Bitwackler-ISP

Einsteiger und Anfänger in die Microcontrollerwelt gibt es immer wieder und das ist erfreulich, da das bedeutet, dass es potentiell immer wieder jemanden geben wird, der richtig zündende Ideen hat.

Sehr viele haben ein klein wenig Ahnung wie ein Programm funktioniert, sei es in C, Pascal, Basic oder gar in Maschinensprache und möchten das jetzt gerne mit einem Microcontroller ausprobieren.

Manche von Ihnen kennen sich auch in der Elektronik recht gut aus. Umso erstaunlicher ist es, welche Gedanken es darüber gibt, wie ein Programm – denn jeder Controller benötigt für seine Arbeit ein Programm – in den Controller gelangen soll.

Manche fragen sogar, wie ich die Pins denn an die USB anschließen muß, damit das Programm in den Chip gelangt.

Und auch Fragen der Art: „Muß ich da wirklich einen Programmer kaufen?“ gibt es zuhauf. Relativ bald kommt die Einsicht ohne Programmer geht es wirklich nicht, aber da man sich in Elektronik ja auskennt und man gerne bastelt nimmt man sich vor: Selbstbauen !

Da fängt es dann aber an, denn die allermeisten Bauvorschläge benötigen einen bereits programmierten Microcontroller und da man keinen Programmer hat kann man das auch nicht aufbauen!

Das klassische Problem: Ich brauche einen Programmer um einen Programmer zu bauen. Nur, wenn ich bereits einen Programmer habe, brauche ich mir ja auch keinen zu bauen !!!

Ist man in der Materie schon etwas drin, ist alles relativ easy und leicht und eines der kleinsten Probleme ist dann, ein Programm in den Controller zu bekommen (ein funktionierendes Programm ist bisweilen eine weitaus größere Hürde... später).

Aber am Anfang ist tatsächlich die größte Hürde: Wie bekomme ich mein Programm in den Controller.

Lustigerweise findet der geneigte Bastler im Internet relativ schnell eine Lösung, die ohne programmierten Chip auskommt und meistens landet er auf der Seite von PonyProg:

http://www.lancos.com

Das Programm dort ist wirklich nicht schlecht (gewesen) aber mittlerweile sind Druckerports und serielle Schnittstellen Mangelware geworden, außerdem ist das Programm in die Jahre gekommen. Zudem sind die Schaltpläne für einen nicht in der Materie steckenden bisweilen verwirrend und die Verstellmöglichkeiten des Programms riesengroß (für denjenigen der sich auskennt schön, weil man einiges Ausprobieren kann).

Für denjenigen der gerade einsteigt fatal, er wird ohne fremde Hilfe scheitern.

Was ist also zu tun, wenn jemand partout nicht hören will und sich NICHT einen bereits funktionierenden Programmer zulegen möchte.

Nun, dann eben doch selbst bauen und da kommt man dann an den „Bitwacklern“ nicht vorbei, weil dieses die einzige Methode ist, mit der ein Chip ohne einen bereits programmierten Controller geflasht werden kann.

Die „Bitwackler“ heißen im Übrigen „Bitwackler“, weil ein PC Programm ohne zusätzliche Hardware die Pins einer Schnittstelle ohne ein genormtes Schnittstellenprotokoll und ohne jegliche zusätzliche Hardware auf 1 bzw. auf 0 setzt.

Hier fängt das große Problem an. Im Laufe der Jahrzehnte sind PC’s immer schneller geworden und das Timing hängt vom PC ab, ob ein Controller da das richtige Timing auf seiner Clock-Leitung erhällt ist manchmal etwas kritisch.

Wenn man also einen „Bitwackler“ zum Funktionieren bekommt ist es ratsam, sich den wohl preiswertesten Programmer MIT Controller aufzubauen, den USBtinyISP (dazu jedoch später).
Vor dem USB Programmer (der dann auch mit der neuesten Windows 8 64-Bit Maschine funktionieren wird) muss es erst einmal geschafft werden, sich eine Möglichkeit zu verschaffen, einen Controller flashen zu können.

Hierfür benötigt es dann (leider) älterer PC Hardware.

