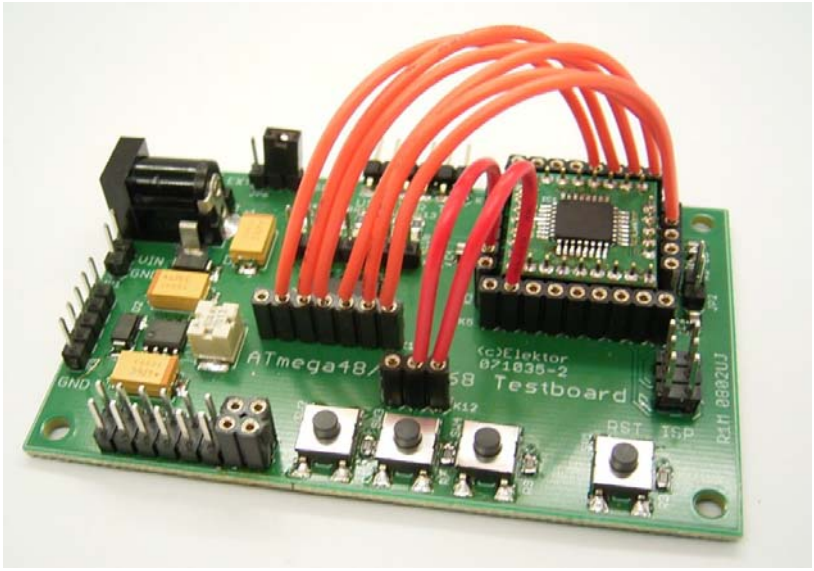
Aller Anfang ist schwer

Erste Schritte mit dem ATM18

Aller Anfang ist schwer, und Tücken lauern überall. Wer zum ersten Mal mit einem AVR-Controller oder überhaupt das erste Mal mit einem Mikrocontroller arbeitet, muss einiges beachten. Am Beispiel des ATM18 sollen die wichtigsten Dinge gezeigt werden.



