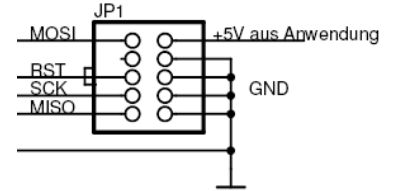


Flashen von Atmel-Prozessoren mittels „In System Programming“ (ISP).

Nick, DF1FO, stellt den jeweils letztaktuellen [Assembler-Sourcecode](#) für seine exzellenten Entwicklungen für ARDF auf seiner homepage www.mydarc.de/df1fo als Assemblerfile (*.asm) zur Verfügung.

Um diese Software oder deren Aktualisierung in die jeweiligen Atmel Prozessoren zu flashen hat Nick üblicherweise die 10 polige Standard-Pfostenleiste für ISP-Programmierung auf seinen Print's vorgesehen. Ausgenommen bei Atiny12 Geräten, bei denen auch die Programmierpins für die Steuerung zusätzlich verwendet werden müssen.



Teil 1: Wie kommt die Software in den Prozessor?

Zuerst ist das der Assembler-File (*.asm) in ein Hex-File (*.hex) zu wandeln. Diese Wandlung wird im Teil 2 beschrieben.

Dieses Hex-File ist mittels eines geeigneten

- Programmes (PonyProg2000, Freeware) und eines geeignetes
- Programmieradapters (realisierbar mit wenigen Bauelementen)

in den Prozessor zu schreiben.

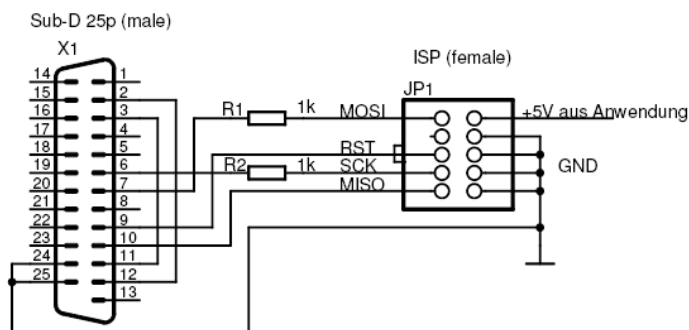
PonyProg2000:

PonyProg 2000 wird von <http://www.lancos.com/ppwin95.html> heruntergeladen und auf dem PC installiert.

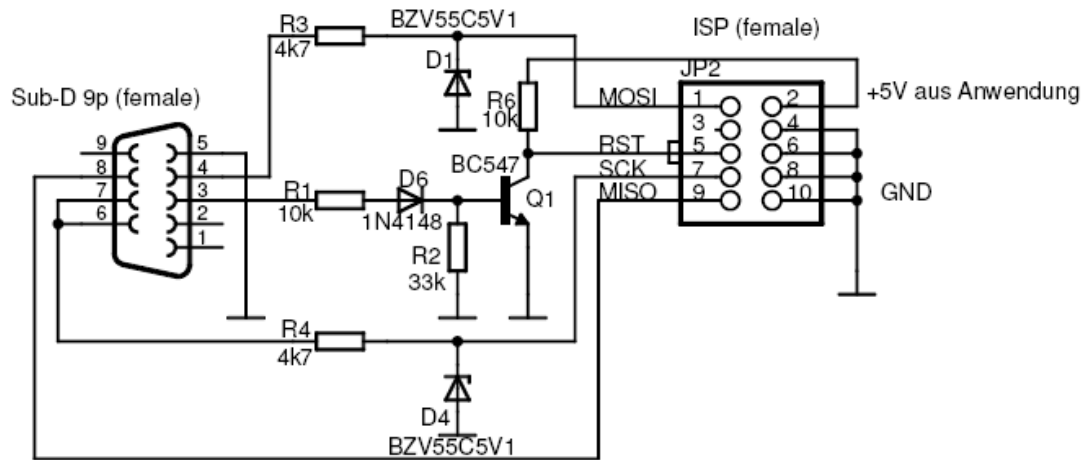
Programmier-IF für PonyProg2000:

Parallelport-IF in der einfachsten Lösung, die beiden Widerstände werden im Gehäuse untergebracht, die 10 polige ISP-Buchse über ein kurzes Kabelstück abgesetzt.

