Bootloader FastBoot von Peter Dannegger

Anleitung von Karsten Donat – www.KarstenDonat.de/AVR
Stand: 05.08.2007

1 Was ist ein Bootloader?

- Der Bootloader ist selbst ein kleines Programm. Es wird beim Start des Controllers zuerst ausgeführt. Damit das PC-Programm für das Firmware-Update sich melden kann, wartet der Bootloader eine gewisse Zeit (hier 0,33 Sekunden) auf ein Zeichen über die serielle Schnittstelle (UART → RS232, USB). Kommt dies Zeichen wird die neue Firmware gebrannt. Andernfalls wird das eigentliche Programm des Controllers ausgeführt.
- Dem eigentlichen Anwendungsprogramm geht natürlich der Platz den der Bootloader benötigt verloren. Das Peters Bootloader jedoch in ein 512Bytes (256 Worte!) großes Segment passt, stört das nicht weiter.
- Normalerweise wird der Programmcode des Microcontrollers mit einem ISP-Dongle in den Flash gebrannt. Aus verschiedenen Gründen kann dies jedoch nicht möglich/ gewünscht sein:

1. Speed: Der ISP-Dongle kann langsam sein (z.B. myAVR mit aktuellem AVRDude)
2. PINs: Man braucht die PINs und will ISP abschalten (→ Fuses, aber Vorsicht!, danach kann man nur mit dem Bootloader oder einem STK-500 noch an den Flash)
3. Komfort: Man möchte dem Kunden/ Nutzer die Möglichkeit geben, eine neue Firmware selbst einzuspielen. In der Regel hat dieser jedoch keinen ISP-Dongle zur Hand. Eine Rs232/ USB ist aber oftmals vorhanden.
4. Sicherheit: Man möchte dem Kunden nicht die Firmware in deassemblierbarer Form geben (über geänderten Bootloader kann die Datei verschlüsselt sein)

2 Download/ Forum

http://www.mikrocontroller.net/topic/73196

3 Bootloader anpassen

3.1 CPU-Frequenz, Wartezeit - FASTLOAD.H

- bei XTAL die benutzte Frequenz des Controllers einstellen (jungfräuliche AVRs haben oft intern 1MHz aktiviert!)
- im Standard Makefile von WinAVR steht die Frequenz unter F_CPU

```
.equ    XTAL            = 3686400       ; 8MHz, not critical
```

- um die Wartezeit auf das Firmware-Update beim Booten anzupassen:

```
.equ    BootDelay       = XTAL / 3      ; 0.33s
```

3.2 UART Port/ Pins – M*.ASM

- Da die verwendeten Pins für die Schnittstelle freiwählbar sind, müssen diese noch eingestellt werden. Für einige Controller liegen bereits Definitionsdateien bei. Für den ATMega 8 heißt die Datei beispielsweise M8.ASM.
- STX_PORT
  → hier den Sende Port angeben. Für den Hardware-UART des M8 wäre das Port D.
- Es muss aber nicht der Hardware-UART sein! (wenn man den aber eh als Schnittstelle nutzt ist es sinnvoll)

```
.equ    STX_PORT        = PORTD
.equ    STX_DDR         = DDRD
.equ    STX             = PD1

.equ    SRX_PIN         = PIND
.equ    SRX_PORT        = PORTD
.equ    SRX             = PD0
```

Beispiel für Hardware-UART im ATMega 8, 48, 168

- Wenn man einen neuen Controller hinzufügen möchte, muss noch die BufferSize angepasst werden.

4 Compilieren des Booloaders

- Da der Booloader in Assembler geschrieben ist, kann der WinAVR nicht so ohneweiteres damit umgehen.
- Das einfachste ist, sich das AVRStudio von Atmel herunterzuladen http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725
- Im Unterverzeichnis AvrAssembler2 befindet sich der benötigte Assembler. Das Einfachste ist, sich die unter Appnotes benötigte Include Datei des jeweiligen Controllers (oder das komplette Verzeichnis) und die avrasm2.exe ins Verzeichnis des Bootloaders zu kopieren.
- Danach wird der Assembler aufgerufen (m8.asm für ATMega 8)

```
avrasm2 -fI m8.asm
```