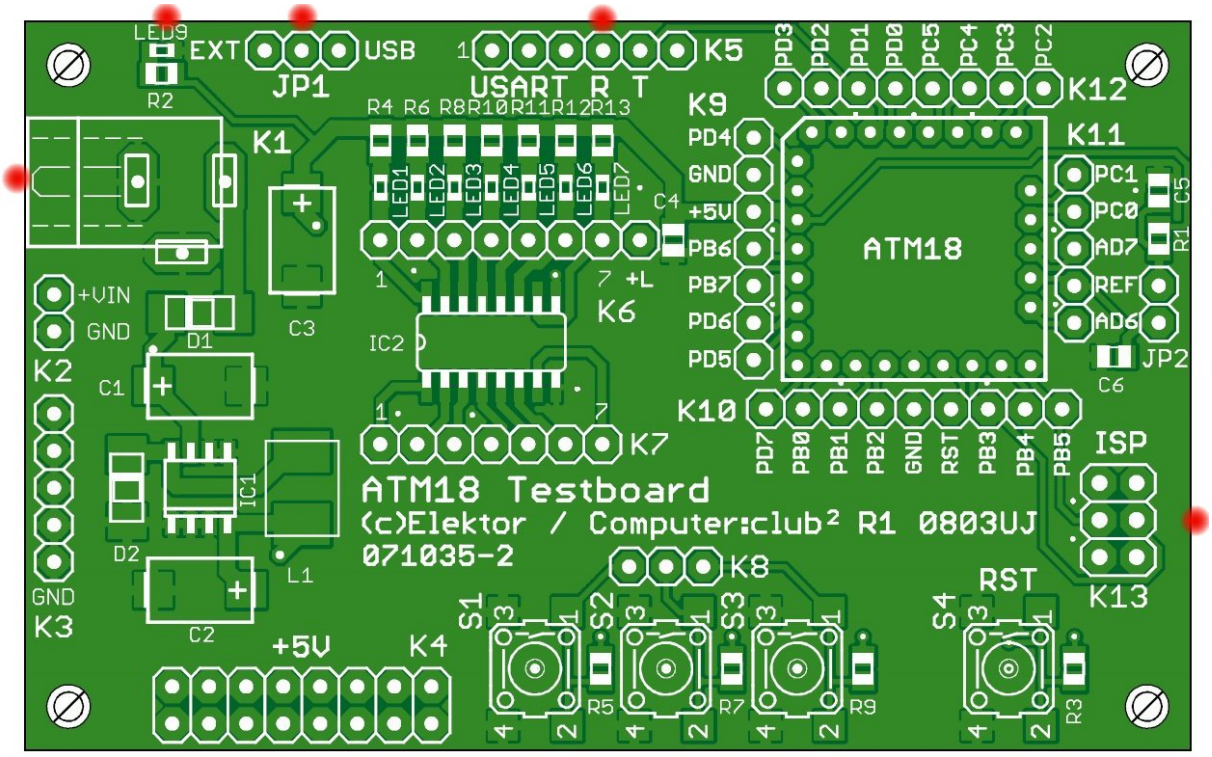
Schritt 1: Betriebsspannung überprüfen

Die grüne LED (LED9) auf dem Testboard zeigt an, ob die Betriebsspannung +5 V vorhanden ist. Wenn die LED nicht leuchtet, ist alles Weitere zwecklos. Es gibt drei mögliche Wege, über die das ATM18-Testboard Energie erhält:

1. Ein Netzteil mit 9 V bis 12 V Gleichspannung und mindestens 100 mA Belastbarkeit (also praktische jedes Steckernetzteil) wird an K1 angeschlossen und versorgt den Schaltregler IC1. Dazu muss aber auch der Jumper JP1 in Position EXT gesteckt werden. Bei manchen Steckernetzteilen muss man auf die Polung achten, sonst bleibt die grüne LED dunkel.

2. Wenn der Jumper JP1 in Position USB gesteckt ist, kommt die Stromversorgung über den USART-Anschluss K5. Hier sollte dann zum Beispiel der USB/Seriell-Wandler von FTDI (siehe separater Artikel in dieser Ausgabe) angeschlossen sein, der nicht nur die seriellen Signale weitergibt, sondern auch +5 V vom USB liefert.

3. Viele Programmiergeräte liefern die Betriebsspannung für das Zielsystem gleich mit. Das gilt zum Beispiel für das STK500 von Atmel, das sogar eine einstellbare Spannung bietet. In dem Fall kommt der Saft also über den ISP-Stecker K13 und JP1 kann offen bleiben. Der USB-AVR-Programmer, den viele Anwender gleich mitbestellt haben, lässt dagegen den VCC-Anschluss offen und schützt damit den PC auch im Falle eines Fehlers.



Was man möglichst vermeiden sollte, ist die gleichzeitige Versorgung aus zwei verschiedenen Quellen. Die übliche Regel „doppelt gemoppelt hält besser“ gilt hier nicht. Meist passiert zwar nichts, aber es kann auch unangenehme Überraschungen geben. So wurde zum Beispiel einmal versehentlich die Versorgung über den ISP-Stecker verwendet und gleichzeitig das FTDI-USB-Seriell-Konverterkabel angeschlossen. Das Ergebnis war, dass der USB-Anschluss des PCs, der normalerweise 5 V liefert, von außen (also quasi rückwärts) mit Strom versorgt wurde. Als dann der PC ganz ausgeschaltet wurde, lag immer noch Spannung am USB und über geheime Wege im PC auch an der Tastatur und der Maus. Beim nächsten Hochfahren funktionierten dann beide nicht, weil ein ordentlicher Reset fehlte. Zum Glück ist kein dauerhafter Schaden entstanden, aber man weiß nie, wie ein bestimmter PC sich in so

einem Fall verhält. Der USB-AVR-Programmer schließt diesen Fehler sicher aus, eben weil er VCC offen lässt und so eine andere Stromversorgung des ATM18-Testboards erzwingt.

