Fuses mit BASCOM und USBASP setzen

Auch wenn mir dieser Programmteil von BASCOM wenig gefällt und es hübschere Programme zum Setzen der AVR Fusebits gibt, sei's drum. Es gibt immer wieder OM's, die danach fragen.

Zugute halten kann man, dass auch weniger gebräuchliche Controller hiermit zu bearbeiten sind, die manch andere Brennprogramme nicht können. Deshalb sei hier als Beispiel der ATmega1284P, wie er im ATU Controller eingesetzt ist, gezeigt.

1 Programmer festlegen

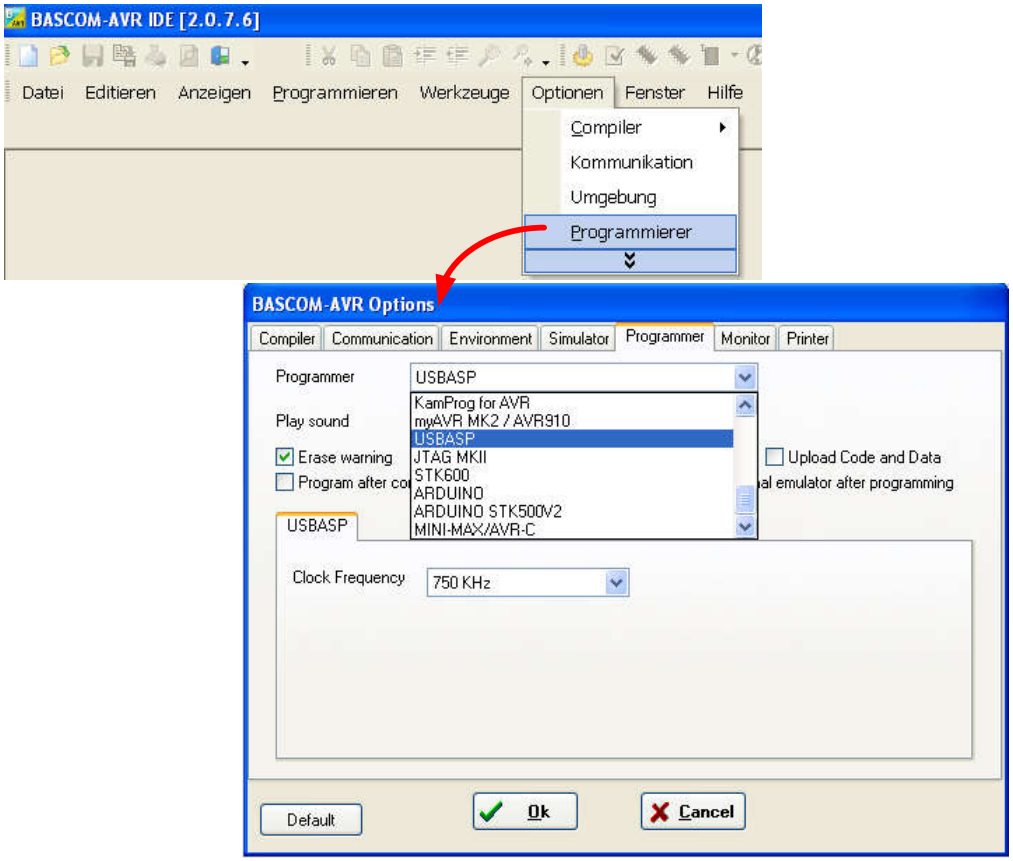

Abb. 1: Auswahl des Programmers, hier USBASP.

Bei einem jungfräulichen Controller ist der interne RC-Oszillator i.d.R. auf 1MHz vorbesetzt. Beim erstmaligen Fuse setzen ist die Clock Frequency auf <1/4, d.h. <250kHz zu reduzieren.

2 Fuses setzen

Der USBASP-Programmer ist an den ISP des Controllers angeschlossen. Der Controller ist eingeschaltet.

Abb. 2: Anwahl des Programmer-Fensters.

Fuses mit BASCOM und USBASP setzen

Im nachfolgenden Programmer-Fensters können wahlweise der Flash-Speicher programmiert, das EEPROM beschrieben oder die Fuses gesetzt werden. Mit Öffnen des Fensters erfolgt ein Zugriff auf den Controller über ISP. Damit wird er identifiziert, hier ATmega1284P.

