Fehler auf.

Aufgabe 2: Bewässerungssystem (30 Punkte)

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

Schreiben Sie eine Steuerung für einen AVR-Mikrocontroller, die ein Bewässerungssystem für Pflanzen realisiert. Über einen Feuchtigkeitssensor wird der aktuelle Wasserstand periodisch abgefragt und bei Bedarf über eine Pumpe Wasser nachgefüllt. Eine LED-Leiste visualisiert zu jeder Zeit den aktuellen Füllstand.

Im Detail soll Ihr Programm wie folgt funktionieren:

- Initialisieren Sie die Hardware in der Funktion **void init(void)**, so dass die LEDs und die Pumpe aus sind. Treffen Sie keine Annahmen über den initialen Zustand der Hardware-Register.
- Zu Beginn befindet sich das Programm im Überwachungsmodus: Die LED-Leiste zeigt den aktuellen Wert des Feuchtigkeitssensors an, der alle 5 Minuten periodisch abgefragt wird. Dazu steht ein externer Zeitgeber zur Verfügung, der jede Sekunde einen Interrupt durch kurzzeitigen High-Pegel anfordert.
- Während des Wartens auf den Interrupt des Zeitgebers soll der Mikrocontroller jeweils zum Stromsparen in den Schlafmodus gehen.
- Der aktuelle Wert des Feuchtigkeitssensors wird von einem Analog-Digital-Umsetzer (ADC) bereitgestellt, dessen Wert mit der Bibliotheksfunktion **uint16_t adcread(void)** abgefragt werden kann. Die Funktion liefert einen 12-Bit-Wert zurück, wobei der Wert 0 den minimalen Füllstand angibt.
- Die LED-Leiste muss nach jeder Abfrage des Sensors aktualisiert werden. Implementieren Sie hierfür die Funktion **void showlevel(uint16_t value)**, in der der übergebene Wert gleichmäßig auf die acht LEDs abgebildet wird (eine leuchtende LED entspricht dem minimalen, acht LEDs dem maximalen Füllstand).
- Unterschreitet der Sensorwert die Hälfte des maximalen Wertes, muss die Pumpe eingeschaltet werden, bis der maximale Füllstand erreicht ist.
- Implementieren Sie die Überwachung des Pumpvorgangs durch periodisches Abfragen des Sensors alle 100 ms 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 gewartet werden muss, um eine Millisekunde verstreichen zu lassen.

Information über die Hardware

Pumpe: **PORTC**, Pin 3, aktiviert bei High-Pegel

- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRC**-Register auf 1

LEDs: **PORTB**, Pins 0-7, LED 1 an Pin 0, leuchtet bei Low-Pegel

- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRB**-Register auf 1

Zeitgeber:**PORTD**, Interrupt-Leitung an Pin 3

- Pin als Eingang konfigurieren: entsprechendes Bit in **DDRD**-Register auf 0
- externe Interruptquelle **INT1**, ISR-Vektor-Makro: **INT1_vect**.
- Aktivierung der Interruptquelle erfolgt durch Setzen des **INT1**-Bits im Register **GICR**.
- IRQ wird durch kurzen externen High-Pegel signalisiert, der Pullup-Widerstand muss daher abgeschaltet sein (entsprechendes Bit in **PORTD**-Register auf 0 setzen).

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

ISC11	ISC10	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

