3.3 Atmega-Programmierung in C/Zugriff auf SRAM, Flash und EEPROM

3.3.1 SRAM

Ein programmierbares Lauflicht wird gebraucht. Kann man einfach ein Array verwenden? Muss man angeben, dass es im SRAM liegen soll?

Das stellt in C kein Problem dar. Alle Variablen werden automatisch im SRAM angelegt. Es gibt dort zwei Bereiche:

a) Das Segment `.bss` enthält alle normalen Variablen.

b) Das Segment `.data` enthält die Variablen, die initialisiert werden, das heißt, die Inhalte werden beim Beginn des Blocks (geschweifte Klammer auf) aus dem Flash-EPROM in die Variablen kopiert.

3.3.2 Flash-EPROM

Nun brauch das Lauflicht im SRAM zu viel Platz. Wie kann man es komplett im Flash-EPROM halten?

Zunächst muss man sehen, welche Bedingungen für Variablen gelten sollten, die sinnvoll im Flash-EPROM gehalten werden können:

a) Sie müssen konstant sein (also eigentlich Konstanten statt Variablen).

b) Sie müssen eine Lebensdauer haben, die der Programmdauer entspricht. Da gibt es zwei Gruppen:

   1) Globale Variablen (außerhalb von Funktionen)
   2) Statische Variablen (innerhalb von Funktionen); sie werden nicht bei jedem Funktionsaufruf neu auf dem Stack angelegt, sondern sind für jede Funktion nur einmal vorhanden.

Für diese Variablen kann über die Headerdatei `<avr/pgmspace.h>` die Speicherklasse `PROGMEM` ermöglicht werden. Mit dieser Speicherklasse wird die Variable im Flash-EPROM abgelegt:

```
1 const unsigned char PROGMEM x=7;
2 int beispiel(void)
3 {
4    static const int PROGMEN y=3;
5    /* ... */
6 }
```

Nun braucht man noch Funktionen, die es erlauben, diese Variablen, die ja woanders liegen als üblich, zu benutzen. Auch sie sind in `<avr/pgmspace.h>` beschrieben:

```
1    y=pgm_read_byte(&x); /* ohne klappt es nicht! */
```

Ebenso gibt es in `<avr/pgmspace.h>` eigene Stringfunktionen für diese Variablen. Achtung: Verwendet man diese Funktionen nicht, wird entgegen der Vorgabe trotzdem Speicherplatz im SRAM benutzt!

3.3.3 EEPROM

Nun wird ein Zähler gebraucht, der festhält, wie oft das System neu gestartet wurde. Auch im EEPROM kann man Variablen ablegen. Die Bedingung hier ist nur:

a) Sie müssen eine Lebensdauer haben, die der Programmdauer entspricht.

1) Globale Variablen (außerhalb von Funktionen)
2) Statische Variablen (innerhalb von Funktionen)

Hier heißt die Speicherklasse `EEPROM` und wird durch die Headerdatei `<avr/eeprom.h>` unterstützt:

```
#include <avr/eeprom.h>
unsigned char EEMEM ch=0x55;
```

Und es gibt wiederum Funktionen, die für EEPROM-Variablen vorgesehen sind:

```
    unsigned char x;
    x=eeprom_read_byte(&ch);
    x=~x;
    eeprom_write_byte(&ch, x);
```

3.3.4 Übertragen der EEPROM-Daten

Nun sollen die EEPROM-Daten nicht nur in der Datei `a.out` landen, sondern von dort auch im Mikrocontroller. Dazu muss man sie zuerst in eine eigene Hex-Datei mit der Endung `.eep` bringen:

Terminal

```
root@debian964:~# avr-objcopy -j .eeprom --change-section-lma \
> .eeprom=0 -O ihex a.out a.eep
```

Anschließend muss diese eigene Hex-Datei auch noch übertragen werden:

Terminal

```
root@debian964:~# avrdude -p m32 -c usbasp -e -U flash:w:a.hex:i \
> -U eeprom:w:a.eep:i
```