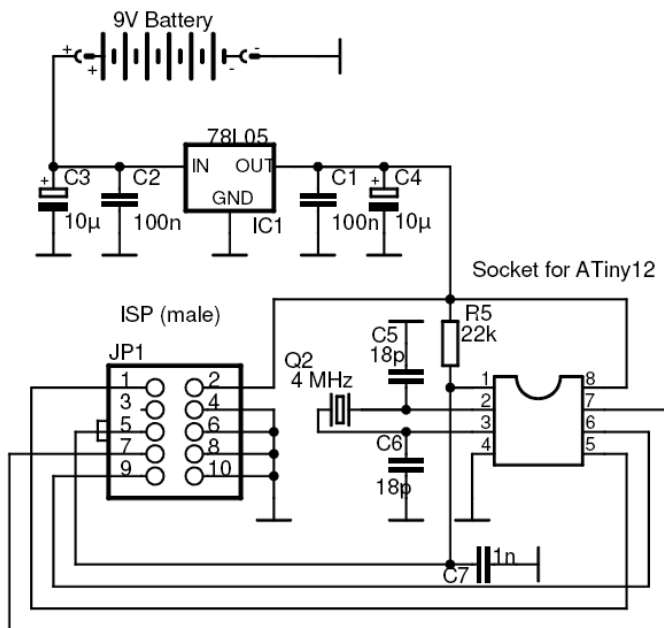
In der Literatur wird allerdings darauf hingewiesen, dass die Gefahr der Beschädigung des PC-Druckerports bei falscher Anschaltung an das externe Gerät besteht. Also bitte sehr sorgfältig alles vor der Zusammenschaltung prüfen.



Seriellport-IF, hinsichtlich Beschädigung des PC-Ports wesentlich unkritischer und auch von PonyProg ausdrücklich empfohlen.



Programmieradapter ist erforderlich, sofern die Anwendungsschaltung über keinen ISP-Sockel verfügt, beispielsweise für Atiny12 Anwendungen. Er ist ein einfach anzufertigen und der jeweilige Prozessor wird in diesem Adapter programmiert.



Das Foto zeigt einen Aufbau auf Lochrasterplatine mit einem zweiten Sockel für Atmega8-16 und nicht weiter erforderlichen zusätzlichen Bauteilen.

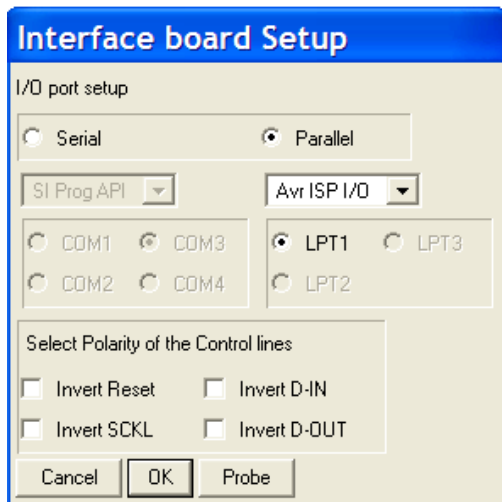
Schreiben des Hex-Files in den Prozessor mit Hilfe der Freeware PonyProg 2000:

Es wird nur das Hex-File dazu benötigt.

