

# Hardwarenahe Programmierung

## Übungsaufgaben – 9. November 2023

Diese Übung enthält Punkteangaben wie in einer Klausur. Um zu „bestehen“, müssen Sie innerhalb von 80 Minuten unter Verwendung ausschließlich zugelassener Hilfsmittel 14 Punkte (von insgesamt 29) erreichen.

### Aufgabe 1: Text-Grafik-Bibliothek

Schreiben Sie eine Bibliothek für „Text-Grafik“ mit folgenden Funktionen:

- **void clear (char c)**  
Bildschirm auf Zeichen `c` löschen, also komplett mit diesem Zeichen (z. B.: Leerzeichen) füllen
- **void put\_point (int x, int y, char c)**  
Punkt setzen (z. B. einen Stern (\*)) an die Stelle  $(x, y)$  „malen“
- **char get\_point (int x, int y)**  
Punkt lesen
- **void display (void)**  
das Gezeichnete auf dem Bildschirm ausgeben

(8 Punkte)

Hinweise:

- Eine C-Bibliothek besteht aus (mindestens) einer `.h`-Datei und einer `.c`-Datei.
- Verwenden Sie ein Array als „Bildschirm“. Vor dem Aufruf der Funktion `display()` ist nichts zu sehen; alle Grafikoperationen erfolgen auf dem Array.
- Verwenden Sie Präprozessor-Konstante, z. B. `WIDTH` und `HEIGHT`, um Höhe und Breite des „Bildschirms“ festzulegen, z. B.:  

```
#define WIDTH 72  
#define HEIGHT 24
```
- Schreiben Sie zusätzlich ein Test-Programm, das alle Funktionen der Bibliothek benutzt, um ein hübsches Bild (z. B. ein stilisiertes Gesicht – „Smiley“) auszugeben.

### Aufgabe 2: Mikrocontroller

An die vier Ports eines ATmega16-Mikrocontrollers sind Leuchtdioden angeschlossen:

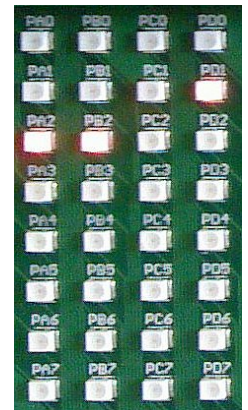
- von links nach rechts an die Ports A, B, C und D,
- von oben nach unten an die Bits Nr. 0 bis 7.

Wir betrachten das folgende Programm ([aufgabe-2.c](#)):

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

```
int main (void)
```

```
{  
    DDRA = 0xff;  
    DDRB = 0xff;  
    DDRC = 0xff;  
    DDRD = 0xff;  
    PORTA = 0x1f;  
    PORTB = 0x10;  
    PORTD = 0x10;  
    PORTC = 0xfc;  
    while (1);  
    return 0;  
}
```



- (a) Was bewirkt dieses Programm? (4 Punkte)
- (b) Wozu dienen die ersten vier Zeilen des Hauptprogramms? (2 Punkte)
- (c) Was würde stattdessen die Zeile `DDRA, DDRB, DDRC, DDRD = 0xff;` bewirken? (2 Punkte)
- (d) Schreiben Sie das Programm so um, daß die durch das Programm dargestellte Figur spiegelverkehrt erscheint. (3 Punkte)
- (e) Wozu dient das `while (1)`? (2 Punkte)
  - Alle Antworten bitte mit Begründung.

### Aufgabe 3: LED-Blinkmuster

Wir betrachten das folgende Programm für einen ATmega32-Mikro-Controller (Datei: [aufgabe-3.c](#)).

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

uint8_t counter = 1;
uint8_t leds = 0;

ISR (TIMER0_COMP_vect)
{
    if (counter == 0)
    {
        leds = (leds + 1) % 8;
        PORTC = leds << 4;
    }
    counter++;
}

void init (void)
{
    cli ();
    TCCR0 = (1 << CS01) | (1 << CS00);
    TIMSK = 1 << OCIE0;
    sei ();
    DDRC = 0x70;
}

int main (void)
{
    init ();
    while (1)
        ; /* do nothing */
    return 0;
}
```

An die Bits Nr. 4, 5 und 6 des Output-Ports C des Mikro-Controllers sind LEDs angeschlossen. Sobald das Programm läuft, blinken diese in charakteristischer Weise:

Phase	LED oben (rot)	LED Mitte (gelb)	LED unten (grün)
1	aus	aus	an
2	aus	an	aus
3	aus	an	an
4	an	aus	aus
5	an	aus	an
6	an	an	aus
7	an	an	an
8	aus	aus	aus

Jede Phase dauert etwas länger als eine halbe Sekunde. Nach 8 Phasen wiederholt sich das Schema.

Erklären Sie das Verhalten des Programms anhand des Quelltextes:

- Wieso macht das Programm überhaupt etwas, wenn doch das Hauptprogramm nach dem Initialisieren lediglich eine Endlosschleife ausführt, in der *nichts* passiert? (1 Punkt)
- Wieso wird die Zeile `PORTC = leds << 4;` überhaupt aufgerufen, wenn dies doch nur unter der Bedingung `counter == 0` passiert, wobei die Variable `counter` auf 1 initialisiert, fortwährend erhöht und nirgendwo zurückgesetzt wird? (2 Punkte)
- Wie kommt das oben beschriebene Blinkmuster zustande? (2 Punkte)
- Wieso dauert eine Phase ungefähr eine halbe Sekunde? (2 Punkte)
- Was bedeutet „`ISR (TIMER0_COMP_vect)`“? (1 Punkt)

Hinweis:

- Die Funktion `init()` sorgt dafür, daß der Timer-Interrupt Nr. 0 des Mikro-Controllers etwa 488mal pro Sekunde aufgerufen wird. Außerdem initialisiert sie die benötigten Bits an Port C als Output-Ports. Sie selbst brauchen die Funktion `init()` nicht weiter zu erklären.

*Viel Erfolg!*