Das Pollin Atmel Evaluationboard Ver. 2.01 unter Verwendung von MyAVR_Prog Tool und der Arduino IDE – ein Kurztutorial

15.07.2012 – V0.9c

Einleitung

Angeregt durch Torsten, DL8KFO habe ich mich angefangen mit der Arduino-Programmierumgebung zu beschäftigen (http://www.arduino.cc/). Da ich zum einen nicht in ein Arduino-Board investieren wollte, mir aber bereits ein fertiges AVR-Evaluationsboard von Pollin (810 074 Fertigmodul bzw. 810 038 Bausatz) zur Verfügung stand entschied ich mich, dieses Board zum Experimentieren und Programmieren zu nutzen.

Grundsätzliches

Die Arduino-IDE beruht im Wesentlichen auf einem Programm dass den AtMega per serieller Schnittstelle beschreibt. Dazu ist es aber notwendig, dass der Controller bereits mit einem sogenannten Bootloader beschrieben ist, sozusagen das „BIOS“ des Prozessors. In einem neu gekauften AtMega ist nun aber kein Arduiono-Bootlader enthalten – also benötigen wir ein Programm, welches in der Lage ist den AtMega auf einer sehr einfachen Ebene anzusprechen und zu beschreiben.

Voraussetzungen

Vorausgesetzt wird ein Windows-PC mit zwei seriellen Schnittstellen. Optimal sind natürlich „echte“ Com-Ports in Hardware, ggf. auch als PCI-Steckkarte, speziell für den ISP-Programmerteil. Wenn USB-Seriell-Adapter verwendet werden muss die Eignung im Einzelfall überprüft werden. Der RS-232-Anschluß zur Arduino-Programmierung ist nach meiner Erfahrung dahingehend unkritisch, ich verwende dazu einen USB-Seriell-Adapter mit FTDI-Chip. –Das geht natürlich alles auch mit Mac oder Linux-PC, ist aber dem geneigten Zuhörer als Hausaufgabe überlassen;-)

Verwendete Software

Als Programm zum „low-level“-Programmierung kommen Software wie „AVR-Studio“ von Atmel oder „Pony-Prog“ in Frage. Ich habe mich für myAVR_Prog Tool (http://shop.myavr.de/index.php?sp=download.sp.php&suchwort=dl112) auf Grund des sehr Benutzerfreundlichen Interfaces entschieden. Daher werde ich mich hier ausschließlich auf die Einstellungen in diesem Programm beziehen.

Vorbereitung (Software)

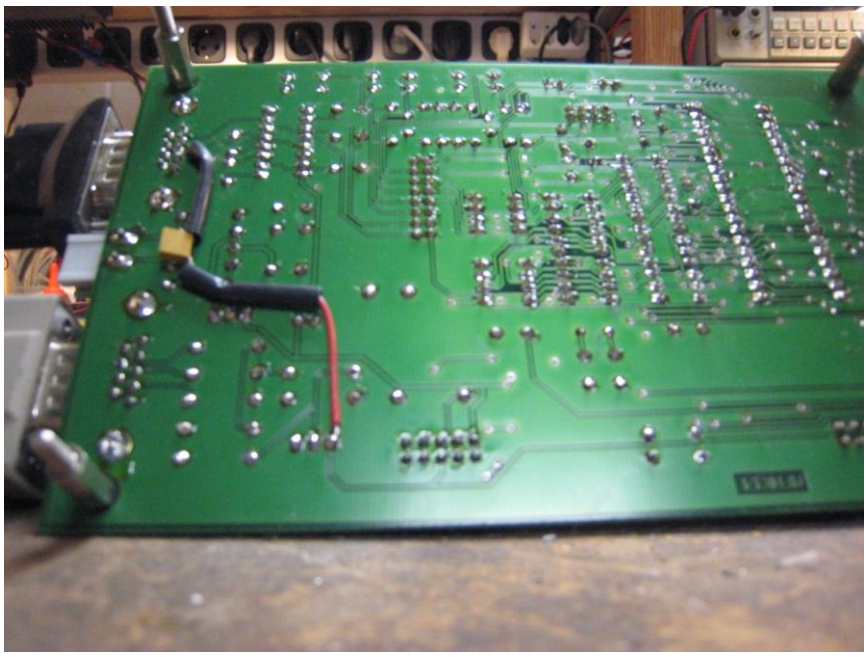
Ich gehe jetzt davon aus dass sowohl MyAVR Prog Tool wie auch die Arduino IDE installiert sind.