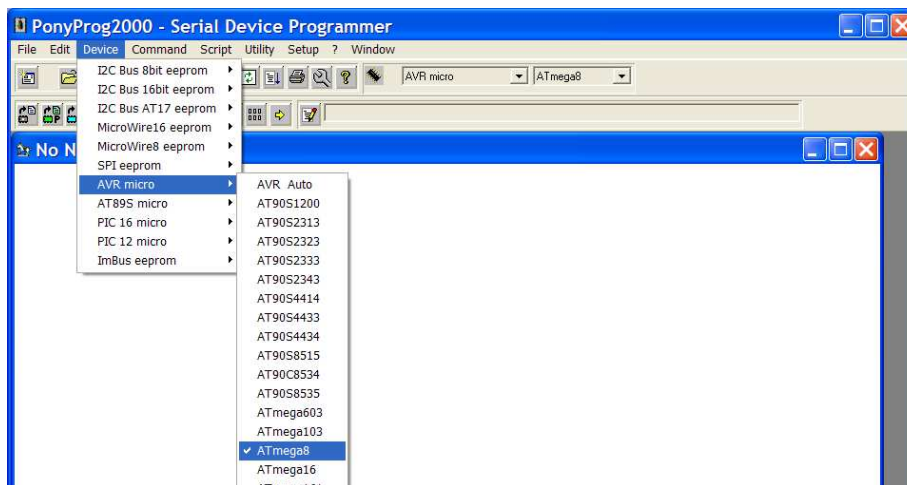
Der PC wird mit der Anwendungsschaltung (Peiler, Steuerung oder Programmieradapter) verbunden, PonyProg 2000 gestartet, die Anwendungsschaltung von einer neuen Batterie

gespeist und eingeschaltet. Batteriespeisung deshalb, da damit Stromversorgungsprobleme bei den Schnittstellen des PC's mit Sicherheit vermieden werden.

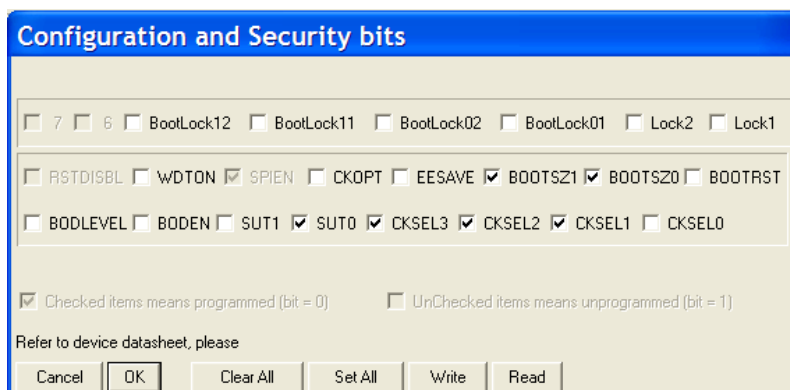
Zuerst im Setup des PonyProg 2000 das Programminterface auswählen, Beispiel gilt für den Parallelportprogrammer. Dann Calibration aus dem Setup laufen lassen.



Als nächstes zu programmierendes Device auswählen:
Beispiel Atmega 8 für 2m Fuchsjagd Empfänger.



Zuerst unbedingt über Command die Security und Configuration Bits aufrufen und auslesen.
Bei einem jungfräulichen ATmega8 muss dieses Bild erscheinen:

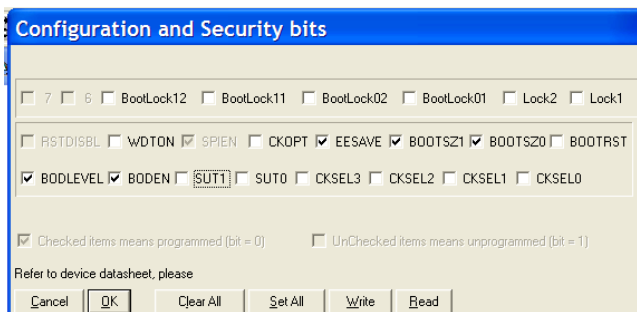


Nachstehend die Funktion der Fuse bits im Auslieferungszustand und die erforderlichen Änderungen für den DF1FO Peiler:

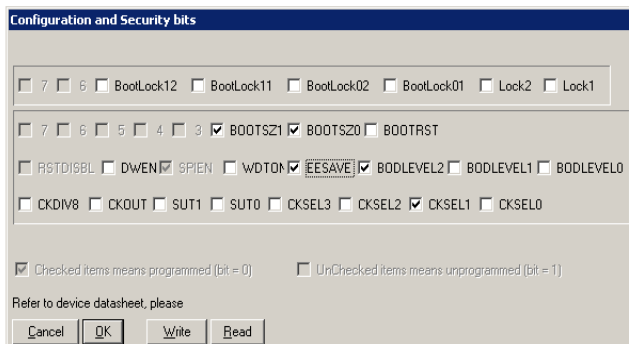
| Bezeichnung | Auslieferung | Funktion im Auslieferungszustand | DF1FO Änderung |
|-------------|--------------|--|----------------|
| BootLock12 | 1 | Boot lock bit | |
| BootLock11 | 1 | Boot lock bit | |
| BootLock02 | 1 | Boot lock bit | |
| BootLock01 | 1 | Boot lock bit | |
| Lock2 | 1 | Lock bit | |
| Lock1 | 1 | Lock bit | |
| RSTDISBL | 1 | PC6 is RESET-pin) | |
| WDTON | 1 | WDT enabled by WDTCR | |
| SPIEN | 0 | SPI prog. enabled | |
| CKOPT | 1 | Oscillator options | |
| EESAVE | 1 | EEPROM not preserved) | 0 |
| BOOTSZ1 | 0 | Select Boot Size, Table 82 | |
| BOOTSZ0 | 0 | Select Boot Size, Table 82 | |
| BOOTRST | 1 | Select Reset Vector | |
| BODLEVEL | 1 | Brown out detector trigger level | 0 |
| BODEN | 1 | BOD (Brown out Detector) | 0 |
| SUT1 | 1 | start-up | 1 |
| SUT0 | 0 | start-up | 1 |
| CKSEL3 | 0 | Calibrated Internal RC Oscillator 1MHz | 1 |
| CKSEL2 | 0 | Calibrated Internal RC Oscillator 1MHz | 1 |
| CKSEL1 | 0 | Calibrated Internal RC Oscillator 1MHz | 1 |
| CKSELO | 1 | Calibrated Internal RC Oscillator 1MHz | 1 |

Unter Berücksichtigung der Atmega-Konventionen und der des PonyProg 2000 sind für die DF1FO Programmierung zuallererst folgende Fuse-Änderungen einzustellen und dann zu schreiben.

Beim Atmega 8:



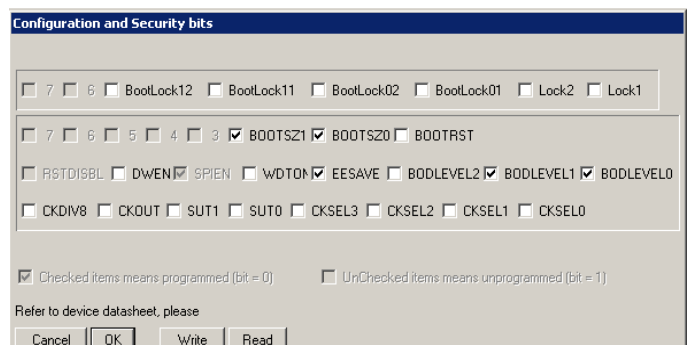
Atmega 168 (3-8MHz Crystal):



Beim ATtiny12 (für Fuchssteuerung):



Atmega 168 (8-20 MHz Crystal)



Das Setzen der BOD.. Fuses bewirkt, dass bei einer Einsenkung der Prozessorbetriebsspannung unter einen bestimmten Wert, ein Neustart des Programmes erfolgt und keine Ablaufstörung eintreten kann. Nach einem solchen Neustart, dieser kommt

einem Ausschalten und Wiedereinschalten des Peilers gleich, sind natürlich die Fuchs-Timerwerte wieder einzustellen.

Erst danach das Hex-File (wenn nur Assembler-File vorhanden, dann Hex-File nach Teil 2 erstellen) über File/Open Device File in das PonyProg laden und über Command/Write All in den Atmega8 schreiben.