Der „Bitwackler“ oder der wohl schrottigste (und billigste) Programmer der Welt

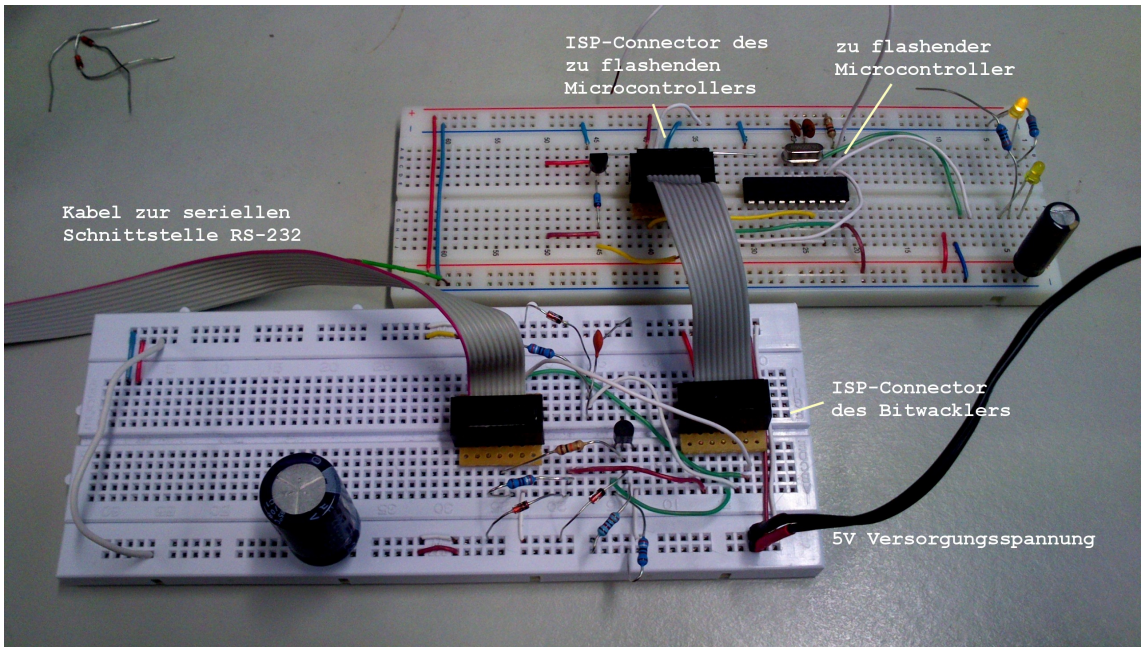
Die folgende Methode habe ich an diesen PC’s zum Funktionieren gebracht:

- **ZWINGEND benötigt der PC eine echte serielle Schnittstelle (ein USB zu RS232 Adapter funktioniert definitiv nicht, ich habe mehrere ausprobiert)**

- **Betriebssystem ist Windows 2000 oder Windows XP (aber außsschließlich 32 Bit Systeme, ich habe es zuverlässig nicht mit einem 64 Bit System geschafft).**

Wer also Zugriff auf einen Win 2000 oder WinXP Computer mit ECHTER serieller Schnittstelle hat, hat gute Chancen, einen Microcontroller OHNE bereits programmierten Chip aufbauen zu können.