Bootloader compilieren

5 Brennen des Bootloaders

- Der nun erzeugte Bootloader wird mit dem vorhanden ISP Dongle in den AVR gebrannt (Intel-Hex-Format)

```
"avrdude" -p m8 -c avr910 -P com1 -U flash:w:"C:\Developing\AVR\Test\Bootloader Peter
Dannegger\fastload_V14\m8_3686400.hex":i -U flash:v:"C:\Developing\AVR\Test\Bootloader
Peter Dannegger\fastload_V14\m8_3686400.hex":i -y
```

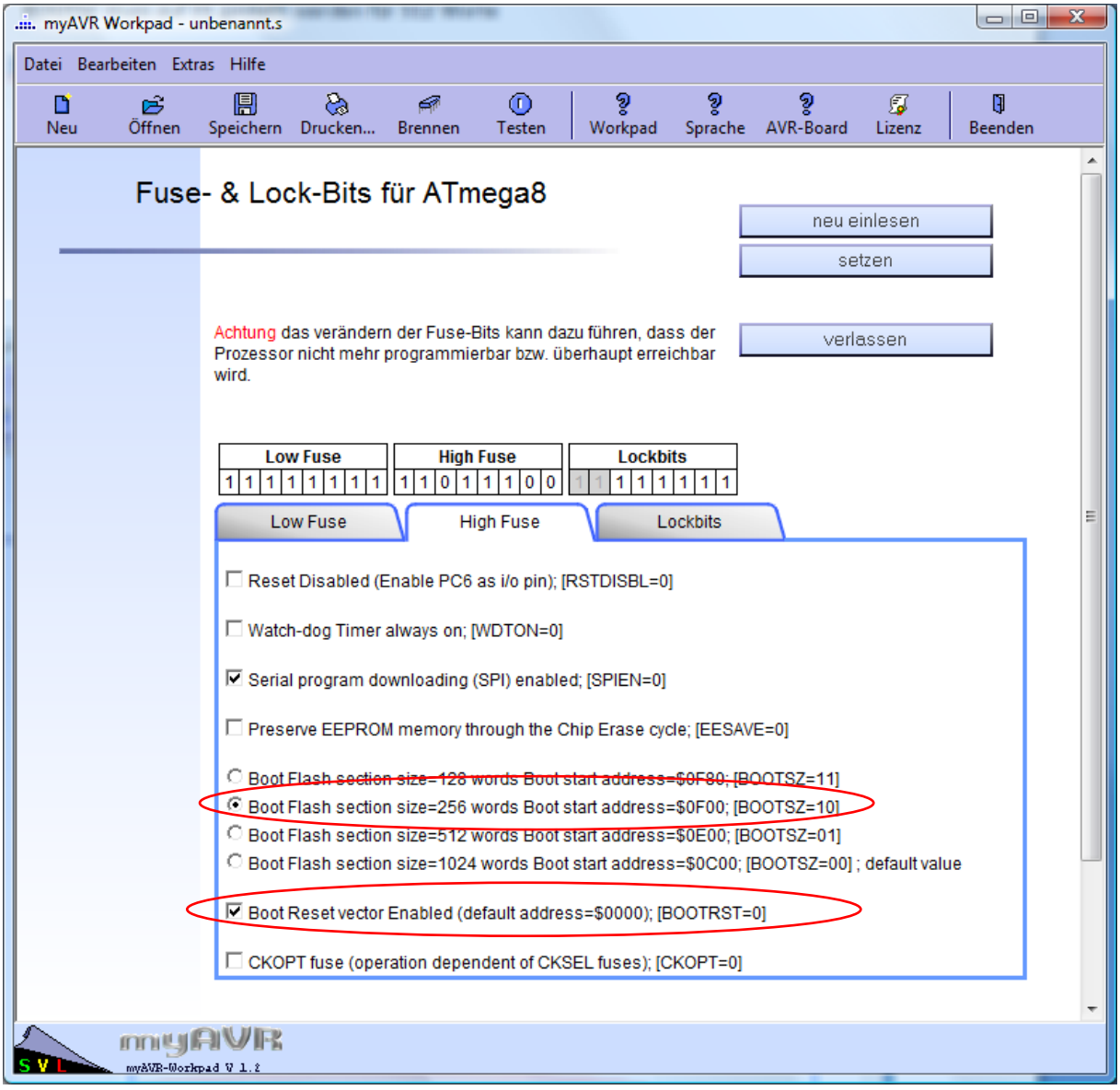
Brennen mit AVRDude auf COM 1 und einem AVR910 kompatiblen Dongle (z.B. myAVR USB)