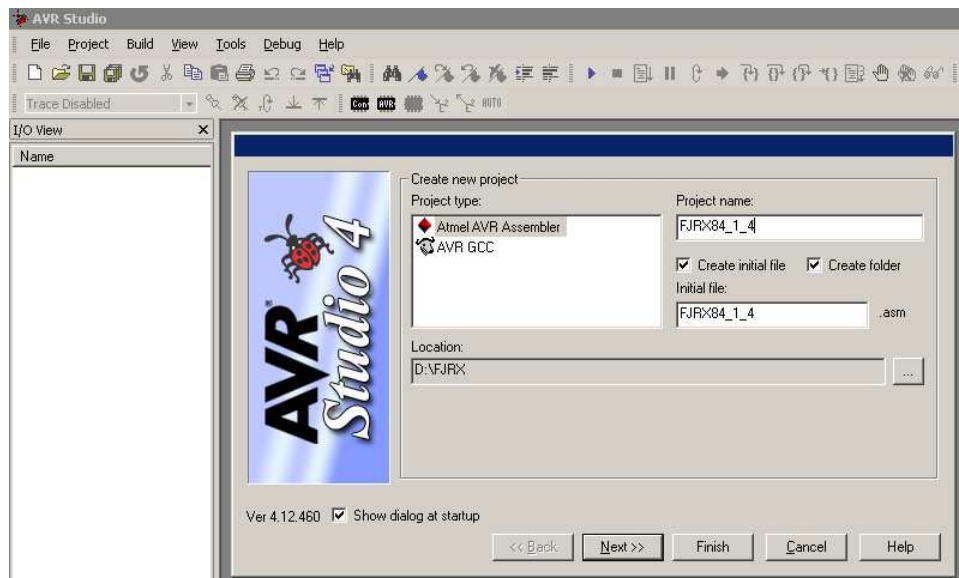
Das dauert eine kurze Weile, der Vorgang wird automatisch verifiziert und es muss abschliessend eine Meldung über den positiven Abschluss erscheinen. Dann Spannung der Anwendungsschaltung ausschalten, ISP-Programmier-IF abziehen und Neustart der Anwendung.

Teil 2: Wandlung des Assembler Sourcecode (*.asm) in ein Hex-File (*.hex).

Für die Wandlung in ein Hex-File ist zuerst von der ATMEL Homepage http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725 das Programm AVR Studio 4 herunter zuladen und auf dem PC zu installieren.

Auf dem PC ist ein Ordner für dieses Projekt einzurichten. Zum Beispiel D:\FJRX. In diesen Ordner wird das Assembler Sourcecode File (beispielsweise FJRX84_1_4.asm (Anmerkung: ich ergänze das asm File gleich immer mit der Versionsnummer)) kopiert.

Programm AVR Studio 4 aufrufen und unter Project/neues Project definieren. Atmel AVR Assembler markieren und als Projektname beispielsweise FJRX84_1_4 verwenden,