Vorbereitung (Hardware)

Da es –ohne Modifikation- beim Pollin-Board nötig ist zum Schreiben eines Arduino-Sketches „im richrtigen Moment“ den Reset-Taster zu drücken habe ich mich zu einer kleinen Hardware-Änderung des Pollin-Boards entschlossen, angeregt durch

http://www.arduino.cc/playground/Learning/AutoResetRetrofit

Dabei wird einfach ein kleiner C, 100nF, zwischen dem RS232-DTR-Anschluß Pin 4 und der Reset-Leitung des EvalBoards gelegt. Das sieht dann bei mir so aus:

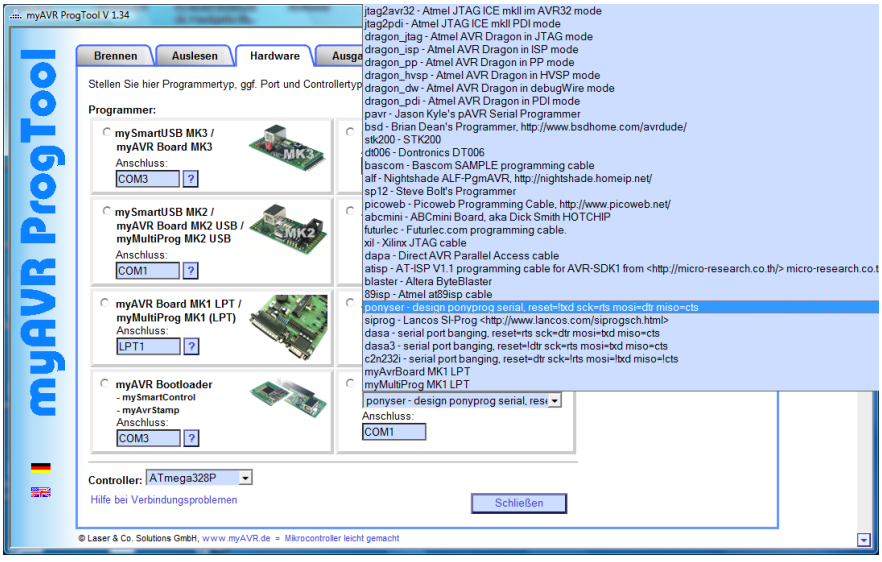


Pollin-AVR-Evaluationsboard mit Modifikation (Autoreset f. Arduino)