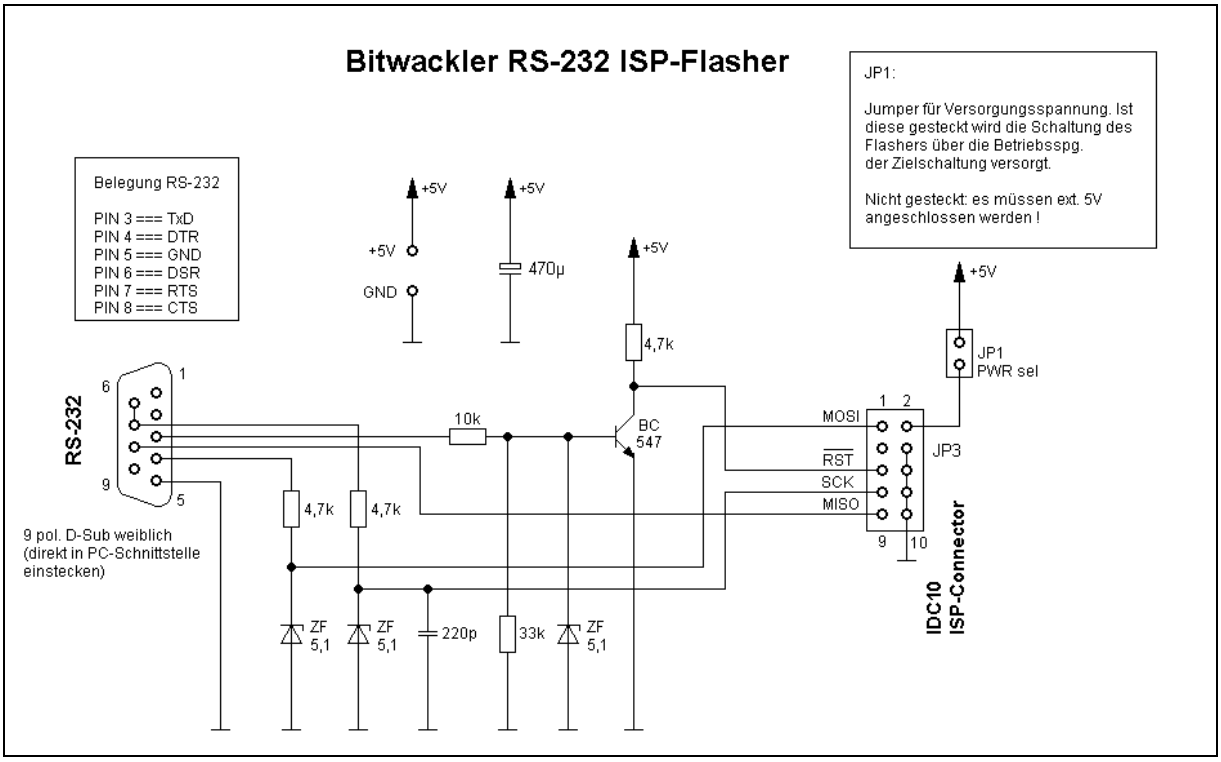
Hier ist der Schaltplan einer der wohl einfachsten ISP-Programmers die es geben kann (es gibt auch noch Versionen mit LPT-Druckerschnittstelle, diese ist meiner Meinung nach noch etwas schwieriger zum Laufen zu bringen, außerdem ist der Stecker noch breiter als der einer seriellen Schnittstelle).



Der wohl schrottigste ISP-Flasher mit dem wohl schrottigsten Aufbau (und er funktioniert dennoch)

Der fliegende Aufbau auf einem Steckbrett diente nur dem einzigen Zweck, einen anderen Controller mit einem Flashprogramm flashen zu können !

Schaltplan des Bitwacklers



An Einfachheit wohl nicht zu Überbieten !!

Bauteile:

```
3 x Widerstand 4,7k
1 x Widerstand 10k
1 x Widerstand 33k
1 x Kondensator 220p
3 x Zenerdiode mit Zenerspannung 5,1V (z.Bsp. ZF 5,1 für 9 ct)
1 x NPN Transistor BC 547
1 x IDC Wannenstecker 10 pol.
1 x 9 pol. D-Sub Buchse (weiblich)
```

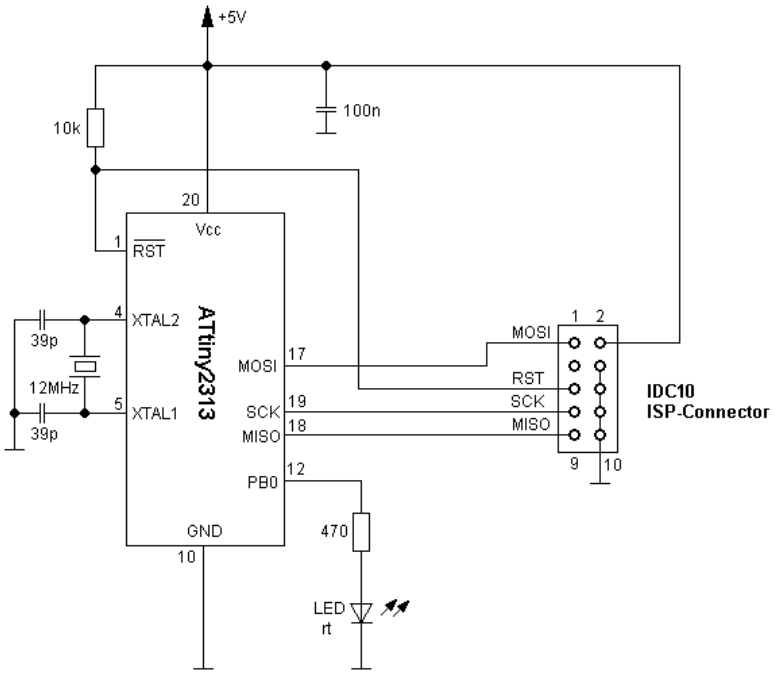
Die Schaltung versteht sich in der Hinsicht, dass die im Schaltplan der Buchse zugeführten Leitung direkt an die Buchse zu verlöten sind und diese direkt in den RS-232 Anschluss des PC's gesteckt werden.