Schritt 2: Schwingt der Quarz?

Der erste Test an einem Mikrocontroller-System ist fast schon im Erbgut jedes Elektronikers verankert: Man hält die Tastspitze des Oszilloskops an den Quarz und schaut nach, ob er schwingt.
Aber wenn ein ATmega noch ganz jungfräulich ist, dann lässt er angeschlossene Quarze eben nicht schwingen! Es wäre völlig falsch, daraus den Schluss zu ziehen, dass ein Fehler vorliegt. Alle ATmegas besitzen ja auch noch einen internen RC-Oszillator, und dieser ist werksseitig eingeschaltet. Im Auslieferungszustand ist der interne RC-Oszillator auf 1 MHz eingestellt. Wenn man das ändern will, muss man die „Fuses“ umprogrammieren. Was dabei zu beachten ist steht weiter unten im Tipp Nr. 6.

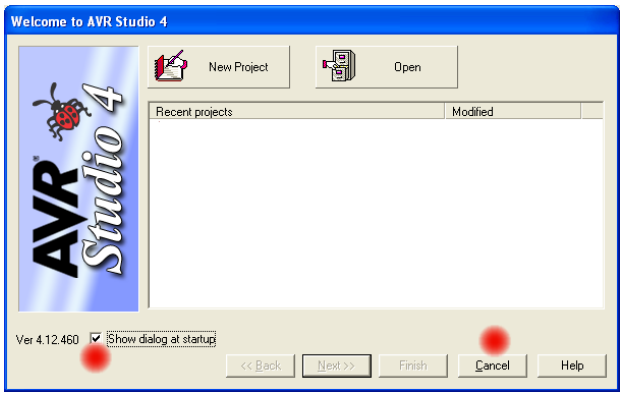
Schritt 3: AVR-Studio installieren

Wer ganz neu anfängt, muss sich erst einmal die passende Software besorgen. Am besten verwendet man AVR-Studio von ATMEL. Gehen Sie auf die Seite http://www.atmel.com/ und geben Sie dort unter Search das Suchwort „AVR Studio“ ein. So kommen Sie am schnellsten zur gesuchten Software AVR Studio 4. Beim Download ist eine Registrierung nötig, aber das geht ganz formlos, Atmel will nur wissen, wer diese schönen Sachen verwendet. Das ganze Studio hat übrigens mehr als 80 MB und enthält wesentlich mehr als man üblicherweise braucht.
Bei der Installation wird gefragt, ob auch der USB-Treiber installiert werden soll. Das sollte man tun, denn es handelt sich um den Treiber für den Atmel-Brenner AVRISP mkII, zu dem der USB-AVR-Programmer von ELEKTOR kompatibel ist. Wenn dann der USB-Brenner an einen USB-Port gestöpselt wird, findet Windows XP den Treiber (normalerweise) automatisch.

