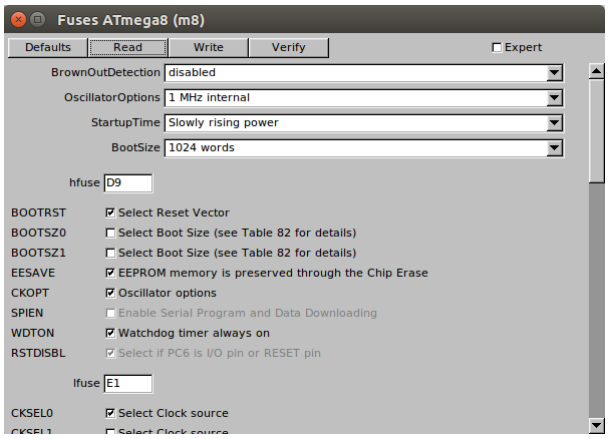
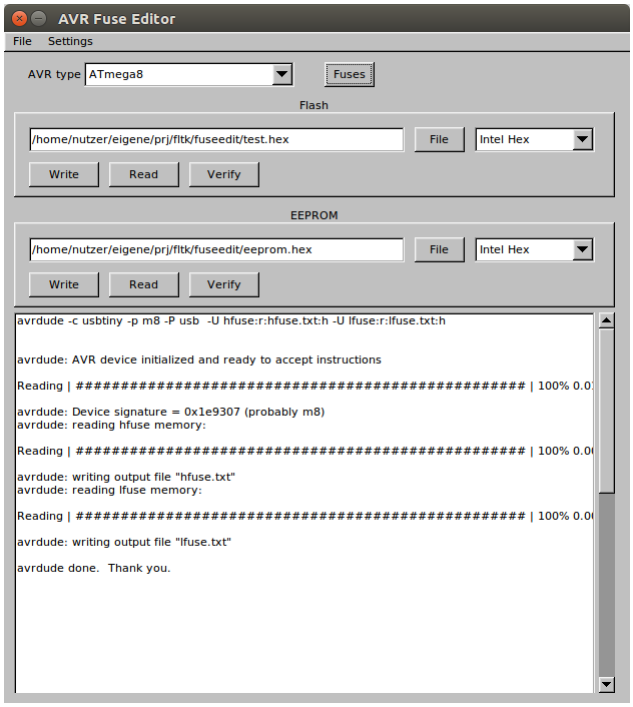
Fuseeditor und grafische Benutzeroberfläche für AVRDUDE





Aus dem
https://www.mikrocontroller.net/topic/402931#new

Einleitung

Bisher hab ich zum Einstellen der Fuses, gern den AVR-Burn-O-Mat genutzt:

http://avr8-burn-o-mat.aaabbb.de/avr8_burn_o_mat_a...

https://www.mikrocontroller.net/articles/Burn-o-mat

Dieses Projekt wird allerdings anscheinend vom Entwickler nicht mehr weiter verfolgt, hier im Forum wurde bereits von Teilnehmern die Konfig weiter gepflegt, allerdings funktioniert die Abbildung der Funktionen zu den Fuses nur mit hart kodierten Modellen.

Der hier vorgestellte Editor bezieht seine Informationen über die Funktion der einzelnen Bits ausschließlich über eine, im XML-Format vorliegende Konfigurationsdatei, die auf dem Format des AVR-BURN-O-MAT aufbaut. Damit ist das Zufügen neuer Controllertypen, später auch ohne Änderung des Programms möglich.

Diese Konfigdatei muss sich beim Programmaufruf, im aktuellen Verzeichnis befinden. Im selben Verzeichnis wird auch eine Datei mit aktuellen Einstellungen gespeichert, es sollte daher beschreibbar sein. Das Programm muss sich nicht zwangsweise dort befinden, sondern kann auch in einem der üblichen Programmverzeichnisse (/usr/bin, $HOME/bin oder c:\programs\fuseedit) abgelegt sein.

Installation

Windows

Für Microsoft Windows kann der Programmordner aus dem aktuellen „win32bin“-Archiv an einen Ort nach Wahl entpackt werden. In diesem Archiv befindet sich die Programmdatei „fuseedit.exe“, die zum Compiler „MinGW“ gehörenden Bibliotheken, sowie die Konfigurationsdatei „config.xml“

Wenn Schreibrechte für diesen Ordner existieren, kann das Programm von hier gestartet werden.