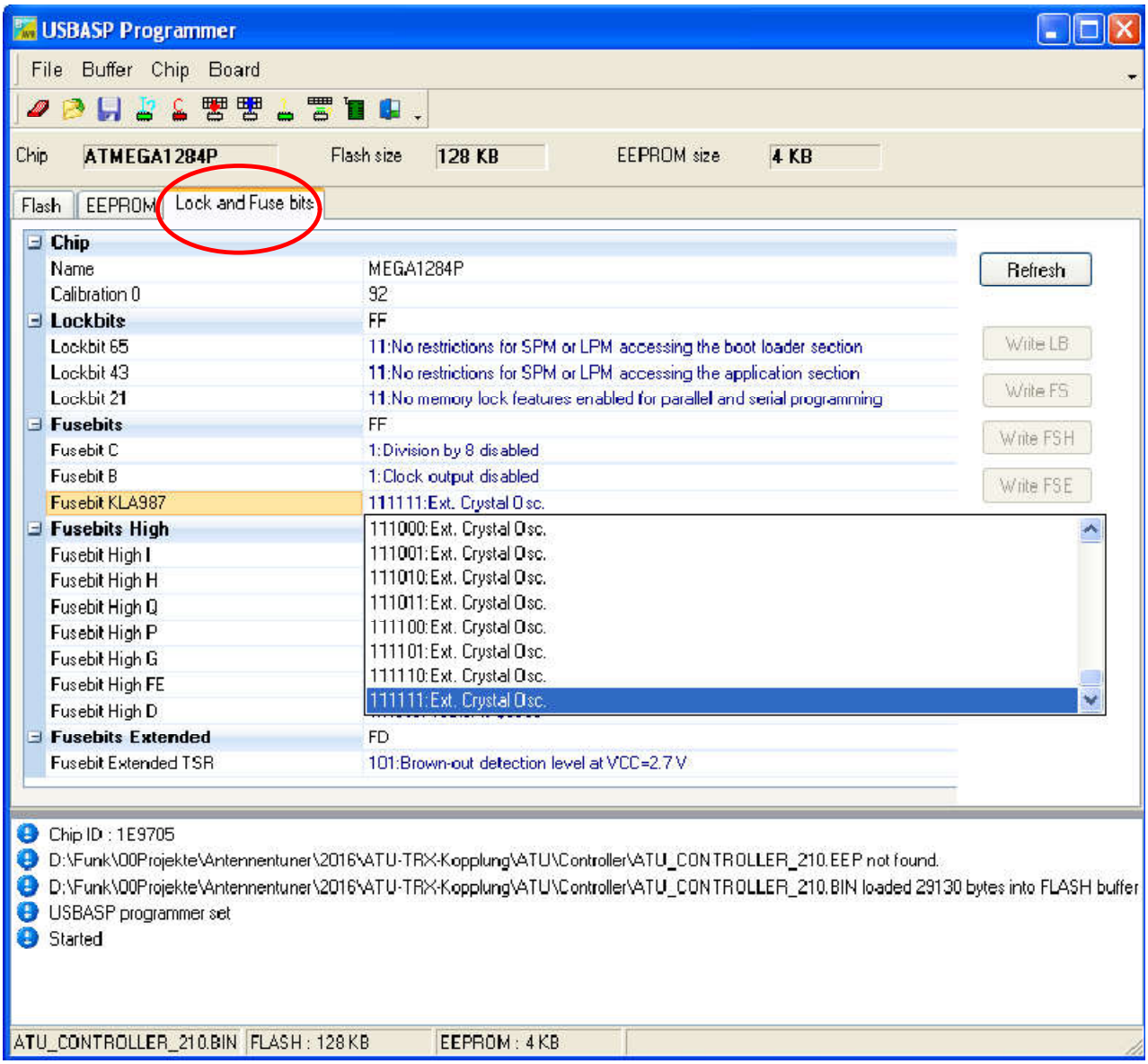

Abb. 3: Programmer, Reiter Lock and Fuse bits.

Zunächst die für BASCOM-Anwendungen uninteressanten Einstellungen (von oben):

Calibration	Kalibrieren des internen RC-Oszillators. Uninteressant, wie verwenden einen Quarz.
Lockbits 65, 43	Beschränkungen für die Assembler Befehle SPM und LPM. Uninteressant, mit Assembler mühen wir uns nicht ab.
Lockbit 21	Speichersperre (Memory Lock). 11. No Memory Lock features enabled…: Programm und EEPROM-Inhalte können geschrieben und ausgelesen werden. Die anderen Optionen sperren den Zugriff auf Flash und EEPROM.

Am besten Finger weg von diesen Einstellungen.

Fuses mit BASCOM und USBASP setzen

In der Regel anzupassen sind die Einstellungen von

- Fusebits (low Byte),
- Fusebits High und
- Fusebits Extended.

