# Direkte Programmierung mit Programmer

Mikrocontroller (u.a. auch die auf vielen Baugruppen verwendeten Atmel-Prozessoren) müssen vor der Benutzung mit einem Programmcode (Firmware) versehen werden. Es gibt verschiedene Methoden, eine Firmware in einen Mikrocontroller hineinzubringen, nachfolgend ist der Weg über die ISP-Schnittstelle beschrieben. Benötigt wird hierzu ein Gerät, welches diese Schnittstelle bedienen kann. Am unkompliziertesten für Atmel-Prozessoren hat sich der AVRISP MKII erwiesen.

## Anschluß Programmiergerät

Um eine Firmware in einen AVR-Mikroprozessor zu laden, der bereits auf einer Platine verbaut ist, benutzt man die ISP-Schnittstelle. Das ist eine 6-polige Schnittstelle, die meistens in Form einer 2×3-poligen Stiftleiste auf der Platine vorhanden ist. Diese ISP-Schnittstelle verbindet man mit einem Programmiergerät. Das Programmiergerät hat außerdem einen USB-Anschluß, den man mit dem PC verbindet. Als letztes sollte man die Platine mit dem zu programmierenden Mikroprozessor mit Spannung versorgen.

## Programmiersoftware

Nun braucht man noch Software, um die Firmware vom PC in den Mikroprozessor zu laden. Von Atmel gibt es dazu das Atmel Studio. Das ist ein umfangreiches Entwicklungswerkzeug mit grafischer Benutzeroberfläche. Andere bevorzugen jedoch avrdude, dass es für Linux und Windows gibt.

### Programmierung mit avrstudio

Hier fehlt noch Text.

### Programmierung mit avrdude

Meine Beispiele unten beziehen sich auf Linux. Für Windows läßt man einfach „sudo“ am Anfang der Zeile weg und ersetzt den Namen des USB-Ports (Option -P) durch den entsprechenden COM-Port.

Für Linux-Benutzer muß ich wahrscheinlich nichts weiter erklären. Die Windows-Benutzer müssen nun ein cmd.exe starten und dort den Befehl (ohne sudo) eingeben. Die Firmwaredateien werden im aktuellen Verzeichnis erwartet, ansonsten muß man den vollständigen Pfad zu den Firmwaredateien

angeben. Tipp für Windowsanwender: Wenn man auf einen Ordner mit der rechten Maus klickt und dabei die Shifttaste gedrückt hält, erscheint im Kontextmenu die Auswahl 'Eingabeaufforderung im aktuellen Verzeichnis öffnen', das spart die lästige Pfadeingabe.

Achtung: Auch wenn hier auf der Wikiseite der Befehl in mehreren Zeilen dargestellt wird, muß er immer in einer Zeile geschrieben werden!

#### GBM16T

```
sudo avrdude -v -P usb -c avrisp2 -p x128a1 -B10 -e -U flash:w:gbm16t.hex -U
eeprom:w:gbm16t.eep
```

#### GBMBoost

```
sudo avrdude -v -P usb -c avrisp2 -p x128a1 -B10 -e -U
flash:w:gbmboost_master.hex -U eeprom:w:gbmboost_master.eep
```

#### LightControl

```
sudo avrdude -v -P usb -c avrisp2 -p x128a1 -B10 -e -U
flash:w:LightControl.hex -U eeprom:w:LightControl.eep
```

#### MoBaLiSt

```
sudo avrdude -v -P usb -c avrisp2 -p m32 -B10 -e -U flash:w:mobalist.hex -U
eeprom:w:mobalist.eep
```

#### OpenDCC Z1 mit Xpressnet

##### udev-Regel

```
KERNEL==„ttyUSB*“, ATTRS{product}==„USB-IF OpenDCC V1.2“,
SYMLINK+=„opendcc_z1“
```

```
avrdude -v -c avr911 -p m644p -P /dev/opendcc_z1 -U flash:w:OpenDCC_XP.hex -U
eeprom:w:OpenDCC_XP.eep -b 19200
```

#### OpenDCC-Dekoder Version 1

```
sudo avrdude -v -P usb -c dragon_isp -p t2313 -e -B10 -U lfuse:w:0xee:m -U
hfuse:w:0xd9:m -U efuse:w:0xff:m -U flash:w:OpenDecoder.hex -U
eeprom:w:OpenDecoder.eep
```