Der Jumper JP1 dient zum An- oder Wegschalten der Versorgungsspannung. Wird die Versorgungsspannung am zu flashenden Controller angelegt, so ist JP1 zu stecken, damit der Bitwackler spannungsversorgt ist.

Soll der Bitwackler (warum auch immer) eine eigene Versorgungsspannung erhalten, ist JP1 zu trennen.

Die Verschaltung eines ATtiny2313 mit einem ISP-Anschluss 10pol.

Weil gerade Einsteiger auch mit der Verschaltung der Anschlüsse des ISP-Connectors Probleme haben (können), hier ein Schaltplan, wie dieses bewerkstelligt sein muss. Da ein Programm geflasht werden soll, das später mit einem Quarz laufen wird, ist in der Verschaltung ein Quarz angeschlossen.



Ein 10 pol. Flachbandkabel, das an beiden Enden mit einem IDC Steckverbinder verpresst ist, stellt die Verbindung ziwschen Flasher und Schaltung her (dieses Kabel ist auch für andere ISP-Flasher verwendbar).

Software

Jetzt wird es „schwierig“. Welches Programm wackelt die Daten über die serielle Schnittstelle zum Bitwackler. Prinzipiell wäre das Eingangs erwähnte PonyProg dafür geeignet. Allerdings ist auch bekannt, dass zum einen die extrem vielen Einstellmöglichkeiten den Anfänger vor eine große Hürde stellen und zum anderen PonyProg nicht so wirklich gut mit Windows XP zusammenarbeitet.

DAS Standartprogramm zum Flashen von Atmels AVR-Microcontrollern dürfte wohl

AVRDUDE

sein.

AVRDUDE hat allerdings nicht wirklich weniger Einstellmöglichkeiten als PonyProg und hat zudem noch die „Schwierigkeit“ dass es ein Konsolenprogramm ist (muß vom sogenannten DOS-Prompt) gestartet werden.

Der Vorteil jedoch liegt darin, dass hier eine Kommandozeile eingegeben werden kann und nicht irreführend lange erklärt werden muss, wo nun auf welche Buttons zu klicken ist, damit ein Programmer funktioniert !

AVRDUDE ist u.a. Bestandteil des Open-Source Projekts „WinAVR“ und von daher kostenfrei (wie gut !).

Damit nun nicht lang und breit erklärt werden muss, wie etwas zu installieren ist, wie Systempfade für WinAVR zu setzen sind (obwohl das im Normalfall die Installationsroutine von alleine bewerkstelligt) kopiert man nun pragmatischerweise folgende Dateien in ein und das selbe Verzeichnis (im folgenden Beispiel war dieses das Verzeichnis C:\bitbang_isp ... es könnte aber genausogut C:\bitwackler heißen).

Folgende Dateien müssen in dieses Verzeichnis kopiert werden:

AVRDUDE.EXE
AVRDUDE.CONF
USBTINYISP_V3.HEX

Als nächster Schritt muss ein Konsolenfenster (DOS-Prompt – Ich mag das Wort „DOS-Prompt“ nicht, weil die Konsole von Windows 2000 und Windows XP KEIN DOS ist!!!)