<u>Programmeinstellungen:</u>

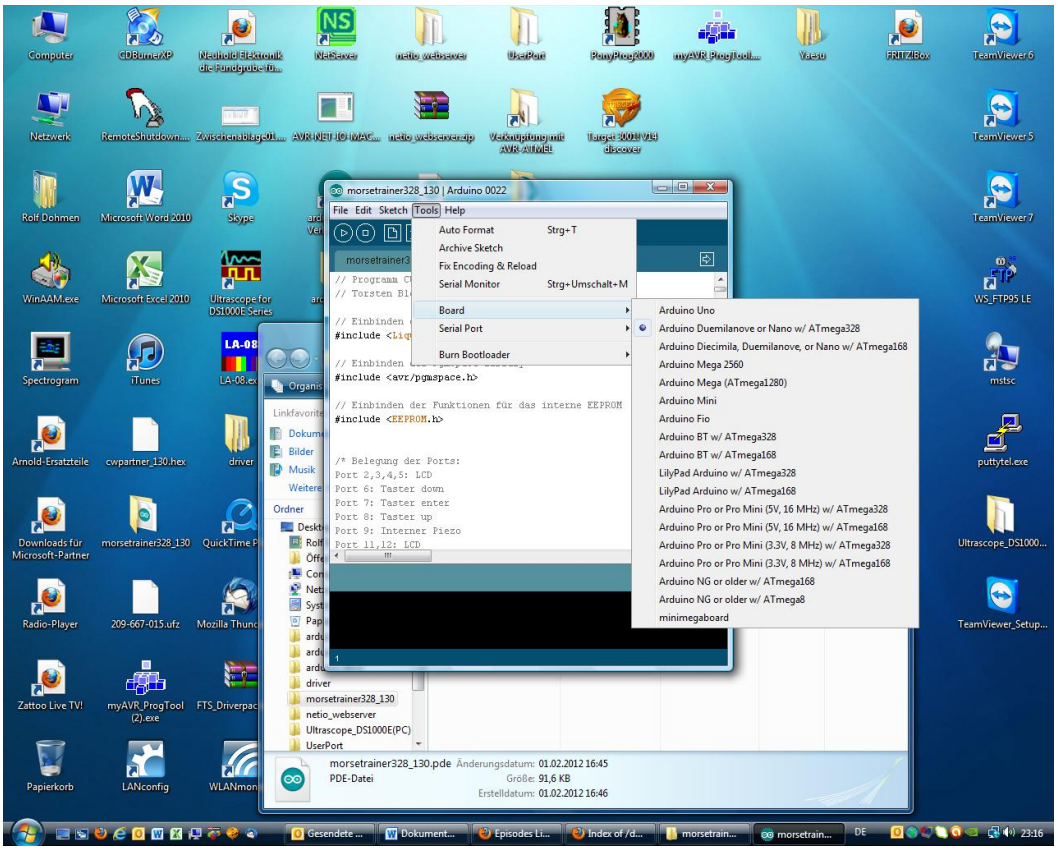
<u>-MyAVRProgTool:</u>

Unter „Hardware“ ist rechts unten „Sonstiges“ mit „ponyser…[…]“ zu wählen, wichtig dabei den *richtigen* COM-Port auszuwählen! Als Controller (für den CW-Partner) wird unten links der ATMega328P gewählt.



-Arduino-IDE:

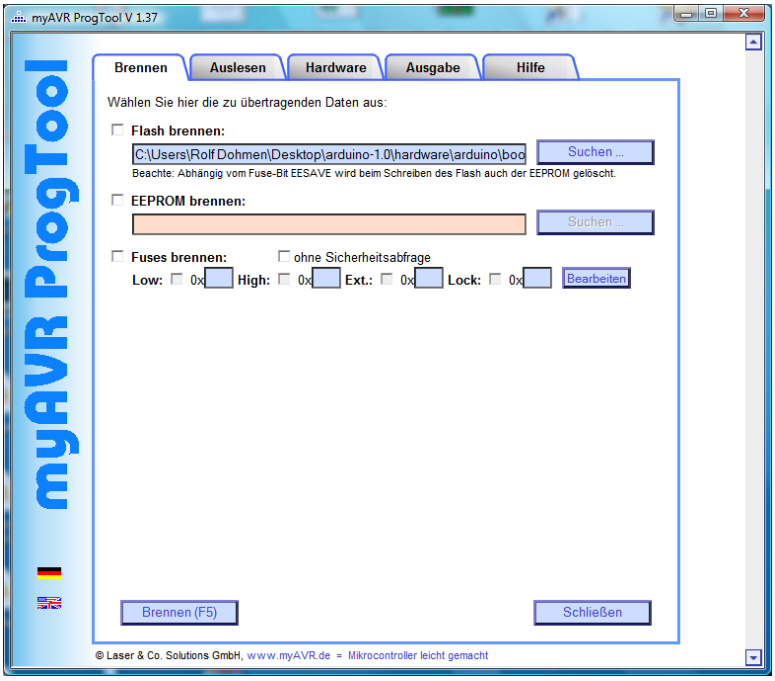
Unter „Tools/Board“ ist „Arduino Duemillanove or Nano w/ ATmega328“ auszuwählen sowie unter „Serial Port“ die verwendete Com-Schnittstelle (sinnvoller Weise eine andere als für MyAVR!):



Schreiben des Bootladers und der Fuses mit MyAVR Prog Tool:

Wir starten MyAvrProgTool durch Doppelklick auf die geladene exe.-Datei.