Als abschreckendes Beispiel ist in Abb. 3 das Fusebit KLA987 aufgeklappt, das die Taktquelle festlegt:

- interner RC-Oszillator,
- externer Quarz/Resonator (Crystal) oder
- externer eigenständiger Oszillator.

Die sichtbaren identisch angezeigten Einstellungen für einen an XTAL1 und XTAL2 angeschlossenen Quarz haben alle verschiedene Wirkungen. Die 6 gezeigten Ziffern repräsentieren die Bits

SUT	Start up time	SUT1, SUT0	Ziffern 1 und 2
CKSEL	Clock select	CKSEL3…CKSEL0	Ziffern 3 bis 6

Die markierte letzte Einstellung "111111" steht für SUT=11 und CKSEL=1111. Und nun? Da hilft nur ein Blick in das Datenblatt. In dem für den ATmega1284P muss man sich im Kapitel 7, System Clock and Clock options, schlau machen. Dann weiß man nach einigem vor- und zurückblättern, dass

"111111" Low power xtal 8-16MHz, slow rising power 14CK + 65ms (14 Takte plus 65ms)

bedeutet.

Diese Einstellung verwende ich bei den meisten Anwendungen. Benutzerfreundlich ist die magere Anzeige in Abb. 3 nicht. Auf der anderen Seite war bisher jedes BASCOM-Update kostenlos. Da kann man auch mal nachsichtig sein oder aber einen anderen Programmer nehmen.

Aber - ohne die BASCOM-Programmierumgebung verlassen zu müssen - kann man auch mal schnell das eine oder andere Fusebit umprogrammieren, etwa das Fusebit High G (EEPROM erase), um beim nächsten Brennen ein vermurkstes EEPROM zu löschen, also mit 0xFF (255 dezimal) wieder zu initialisieren.

Nicht wundern bei einem Vergleich mit anderen Programmern, etwa mit PonyProg.
Eine "1" bedeutet, dass das Bit nicht programmiert ist, eine "0" entsprechend programmiert. In PonyProg werden programmierte Bits, also die mit Wert 0, mit Häkchen gekennzeichnet.

Nach einer Änderung in einem der drei Blöcke und Klick in eine andere Zeile wird der zugehörige Button

- WRITE FS Fusebits (low Byte)
- WRITE FSH Fusebits High
- WRITE FSE Fusebits Extended

aktiviert. Wenn also ein Block abgearbeitet ist, wird mit Klick darauf die Einstellung des geänderten Fusebytes in den Controller geschrieben.

Vorsicht bei der o.a. SUT+CKSEL-Einstellung (Fusebits KLA987). Mit einer falschen Einstellung, z.B. Externer Oszillator statt Externer Quarz, obwohl ein Quarz an XTAL1 und XTAL2 angeschlossen ist, setzt den Controller Schach Matt. Er hat keinen Takterzeuger mehr und reagiert somit auf nichts. Dann hilft nur noch, an XTAL1 (!) ein TTL-Signal anzulegen. Das kann ein TTL-Quarzoszillator oder ein Rechteckgenerator mit z.B. 10 bis 50 kHz (TTL-Pegel) sein. Damit wäre der Controller erst einmal wieder ansprechbar, um die Fuses richtig zu setzen.

Fuses mit BASCOM und USBASP setzen

Finger weg vom Fusebit High Q (Serial Programming)! Da muss

"0 Serial programming enabled"

stehen. Wird dieses Bit nicht gesetzt - Wert 1 - ist die ISP-Schnittstelle ausgeknipst. Der Controller lässt sich dann nicht mehr über ISP programmieren.

Weiter unten gibt es noch einen Button "WRITE PRG". Damit kann man einen entsprechenden $PROG-Befehl in den Code im Hauptfenster schreiben. Es werden die in den Controller geschriebenen Daten verwendet, vorher also mit den o.a. "WRITE XXX" die gemachten Eingaben in den Controller übertragen und den Cursor im Editorfenster an die gewünschte Stelle setzen. $PROG im Programm setzt gleichzeitig mit dem Flashen des Programms auch die Fuse Bits im Controller. Einzelheiten dazu in der BASCOM-Hilfe zu "BASCOM IDE" - "Program send to chip".

Die für den ATU Controller eingestellten Fuses zeigt nächstes Bild. Das ist mehr oder weniger eine Standardeinstellung, die für die meisten Anwendungen und Controller passt, wenngleich sich Fuse- und Lock-Bits je nach Controller unterscheiden.