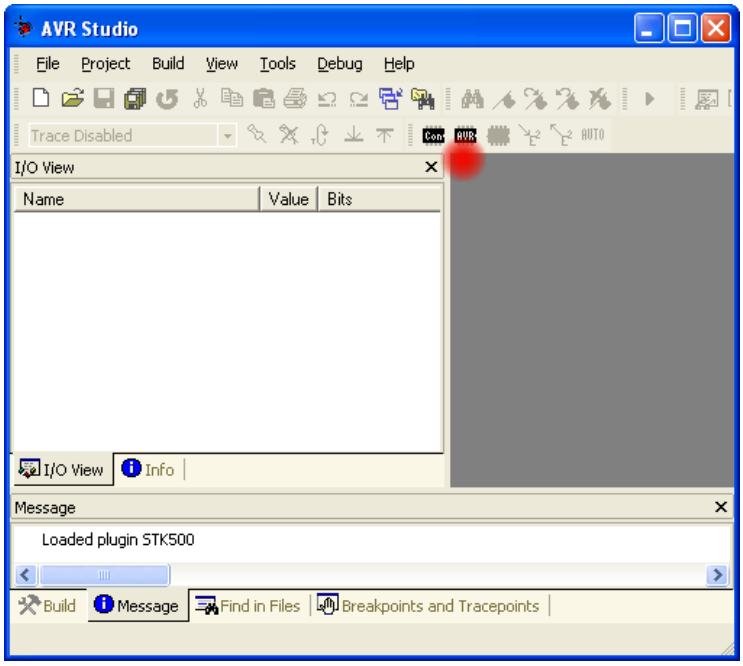
Schritt 4: Starten von AVR-Studio

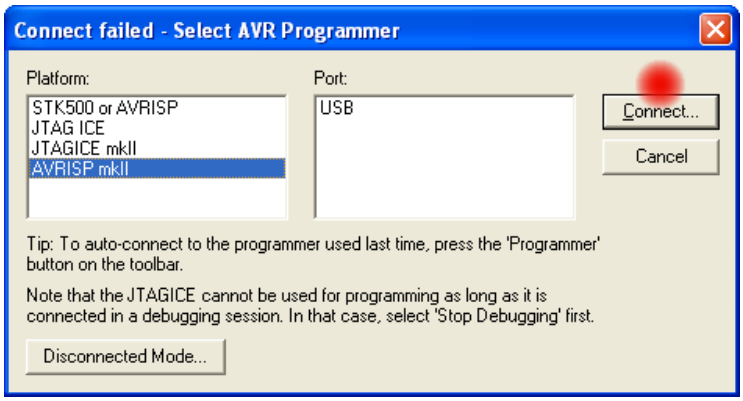
Es gibt viele Möglichkeiten, wie das Hex-File in den Controller gelangt. Einige Compiler bieten die Möglichkeit, die Programmiersoftware direkt aufzurufen. Das kann zwar auf die Dauer bequemer sein, bringt jedoch oft zusätzliche Probleme mit der korrekten Einstellung. Für den Anfang ist es deshalb sicherer, das AVR-Studio zu verwenden. Man hat dann bei der Programmentwicklung immer zwei Programme offen, den Compiler und die Brennersoftware.

Wenn man das AVR -Studio zum ersten Mal startet, kommt erst einmal die Frage, ob man ein neues Projekt erzeugen will. Gemeint ist ein neues Software-Projekt, zum Beispiel in Assembler. Aber was tatsächlich gebraucht wird, ist erst einmal nur die Brenner-Software, und für die braucht man kein Projekt. Man muss dieses Fenster also schließen, damit es weiter geht. Das Häkchen neben „Show dialog as startup“ wird am besten gleich entfernt, damit das Fenster beim nächsten Start nicht mehr nervt.



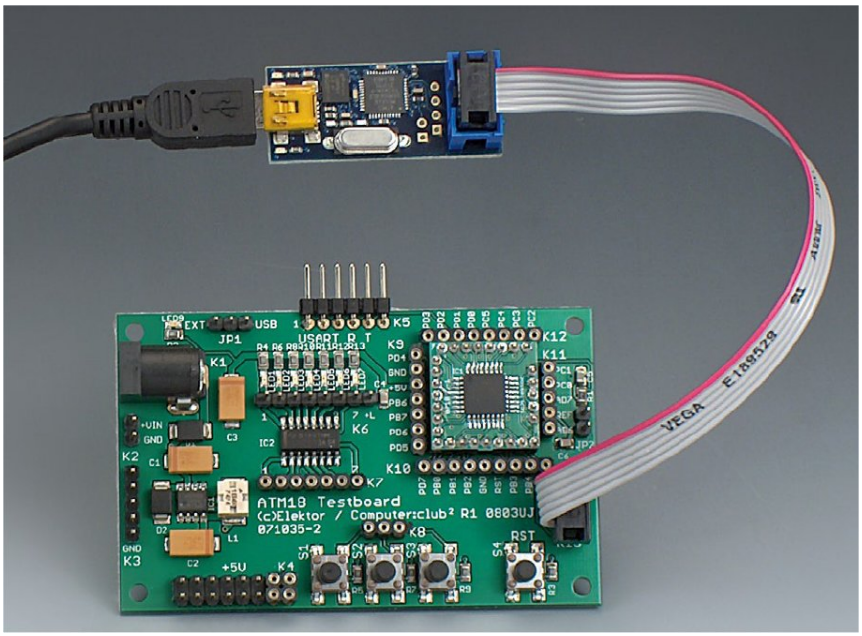
Den Brenner findet man nun über Tools/Program AVR oder über das kleine schwarze IC-Symbol mit der Aufschrift AVR. Eventuell muss man dann erst einmal den verwendeten Brenner auswählen. Mit „AVRISP mkII“ hat man zugleich auch die richtige Einstellung für den USB-AVR-Programmer.