Danach bietet sich uns der Anblick des Hauptfensters von MyAVRProgTool:



Hier sehen wir oben in der ersten Zeile bereits den von mir ausgewählten Pfad zum Arduino Bootlader für einen AtMega328, zu finden unter

<Arduino-Programmverzeichnis>\hardware\arduino\bootloaders\atmega\

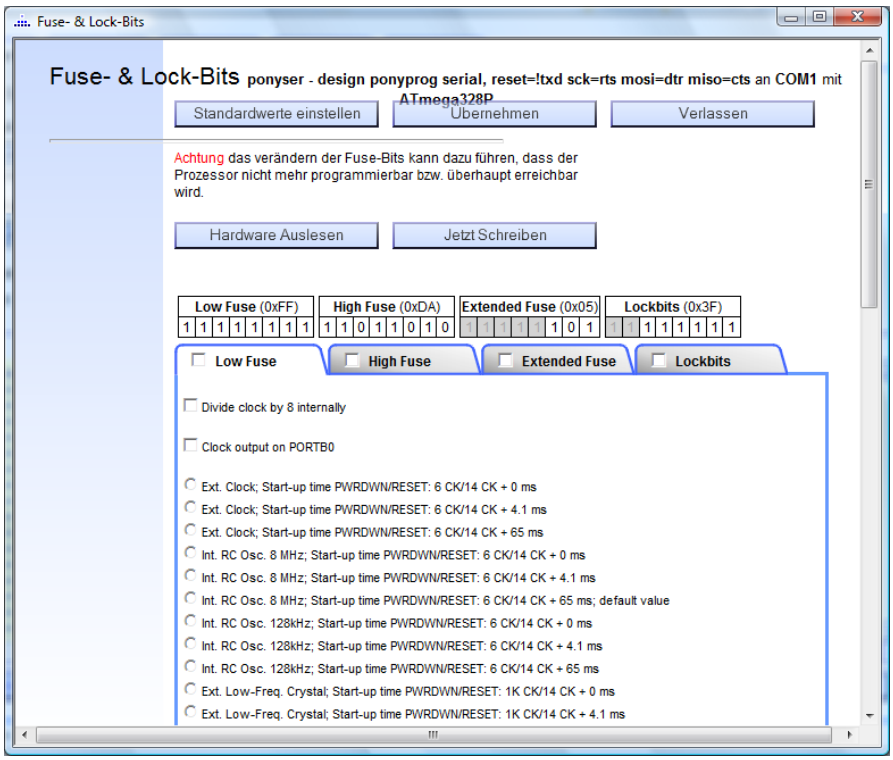
ATmegaBOOT_168_atmega328.hex

Die zweite Zeile, EEprom brennen, ist jetzt ganz interessant für diejenigen, die sich den AtMega für unser Projekt „CW-Partner“ (http://www.qsl.net/d/dl1kj//cwpartner/) brennen wollen; hier ist es nötig das integrierte EEprom des AtMega erst einmal mit sinnvollen Werten zu laden. Dazu kann zum einen Torsten Sketch „cwpartner_init.pde“ per Arduino-IDE geladen werden, oder gleich zusammen mit dem Bootlader und den Fuses „eeprom.hex“ (siehe Download-Link) ins Eeprom geschrieben werden.

Die Fuses:

Ganz wichtig bei der Programmierung eines AtMega sind die Fuses! Hier werden Dinge wie z.B. Einsprungadresse des Bootladers, dessen Größe, interne oder externe Clocksource und Taktfrequenz festgelegt. BEI FALSCHER FUSEPROGRAMMIERUNG IST ES MÖGLICH SICH KOMPLETT „ABZUSCHIESSEN“, d.h. ein Zugriff auf den AVR ist mit normalen Mitteln nicht mehr möglich!!