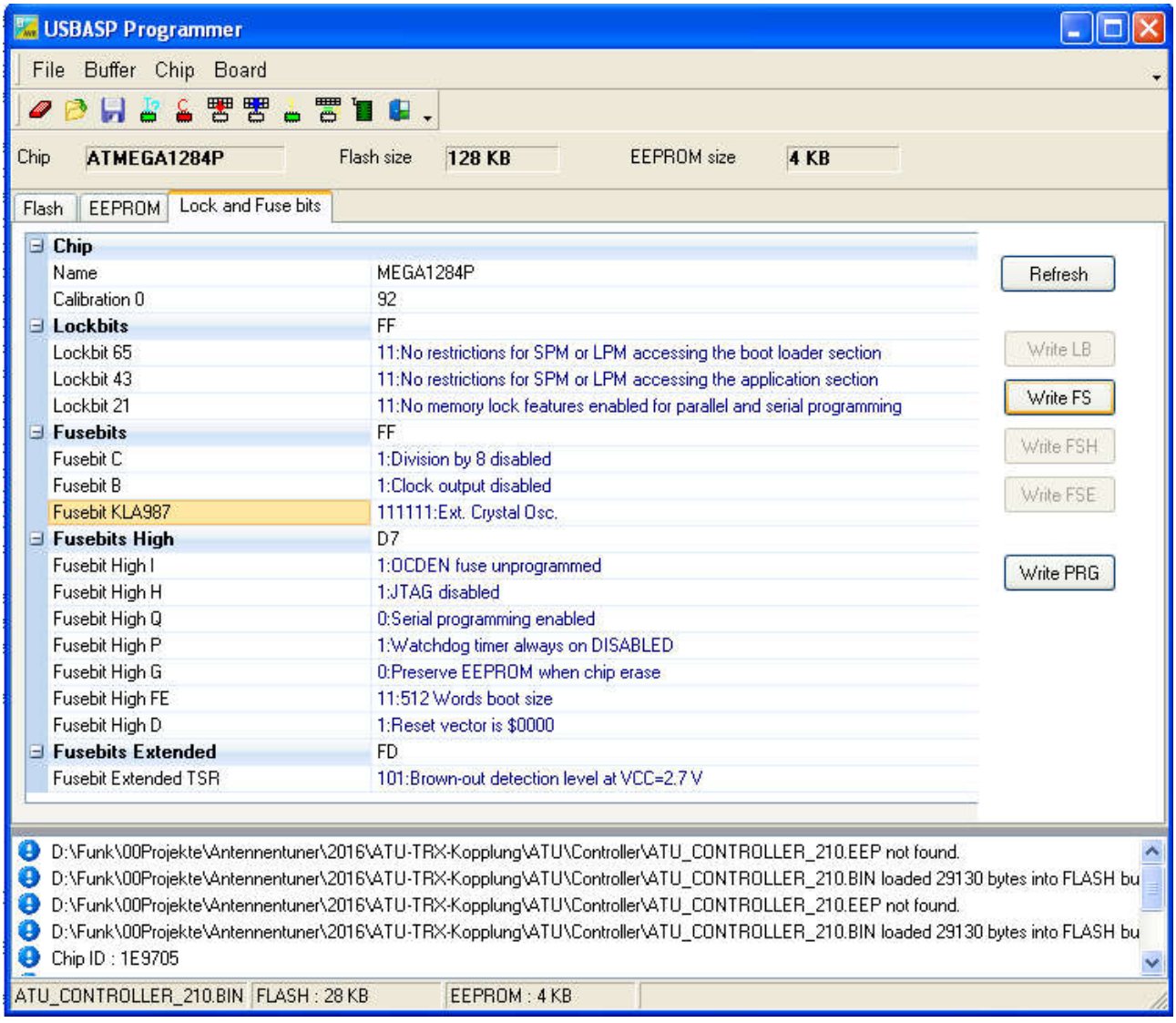

Abb. 4: Fuse-Einstellungen des ATU Controllers.

Fuses mit BASCOM und USBASP setzen

Die sich auf den einzelnen Bits ergebenden Byte-Werte sind bei den einzelnen Fuses als Hex-Werte angegeben, hier

- Lockbits = FF = 1111 1111
- Fuse-Byte low = FF = 1111 1111
- Fuse-Byte high = D7 = 1101 0111
- Fuse Extended = FD = 1111 1101

Beim Flashen etwa mit AVRDUDE können diese Werte zum Setzen der Fuses in der Kommandozeile angegeben werden.

Zur Unterstützung ein Online Fuse Calculator gefällig? Zum Beispiel hier:

http://www.engbedded.com/fusecalc/