6 Einstellen der Fuses

- Damit der Bootloader zu Beginn gestartet wird, müssen die entsprechenden Fuses gesetzt werden. Möglich ist dies mit u.a. AVRDude GUI.
- BOOTRST muss auf 0 gestellt werden (aktiviert den Bootloader)
- BOOTSZ muss auf 10 gestellt werden für 256 Worte

6.1 Fuses mit myAVR ändern

- Der myAVR Dongle unterstützt leider keine Fuses mit AVRDude. Abhilfe liefert hier jedoch das myAVR Worpad Plus. Die Demo-Version (reicht hierfür) kann unter http://www.myavr.de/download/myAVR_WorkpadPLUS_Demo.exe heruntergeladen werden.
- Nach dem Installieren und starten im Menü Extras – Fuse- und Lock-Bits wählen.
- Das Programm ermittelt automatisch den aktuellen Status. Je nach Controller steht der Bootloader Support unter High oder Extended Fuses.



7 Brennen des eigentlichen Programmes

- Damit das Programm mit Peters Tool in den Controller geladen werden kann, muss im Intel-HEX-Format vorliegen.
- Im makefile Template von WinAVR (und auch der ATMegaIDE) ist dies standardmäßig eingestellt.

```
# Output format. (can be srec, ihex, binary)
FORMAT = ihex
```

- Danach wird Peters Firmware-Update Tool aufgerufen (hier COM 2)

```
fboot /C2 /Pmain.hex
```

hier wird die main.hex als Hauptprogramm gebrannt

- Wenn noch keine Firmware im Controller ist (direkt nach Installieren des Bootloaders) startet der Brennvorgang automatisch. Andernfalls wartet das Programm auf den Reset des Controllers.
- Peters Tool arbeitet nur mit COM1 bis COM4 zusammen. Also ggf. den USB Adapter im Gerätemanager umstellen.

8 Booloader-Support in ATMegaIDE 2007

- Die IDE kann wahlweise mit AVRDude und einem entsprechenden ISP-Dongle oder mit dem Bootloader das Programm in den Flash brennen.

8.1 Anpassen des Programms

- Um das Firmware Update komfortabler zu gestalten kann ein Software-Reset eingebaut werden, der per RS232/ USB Schnittstelle oder Tastendruck ausgelöst wird.

8.1.1 Tastendruck

- In der mitgelieferten Standard.h gibt es den Befehl void Rest();. Er löst mit Hilfe des Watchdog-Timers einen Hardwarereset aus.
- Man kann jetzt z.B. auf einen bestimmten Tastendruck (-kombination) hin diesen Reset ausführen.

8.1.2 RS232/ USB

- Alternativ kann ein entsprechender Befehl über die RS232 gesendet werden. Standardmäßig ist dies 0xFF ‚R'.

8.2 Einstellungen in der IDE

- Der Bootloader Support kann bei den Projekt-Eigenschaften eingestellt werden. Beim Brennen wird er dann automatisch benutzt.

9 ToDo

9.1 Support großer ATMegas

- Peter und andere arbeiten momentan noch an der Anpassung für größere Controller wie den ATMega 128.

9.2 Komfortables Stand-Alone Firmware-Update

- Basierend auf dem Code der IDE wird es im September noch ein Standalone Programm für das Firmware-Update geben. Es wird auch ein entsprechendes Protokoll zur Versionskontrollle haben. (bisherige Soft- und viel wichtiger Hardwareversion wird gechecked). Eine Verschlüsselung der Firmware-Datei ist in Planung.

9.3 Automatisches erzeugen des Booloaders

- Sobald ich Zeit habe kommt in die IDE auch die Möglichkeit, die Anpassungen des Bootloaders von der IDE durchführen zu lassen.