Linux

Wegen der Vielzahl an unterschiedlichen Distributionen, erscheint es sinnvoll, das Programm auf dem System zu compilieren, auf dem es genutzt werden soll. Dafür ist der gcc (auf vielen Systemen bereits installiert), sowie das FLTK-Framework erforderlich.

Dieses kann auf Debian/Ubuntu-Systemen mit

```
sudo apt-get install libfltk1.3-dev
```

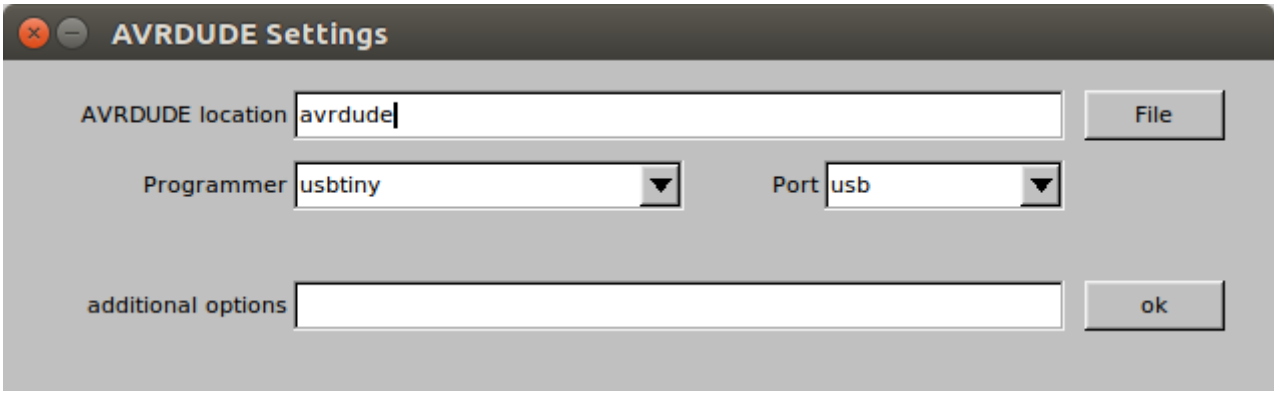
installiert werden. Die Quelltexte befinden sich im aktuellen „src...“-Archiv, welches in ein Verzeichnis nach Wahl entpackt werden sollte. Zum Compilieren des Programms, befindet sich im „src“-Ordner ein Shellscript mit dem Namen „c“, welches die erforderlichen Schritte durchführt:

War die Übersetzung erfolgreich, wird das Programm zum Test auch gleich gestartet (wegen meiner Faulheit bei der Programmentwicklung). Bei Bedarf, kann das Programm „fuseedit“ jetzt in eins der üblichen „bin“-verzeichnisse geschoben werden.