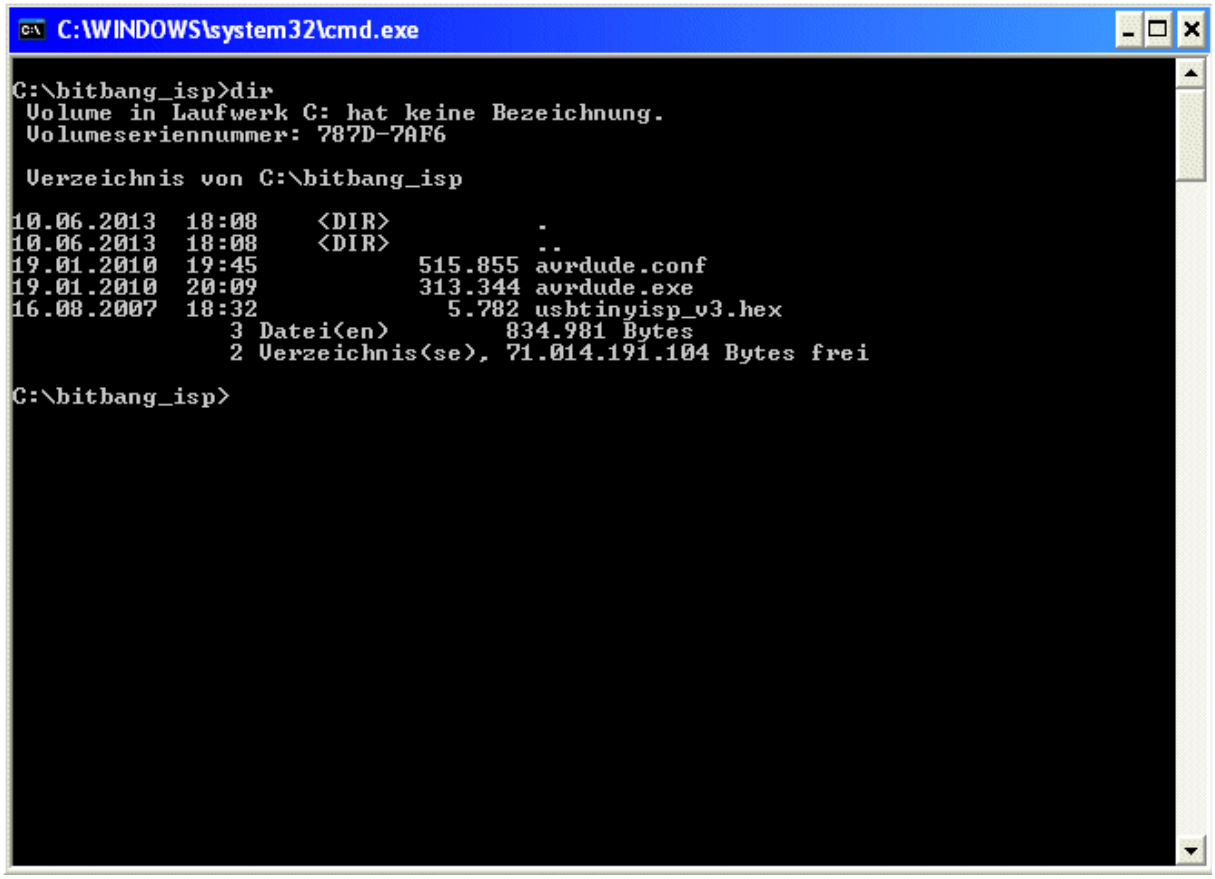
Im Konsolenfenster gibt man folgendes ein:

```
cd c:\bitbang_isp
```

(wenn das Verzeichnis „bitbang_isp“ genannt wurde). Der nächste Schritt ist, sich das Verzeichnis mittels

```
dir
```

anzeigen zu lassen um sich zu vergewissern, dass auch alle Dateien vorhanden sind, es muss dann folgender Screen angezeigt werden:



Jetzt wird es „spannend“, es entscheidet sich nun, ob der geneigte Anfänger, der bis hierher gelesen und die Schaltung aufgebaut hat, in der Lage ist, seinen ersten Microcontroller (einen ATtiny2313) zu flashen.