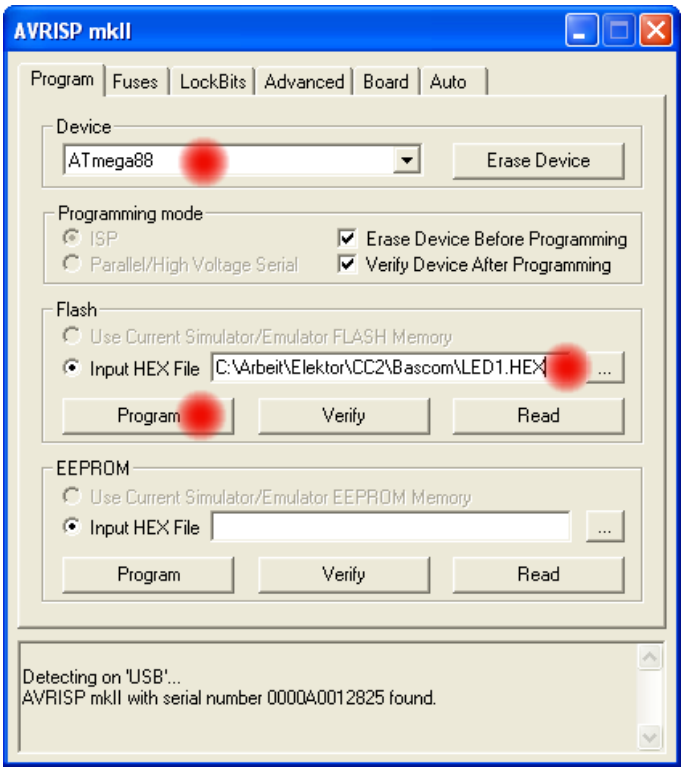


Schritt 5: Kontaktaufnahme mit dem ATmega

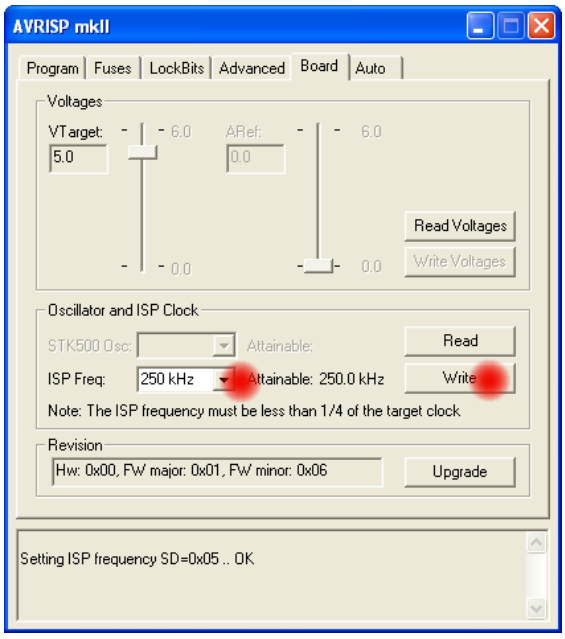
Die Verwendung des AVR-USB-Programmers wurde bereits in ELEKTOR 05/08 genau beschrieben. Verbinden Sie das sechspolige ISP-Kabel wie dort gezeigt mit dem Testboard. Den Stecker kann man auch falsch herum aufsetzen, richtig ist es, wenn das Flachbandkabel von der Mitte der Platine aus angesteckt wird. Außerdem muss natürlich Spannung anliegen, was man an der grünen LED sieht (siehe Schritt 1).