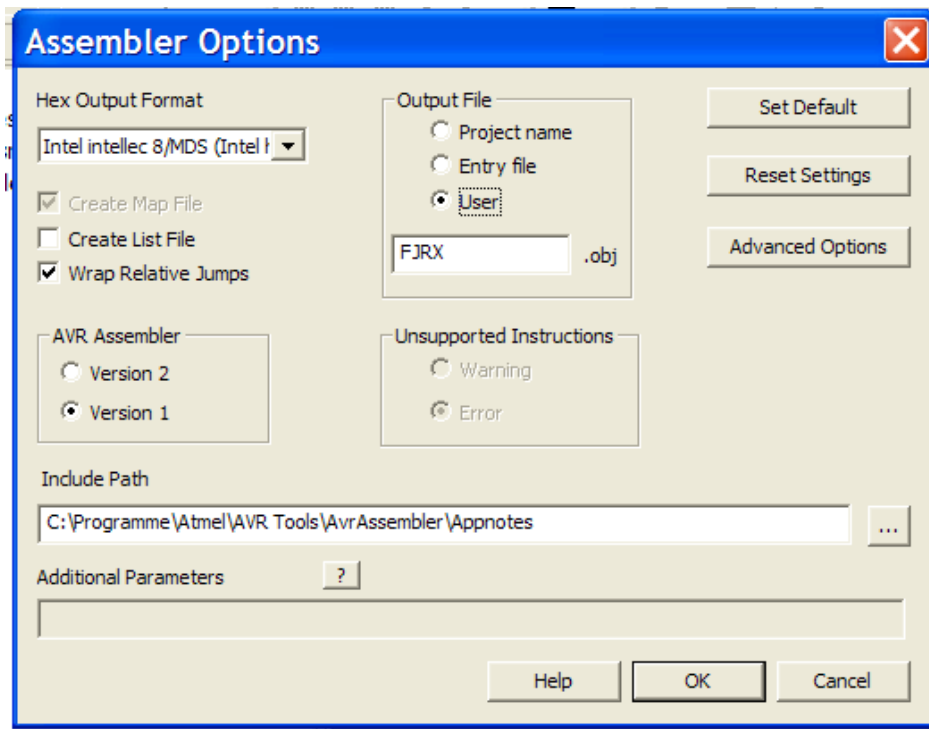


Next und dann Plattform und Prozessor auswählen, beispielsweise ATmega8.



Finish drücken, dann mit File/openFile das vorher in D:\FRJ abgespeicherte File FJRX84_1_4.asm öffnen. Mit File/save as dieses File in den Ordner FJRX84_1_4 abspeichern und mit File ersetzen (damit wird das soeben angelegte leere gleichnamige File überschrieben) bestätigen.

Mit Project/Assembler Options folgende Einstellungen wählen und mit OK bestätigen:



Danach Assemblieren mit Build/Build oder F7.

Ergebnis wird in der unteren Bildschirmhälfte angezeigt. Es sollten keine Fehler gemeldet werden, sonst nochmals alle Einstellungen prüfen. Program memory usage ist natürlich von Version zu Version unterschiedlich.

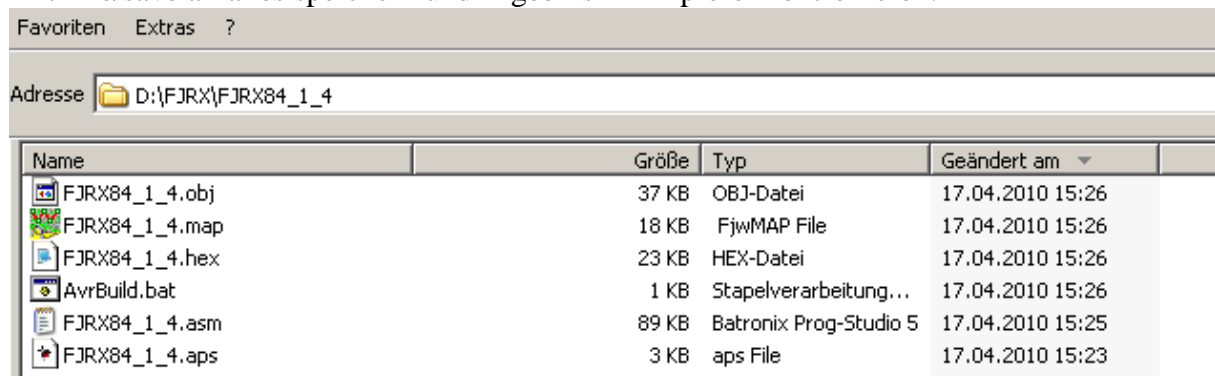
```
Project | I/O ... | Info | D:\FJRX\FJRX84_1_4\FJRX84_1_4.asm | D:\FJRX\FJRX84_1_4\FJRX84_1_4.asm
Build
AVRASM: AVR macro assembler version 1.77.3 (Sep 21 2005 08:43:03)
Copyright (C) 1995-2005 ATMEL Corporation

Assembling 'D:\FJRX\FJRX84_1_4\FJRX84_1_4.asm'

Program memory usage:
Code          : 3223 words
Constants (dw/db): 945 words
Unused        : 0 words
Total         : 4168 words

● Assembly complete with no errors.
```

Mit File/save all alles speichern und Ergebnis im Explorer kontrollieren.



Weiterverwendet mit PonyProg2000 wird nun nur mehr das File FJRX84_1_4.hex, welches wir an passender Stelle abspeichern.

Anschließend löschen wir den Inhalt des Ordners D:\FJRX für spätere weitere Nutzung.

73, Harald, OE6GC, 17.5.2007, ergänzt am 17.4.2010