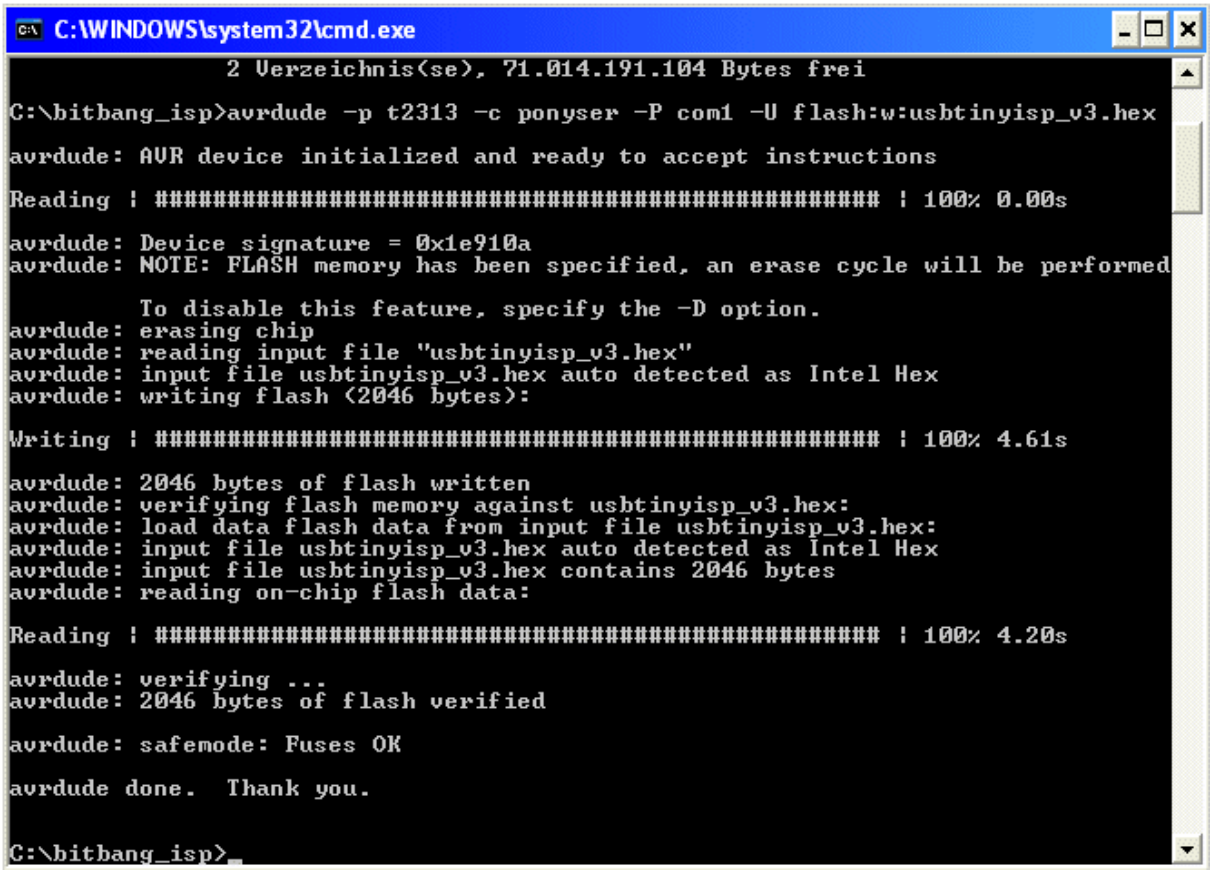
Natürlich hat er aller Verbindungen noch einmal überprüft, die Versorgungsspannung 5V angelegt (die hoffentlich einem geregelten Netzteil entspringt – bei mir ist es ein altes geregeltes Motorola Handynetzteil) und das Flachbandverbindungskabel des Bitwacklers mit dem Zielcontroller / Targetcontroller (so heißt der zu flashende Microcontroller) verbunden.

Wenn also dies alles gemacht ist, gibt am „DOS-Prompt“ folgendes ein:

```
avrdude -p t2313 -c ponyser -P com1 -U flash:w:usbtinyisp_v3.hex
```

Ist der Bitwackler NICHT an com1 angeschlossen, sondern bspw. an COM2 so muss die Zeile entsprechend geändert werden.

Wenn jetzt alles, wirklich alles richtig gelaufen ist, sollte am Bildschirm folgendes angezeigt werden:



Erläuterungen zu AVRDUDE:

AVRDUDE ist ein Kommandozeilenprogramm ohne jegliches Klicken oder „Drag and Drop“. Was AVRDUDE machen soll wird dem Programm durch Parameter gesagt. Parameter sind Anweisungen oder Informationen, die dem Programmnamen folgen und im Falle von AVRDUDE durch ein Leerzeichen getrennt sind. Die Parameter von AVRDUDE haben ihrerseits wieder „Unterparameter“, hier sollen die Parameter erklärt werden, die zum Flashen notwendig sind.

Ein Parameter wird bei AVRDUDE mit einem Minuszeichen „-“ gefolgt von einem Buchstaben eingeleitet. Dieser Buchstabe UNTERSCHEIDET Groß- und Kleinbuchstaben, ein „–p“ bedeutet etwas anderes als ein „-P“.

