

# Hardwarenahe Programmierung / Angewandte Informatik

## Übungsaufgaben – 20. November 2017

### Aufgabe 1: Zahlensysteme

Wandeln Sie ohne Hilfsmittel

- nach Dezimal:
  - (a)  $0010\ 0000_2$
  - (b)  $42_{16}$
  - (c)  $17_8$
- nach Hexadezimal:
  - (d)  $0010\ 0000_2$
  - (e)  $42_{10}$
  - (f)  $192.168.20.254_{256}$
- nach Binär:
  - (g)  $750_8$
  - (h)  $42_{10}$
  - (i)  $AFFE_{16}$

Berechnen Sie ohne Hilfsmittel:

- (j)  $750_8 \& 666_8$
- (k)  $A380_{16} + B747_{16}$
- (l)  $AFFE_{16} >> 1$

(Die tiefgestellte Zahl steht für die Basis des Zahlensystems.)

### Aufgabe 2: Mikro-Controller

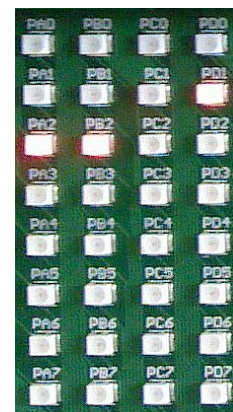
An die vier Ports eines ATmega16-Mikro-Controllers sind Leuchtdioden angeschlossen:

- 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:

```
#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?
- (b) Wozu dienen die ersten vier Zeilen des Hauptprogramms?
- (c) Was würde stattdessen die Zeile `DDRA, DDRB, DDRC, DDRD = 0xff;` bewirken?
- (d) Schreiben Sie das Programm so um, daß die durch das Programm dargestellte Figur spiegelverkehrt erscheint.
- (e) Wozu dient das `while (1);`?

### Aufgabe 3: Länge von Strings

Strings werden in der Programmiersprache C durch Zeiger auf **char**-Variable realisiert.

Beispiel: **char** \*hello\_world = "Hello,\_world!\n"

Die Systembibliothek stellt eine Funktion **strlen()** zur Ermittlung der Länge von Strings zur Verfügung (**#include <string.h>**).

- (a) Auf welche Weise ist die Länge eines Strings gekennzeichnet?
- (b) Wie lang ist die Beispiel-String-Konstante "Hello,\_world!\n", und wieviel Speicherplatz belegt sie?
- (c) Schreiben Sie eine eigene Funktion **int strlen (char \*s)**, die die Länge eines Strings zurückgibt.

Wir betrachten nun die folgenden Funktionen (Datei: [aufgabe-3.c](#)):

```
int fun_1 (char *s)
{
    int x = 0;
    for (int i = 0; i < strlen (s); i++)
        x += s[i];
    return x;
}
```

```
int fun_2 (char *s)
{
    int i = 0, x = 0;
    int len = strlen (s);
    while (i < len)
        x += s[i++];
    return x;
}
```

- (d) Was bewirken die beiden Funktionen?
- (e) Schreiben Sie eine eigene Funktion, die dieselbe Aufgabe erledigt wie **fun\_2()**, nur effizienter.