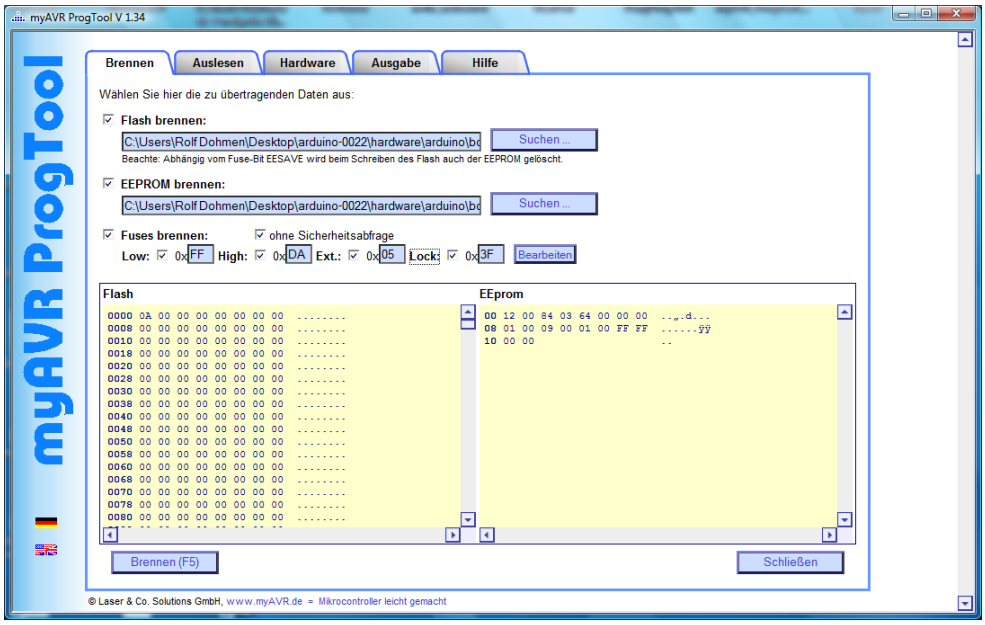
Hier meine Fuse-Einstellungen für 16 MHz externen Takt, Bootladergröße 1024 Bytes ab 0x3C00, Boot reset vector enabled:



Zusammengefasst: **LowFuse 0xFF HigFuse 0xDA Ext'dFuse 0x05 Lockbits 0x3F**

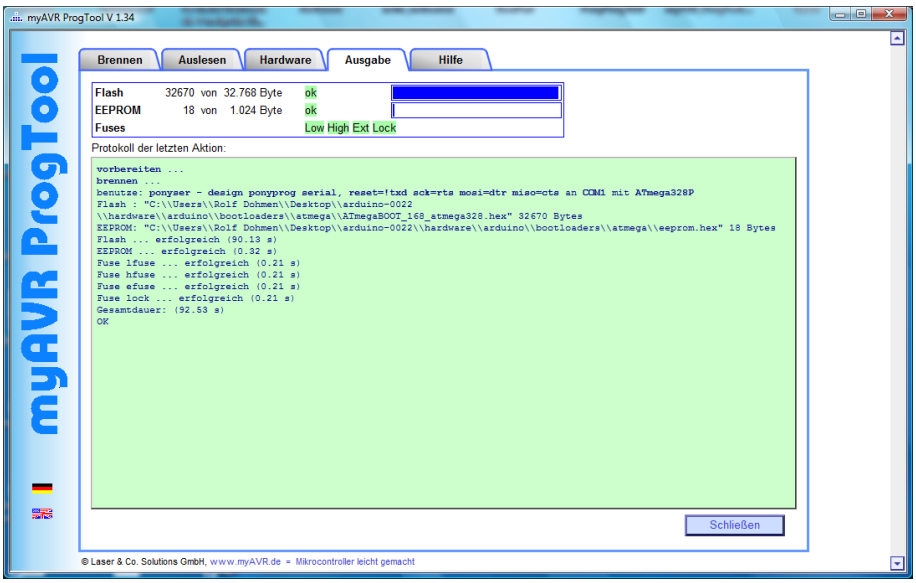
<u>Brennen von Bootlader, EEprom und Fuses:</u>

Das alles zusammen sieht in MyAvrProgTool dann so aus:



Wer jetzt auf „Brennen" klickt (F5) sollte schon ziemlich sicher sein was er/sie tut;-) Das ist nämlich der „Point of no Return"…

Bei Erfolg sollte ein solcher Screen zu sehen sein:



CONGRATS! Der ATmega ist nun mit den nötigen Fuses und em Bootlader „gebrannt“ und kann nun per Arduino-Programmumgebung weiter programmiert werden!