Die verwendeten Parameter:

-p (p)artlist, die Angabe, die –p folgt bestimmt, welchen Chip AVRDUDE flashen soll

Hier gegeben für einen ATtiny2313:	-p t2313
Beispiel für einen ATMega8:	-p m8
Beispiel für einen ATMega168:	-p m168

-c zu verwendender Programmer

Hier gegeben:	-c ponyser
Beispiel für Programmer USBtinyISP:	-c usbtiny
Beispiel für Programmer USBasp	-c usbasp
Beispiel für Programmer STK500	-c stk500

-P (P)ort, an den der Programmer angeschlossen ist, muss für die meisten USB-Programmer wie USBtinyISP oder USBasp entfallen, für den Bitwackler ist die Angabe jedoch essentiell.

Hier gegeben:	-P com1
Beispiel für COM2	-P com2

-U (U)pload, die Hauptaufgabe für AVRDUDE. Hier wird angegeben, welche Datei in den Controller upgeloaded oder welche Fuses gesetzt werden sollen.

Hier gegeben: -U flash:w:usbtinyisp_v3.hex

hier bedeutet:	flash“	- Der Flash-Speicher ist ausgewählt
	w	- (w)rite (schreiben)

Die Fuses

Ein immer wiederkehrendes Thema (und bei manchen sogar mystifiziert) sind die Fuses der AVR-Controller.

AVR Controller sind in vielen Belangen konfigurierbar. Es kann bspw. eingestellt werden, ob ein Controller zur Takterzeugung mittels einem integrierten Taktgebers (Widerstand-Kondensator oder auch RC-Oszillators), einem Resonator oder einem Quarz funktionieren soll. Hierbei muss dann die Geschwindigkeit ebenfalls eingestellt werden. Wird auf einen externen Takt konfiguriert, muss zum weiteren Ansprechen des Chips zwingend ein externer Takt / Quarz angeschaltet sein, weil sonst eine ISP Übertragung nicht mehr möglich ist.

Außerdem verfügen (meines Wissens) alle AVR's über einen internen Taktteiler, der den zur Verfügung stehenden Takt „teilen", also reduzieren kann. Ist ein Teiler durch 8 bspw. aktiv, wird der interne Prozessor mit 1/8 der anliegenden Taktfrequenz betrieben.

Es kann eingestellt werden, ob er einen internen Reset durchführen kann und die eigentliche Resetleitung als Portpin zur Verfügung stehen soll, was mit größter Sorgfalt zu beachten ist, weil nach einer Konfiguration der Resetleitung als Portpin kann der Microcontroller mittels eines ISP-Programmers nicht mehr angesprochen werden.

Es kann sogar eingestellt werden, ob der Controller mittels ISP programmierbar sein soll (auch das ist gefährlich wenn man die ISP Programmierbarkeit deaktiviert, weil es dann besonderer Programmer bedarf um mit dem Chip noch einmal kommunzizieren zu können).