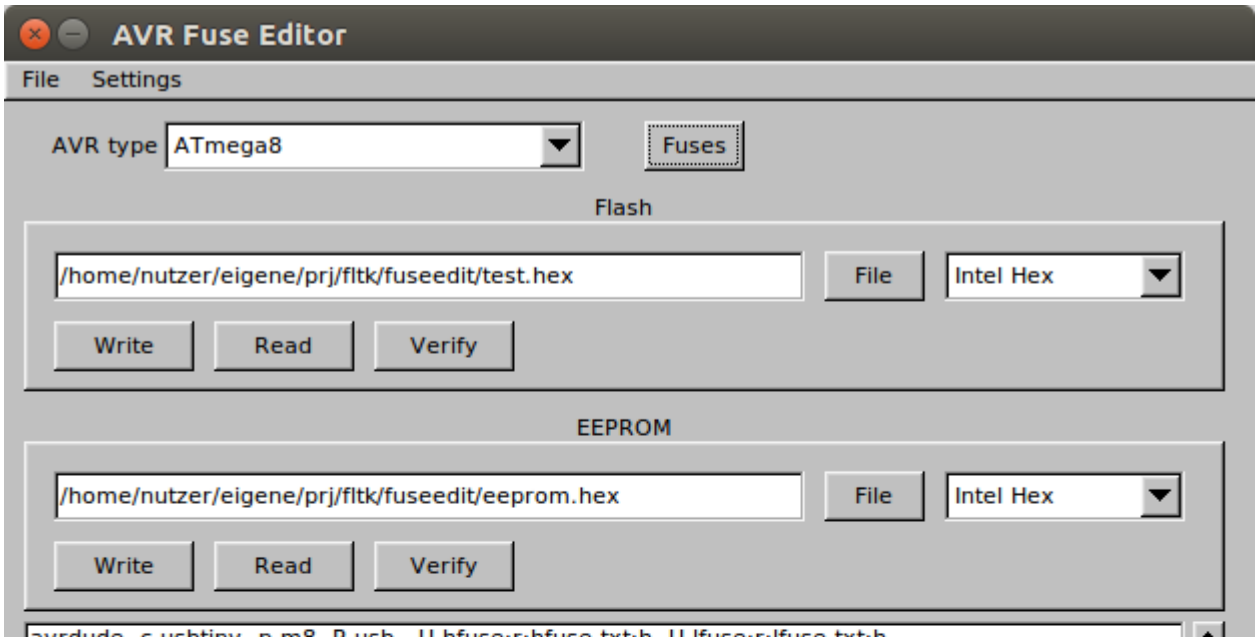
Benutzung

Die Benutzeroberfläche wurde dem Programm AVR-Burn-O-Mat nachempfunden.

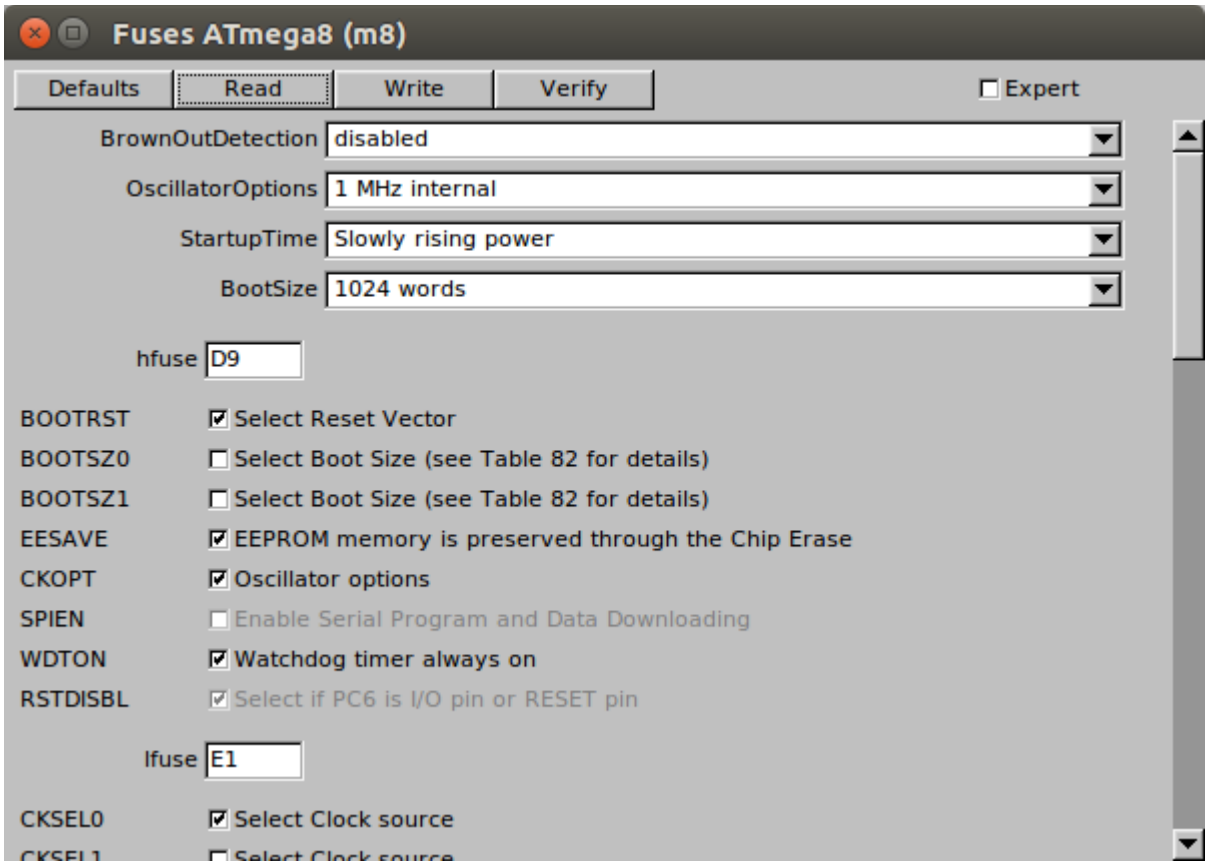
Als Erstes sollte über Settings→AVRDUDE, überprüft werden, ob das Programm „avrdude“ gefunden wird. Sollte es im Suchpfad stehen, genügt der Programmname, wie hier dargestellt:



Ansonsten muss der komplette Pfad eingetragen werden (Dateiauswahldialog mit Schaltfläche „File“). Wenn dies erfolgreich ist, dann ist das Dropdownfeld mit den zur Verfügung stehenden Programmern gefüllt (avrdude wird intern mit „-c ?“ aufgerufen, um die verfügbare Programmerliste zu erhalten). Für den Port, ist das Dropdownfeld mit einigen gebräuchlichen Namen gefüllt. Sollte ein abweichender benötigt werden, kann er auch mit der Hand eingetragen werden. Unter „additional options“, können zusätzliche Einstellungen an den avrdude weitergegeben werden, bei mir z.B. oft „-B 20“, um die SPI-Geschwindigkeit bei langsamem Ziel zu drosseln.



Über die Schaltfläche „Fuses“, kommt man jetzt in den Fuseeditor:



Hier sind für zusammengehörige Funktionsgruppen Dropdownfelder angelegt. Zusätzlich sind die Fusebytes im Hexadezimalformat, die Bits einzeln über Checkboxen zu erreichen. Bei den Bits bedeutet, im Unterschied zum AVR-Burn-O-Mat, eine gesetzte Checkbox, das dieses Bit auf 1 steht!

Die in der XML-Konfig mit „Expert“ gekennzeichneten Fuses sind in der Linuxversion erstmal ausgegraut, können aber durch Aktivieren der Checkbox „Expert“ zugänglich gemacht werden. Für die Hex-Eingabe gilt diese „Kindersicherung“ nicht, hier ist also Umsicht geboten.

Unter Windows ist mir das Deaktivieren der Checkboxen mit FLTK 1.3 nicht gelungen.