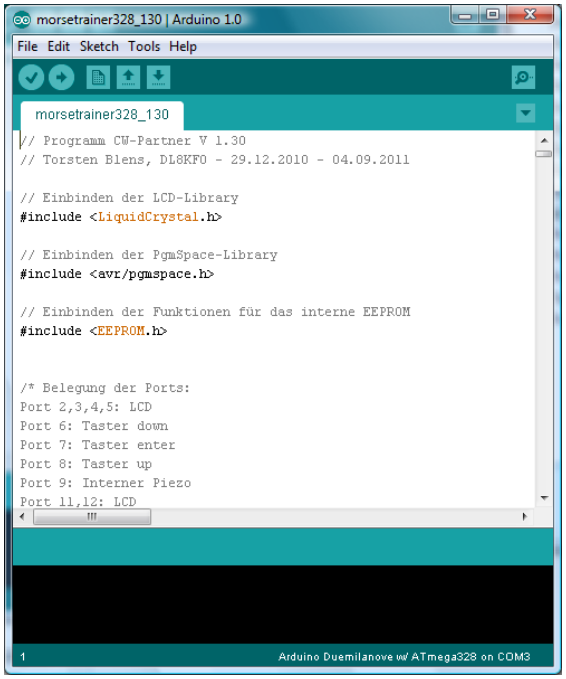
<u>Programmierung per Arduino-IDE:</u>

Einen mit Bootlader versehenen und mit gültigen Fuses gebrannten ATmega kann man nun sehr komfortabel mit der Arduino Programmierumgebung mit entsprechenden Programmen, in der Arduino-Terminologie „sketch“ genannt, beschreiben.

Am Beispiel der Version 1.30 von „CW-Partner“ ginge dies so:

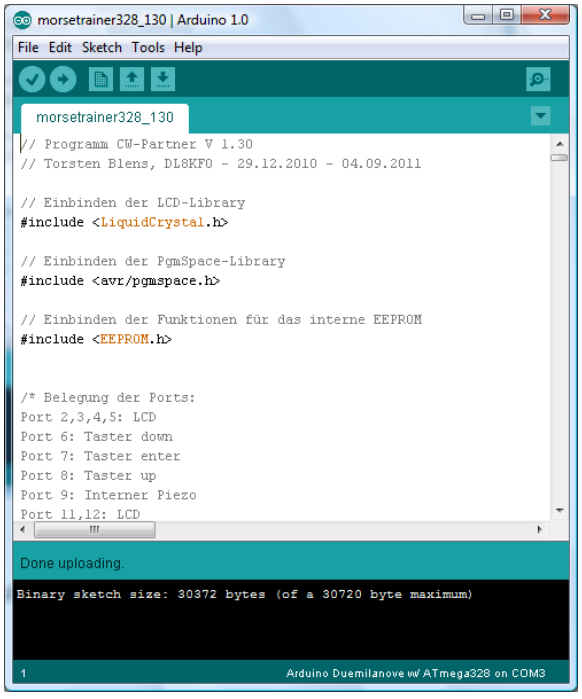
-Starten der Arduino-Programmumgebung

-Öffnen des Programms (hier: morsetrainer328_130.pde)



Brennen: Klick auf den „Upload“-Button (zweiter von links, der mit dem Pfeil nach rechts)

WENN alles gut: Dann steht unten sowas wie „Done uploading“



-Im Pollin-Board merkt man auch gleich dass was „passiert“ ist: LED1 fängt an in unregelmäßigen Abständen zu blinken.

–Damit ist die Programmierung abgeschlossen und der ATmega kann im CWPartner-Board eingesetzt werden!