1.2 Atmega-Programmierung in ASM/Erstes Programm

1.2.1 Vorbereitung

Das fertige Bord soll ausprobiert werden mit einem ersten Programm. Dieses Programm soll die LEDs 6 und 7 einschalten. Man braucht dazu folgende Utensilien:

- Hardware
  - PC
  - Programmiergerät (*programmer*) mit dem Namen mySmartUSBLight und zwei Leitungen (USB und Flachband)
  - Entwicklungsboard mit Namen RN Control 1.4
- Software
  - unser Programmtext, angepasst auf unser Bord
  - PC-Assemblerprogramm, angepasst an unseren Mikrocontroller; ein so genannter *cross assembler* mit dem Namen `avra`
  - PC-Übertragungsprogramm (*programmer*) mit dem Namen `avrdude`

1.2.2 Hardware

Das Programmiergerät wird mit der USB-Schnittstelle an den PC angeschlossen. Die blaue und die gelbe LED am Programmiergerät sollten jetzt leuchten. Wird es auch vom Betriebssystem entdeckt?

Terminal

```
schueler@debian964:~$ lsusb
Bus 005 Device 002: ID 10c4:ea60 Cygnal Integrated Products, Inc.
                       CP210xUART Bridge / myAVR mySmartUSB light
```

Bei Gerät Nr. 005/002 handelt es sich um das Programmiergerät.

Nun wird das Programmiergerät mit der Flachbandleitung an das Board angeschlossen.

Das Board wird wiederum an ein Steckernetzteil (siehe Spezifikation des Boards(angeschlossen. Die Netz-LED sollte leuchten.

Die DIP-Schalter Nr. 7 und 8 müssen auf ON gesetzt sein.

Einmalig muss die Include-Datei m32def.inc vorbereitet werden, in der einige Eigenschaften des verwendeten Controllers hinterlegt sind[1]:

Terminal

```
schueler@debian964:~$ su
Password: geheim
root@debian964:~# cd /usr/share/avra/
root@debian964:~# mv m32def.inc m32def.inc_orig
root@debian964:~# tr "#" ";" < m32def.inc_orig > m32def.inc
root@debian964:~# exit
```

Das erste Programm für die Beispielhardware RN-Control wird mit dem Editor eingegeben (`leds.asm`):

```
1 .include "/usr/share/avra/m32def.inc"
2         ldi r16, 0b11111111 ; alle auf Ausgang
3         out DDRC, r16
4         ldi r16, 0b00111111 ; Nr. 6, 7
```

[1] In der Datei beginnen alle Kommentare mit einer Raute; AVRA sieht dafür ein Semikolon vor; deshalb ändern wir jede Raute in ein Semikolon.

```
5           out PORTC, r16
6   ende:
7           rjmp ende
```

Dieses kurze Programm hat, wie man sieht, nur fünf Befehle für die CPU des Controllers.

Dann kann die Programmdatei `leds.asm` compiliert werden:

Terminal

```
schueler@debian964:~$ avra leds.asm
Copyright (C) 1998-2007. Check out README file for more info
.. bla, bla ..
Assembly complete with no errors.
Segment usage:
   Code      :         5 words (10 bytes)
   Data      :         0 bytes
   EEPROM    :         0 bytes
```

Das Ergebnis (also die fünf Befehle mit je zwei Bytes) liegt nun in binärer Form vor, und zwar in der Datei `leds.obj`.

Terminal

```
schueler@debian964:~$ ls -l leds.*
-rw-r--r-- 1 schueler schueler 179 Sep 20 19:56 leds.asm
-rw-r--r-- 1 schueler schueler 108 Sep 20 19:56 leds.obj
-rw-r--r-- 1 schueler schueler  63 Sep 20 19:56 leds.hex
-rw-r--r-- 1 schueler schueler  13 Sep 20 19:56 leds.eep.hex
-rw-r--r-- 1 schueler schueler   0 Sep 20 19:56 leds.cof
```

Für die Übertragung zum Controller ist das Intel-Hex-Format besser geeignet. Der Assembler war so nett und hat dazu die beiden Dateien `leds.hex` und `leds.eep.hex` gleich mit erzeugt. An der Existenz der Datei `leds.cof` kann man sehen, dass das Compilieren erfolgreich war.

Nun muss das Ergebnis noch zum Board übertragen werden. Auf vielen Systemen braucht man dazu Administrator-Rechte. Dazu muss man einmalig die Rechte für das Übertragungs-Programm `avrdude` verändern.

Terminal

```
schueler@debian964:~$ su
root@debian964:~# chmod 2755 /usr/bin/avrdude
root@debian964:~# exit
```

Nun kann die Übertragung beginnen.

a) mit USBASP

Terminal

```
schueler@debian964:~$ avrdude -p m32 -c usbasp -e -U \
> flash:w:leds.hex
```

b) mit mySmartUSB light

Terminal

```
schueler@debian964:~$ avrdude -p m32 -c stk500v2 \
> -P /dev/ttyUSB0 -e -U flash:w:leds.hex
```

Die angegebenen Optionen haben alle eine spezielle Bedeutung (Tabelle 1). Bei Erfolg meldet das Übertragungsprogramm: `avrdude done. Thank you.` Das Programm `leds` startet sofort; beide LEDs leuchten. Durch Betätigen des RESET-Tasters auf dem Board kann es jederzeit neu gestartet werden.

Option	Bedeutung	Bedeutung hier
`-p m32`	Ziel-Prozessor	m32 (=ATmega32)
`-c stk500v2`	Programmiergerät	STK-500-kompatibel
`-P /dev/ttyUSB0`	Gerätedatei	erste emulierte RS232-Schnittstelle
`-e`	Proz. vorher löschen (erase)	ja
`-U flash:w:leds.hex`	Op. ausführen	Ziel=flash
	ziel:operation:datei[:format]	w=schreiben

Tabelle 1: Benutzte Optionen von AVRDUDE