Aufbau der Konfigurationsdatei

Die Datei „config.xml“ enthält für jeden Controller die Definition der einzelnen Fusebits, sowie die Belegung selbiger in den einzelnen Funktionsgruppen, hier am Beispiel für den Mega8:

```
<AVR name="m8" caption="ATmega8">
 <Fuse name="RSTDISBL" bit="7" fuseByte="hfuse" default="1" desc="Select if PC6 is I/O pin or RESET pin" mode="expert"/>
 <Fuse name="WDTON" bit="6" fuseByte="hfuse" default="1" desc="Watchdog timer always on"/>
 <Fuse name="SPIEN" bit="5" fuseByte="hfuse" default="0" desc="Enable Serial Program and Data Downloading" mode="expert"/>
 <Fuse name="CKOPT" bit="4" fuseByte="hfuse" default="1" desc="Oscillator options"/>
 <Fuse name="EESAVE" bit="3" fuseByte="hfuse" default="1" desc="EEPROM memory is preserved through the Chip Erase"/>
 <Fuse name="BOOTSZ1" bit="2" fuseByte="hfuse" default="0" desc="Select Boot Size (see Table 82 for details)"/>
 <Fuse name="BOOTSZ0" bit="1" fuseByte="hfuse" default="0" desc="Select Boot Size (see Table 82 for details)"/>
 <Fuse name="BOOTRST" bit="0" fuseByte="hfuse" default="1" desc="Select Reset Vector"/>
 <Fuse name="BODLEVEL" bit="7" fuseByte="lfuse" default="1" desc="Brown out detector trigger level"/>
 <Fuse name="BODEN" bit="6" fuseByte="lfuse" default="1" desc="Brown out detector enable"/>
 <Fuse name="SUT1" bit="5" fuseByte="lfuse" default="1" desc="Select start-up time"/>
 <Fuse name="SUT0" bit="4" fuseByte="lfuse" default="0" desc="Select start-up time"/>
 <Fuse name="CKSEL3" bit="3" fuseByte="lfuse" default="0" desc="Select Clock source"/>
 <Fuse name="CKSEL2" bit="2" fuseByte="lfuse" default="0" desc="Select Clock source"/>
 <Fuse name="CKSEL1" bit="1" fuseByte="lfuse" default="0" desc="Select Clock source"/>
 <Fuse name="CKSEL0" bit="0" fuseByte="lfuse" default="1" desc="Select Clock source"/>
  <BrownOutDetection>
    <Setting caption="disabled"           BODEN="1"/>
    <Setting caption="2.6V (2.4V - 2.9V)" BODEN="0" BODLEVEL="1"/>
    <Setting caption="4.0V (3.7V - 4.5V)" BODEN="0" BODLEVEL="0"/>
  </BrownOutDetection>
  <OscillatorOptions>
    <Setting caption="external clock!"      CKSEL3="0" CKSEL2="0" CKSEL1="0" CKSEL0="0"/>
    <Setting caption="1 MHz internal"       CKSEL3="0" CKSEL2="0" CKSEL1="0" CKSEL0="1"/>
    <Setting caption="2 MHz internal"       CKSEL3="0" CKSEL2="0" CKSEL1="1" CKSEL0="0"/>
    <Setting caption="4 MHz internal"       CKSEL3="0" CKSEL2="0" CKSEL1="1" CKSEL0="1"/>
    <Setting caption="8 MHz internal"       CKSEL3="0" CKSEL2="1" CKSEL1="0" CKSEL0="0"/>
    <Setting caption="ext RC 0.1-0.9 MHz"   CKSEL3="0" CKSEL2="1" CKSEL1="0" CKSEL0="1"/>
    <Setting caption="ext RC 0.9-3.0 MHz"   CKSEL3="0" CKSEL2="1" CKSEL1="1" CKSEL0="0"/>
    <Setting caption="ext RC 3.0-8.0 MHz"   CKSEL3="0" CKSEL2="1" CKSEL1="1" CKSEL0="1"/>
    <Setting caption="ext RC 8.0-12.0 MHz"  CKSEL3="1" CKSEL2="0" CKSEL1="0" CKSEL0="0"/>
    <Setting caption="ext LF Crystal"       CKSEL3="1" CKSEL2="0" CKSEL1="0" CKSEL0="1"/>
    <Setting caption="Resonator 0.4-0.9MHz" CKSEL3="1" CKSEL2="0" CKSEL1="1" CKOPT="1"/>
    <Setting caption="Crystal 0.9-3.0MHz"   CKSEL3="1" CKSEL2="1" CKSEL1="0" CKOPT="1"/>
    <Setting caption="Crystal 3.0-8.0MHz"   CKSEL3="1" CKSEL2="1" CKSEL1="1" CKOPT="1"/>
    <Setting caption="Crystal more than1MHz" CKSEL3="1" CKSEL2="1" CKSEL1="1" CKOPT="0"/>
  </OscillatorOptions>
  <StartupTime>
             <Setting caption="BOD enabled (fast)" SUT1="0" SUT0="0"/>
             <Setting caption="Fast rising power " SUT1="0" SUT0="1"/>
             <Setting caption="Slowly rising power" SUT1="1" SUT0="0"/>
             <Setting caption="SUT reserved"       SUT1="1" SUT0="1"/>
  </StartupTime>
  <BootSize>
             <Setting caption="1024 words" BOOTSZ1="0" BOOTSZ0="0"/>
             <Setting caption="512 words"  BOOTSZ1="0" BOOTSZ0="1"/>
             <Setting caption="256 words"  BOOTSZ1="1" BOOTSZ0="0"/>
             <Setting caption="128 words"  BOOTSZ1="1" BOOTSZ0="1"/>
  </BootSize>
</AVR>
```

Der Aufbau dieser Datei sollte weitestgehend selbsterklärend sein. Korrekturen/Erweiterungen sind mit einem geeigneten Texteditor möglich.

Speichern der Einstellungen

Wird das Programm beendet, wird im aktuellen Verzeichnis, eine Datei „config.txt“ gespeichert, mit ähnlichem Inhalt:

```
flashfile=/home/nutzer/eigene/prj/fltk/fuseedit/test.hex
eepromfile=/home/nutzer/eigene/prj/fltk/fuseedit/eeprom.hex
dude_location=avrdude
dude_options=
programmer=usbtiny
dude_port=usb
device_type=ATmega8
```

Diese Einstellungen werden beim nächsten Programmstart wiederhergestellt. Die gleichen Einträge können auch als Kommandozeilenargumente übergeben werden, haben dann Vorrang vor den Einstellungen in der „config.txt“.

Danksagungen

Für die geleistete Vorarbeit, bedanke ich mich bei Torsten Brischalle, für den AVR-Burn-O-Mat, das AVRDUDE Team, sowie die Teilnehmer vom mikrocontroller.net für die Pflege der Konfigdatei.