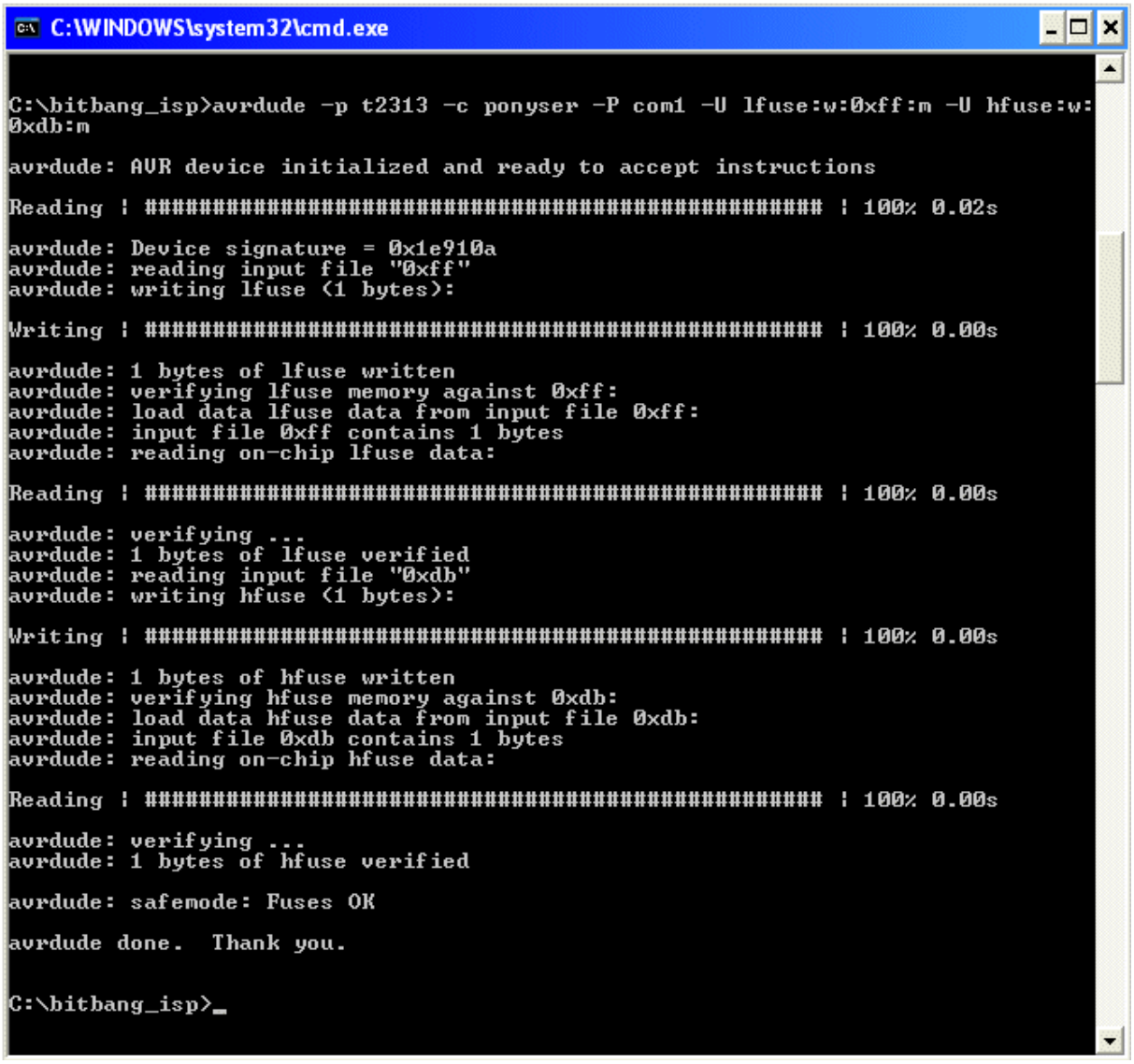
ACHTUNG:

Die folgende Anweisung stellt den ATtiny2313 auf externen Quarz um, es muß zwinend ein Quarz wie im Schaltplan angeschlossen sein:

Um die Fuses für einen Betrieb als USBtiny-Programmer (mit externem 12 MHz Quarz) zu setzen wird am „DOS-Prompt“ folgendes eingegeben:

```
avrdude -p t2313 -c ponyser -P com2 -U lfuse:w:0xff:m -U hfuse:w:0xdb:m
```

Es sollte nun am Bildschirm folgendes zu sehen sein:



Wenn dieser Bildschirm zu sehen ist: Herzlichen Glückwunsch, der erste Microcontroller ist (mit einem USB-Programmer Programm) geflasht worden.

Abschließend zum „Bitwackeln“:

Prinzipiell kann der „Bitwackler“ eine Menge andere Controller als den ATtiny2313 flashen und natürlich kann (sollte) er jedes andere HEX-Programm in den Microcontroller transferieren können.

Getestet habe ich den Bitwackler an COM1 und COM2 (an höheren Schnittstellennummern konnte ich mangels Verfügbarkeit echter RS-232 Schnittstellen nicht testen).

Es wurden getestet:

- ATtiny24
- ATtiny2313
- ATmega8
- ATmega168
- ATmega328
- ATmega644

Dem ATmega328 wurde ein 30 kByte großes Program geuploaded was zu meinem Erstaunen problemlos funktionierte, hatte ich doch erwartet gehabt, dass er sich während des Uploads eventuell „verschluckt“ (was nicht geschehen ist).

Wer nicht auf einen (alten) Windows XP Rechner (mit RS-232 Schnittstelle) angewiesen sein möchte, sollte sich, wenn er sich in die Microcontrollerwelt einarbeiten möchte, wirklich den USBtinyISP Programmer aufbauen. Dieser ist auch an einem Windows 7 / Windows 8 System mit 64 Bit funktionsfähig.