Dann muss noch der richtige Controller ATmega88 ausgewählt und ein Hex-File zur Programmierung in den Flash-Speicher des Systems ausgewählt werden. Wählen Sie zum Beispiel das File LED1.HEX aus dem ersten Bascom-Beispiel zum ATM18. Wenn Sie dann auf „Program“ klicken, sollte alles ganz schnell übertragen werden. Im unteren Fenster erscheinen die Erfolgsmeldungen. Bei richtiger Verdrahtung der passenden Ports sieht man dann das LED-Lauflicht bei der Arbeit.

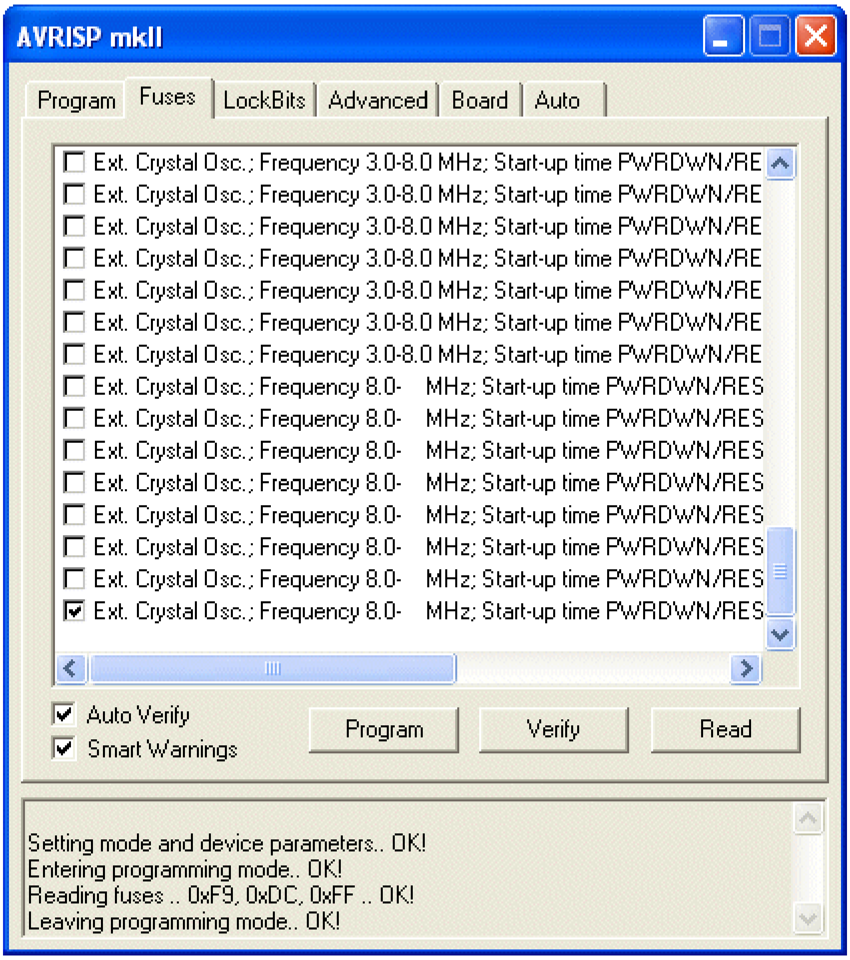


Wenn es Probleme gibt, liegt das meist an einer zu hohen ISP-Taktrate. Diese kann unter dem Reiter „Board“ eingestellt werden. Für den Anfang sollte man nicht über 250 kHz gehen, weil der Controller noch mit 1 MHz arbeitet.

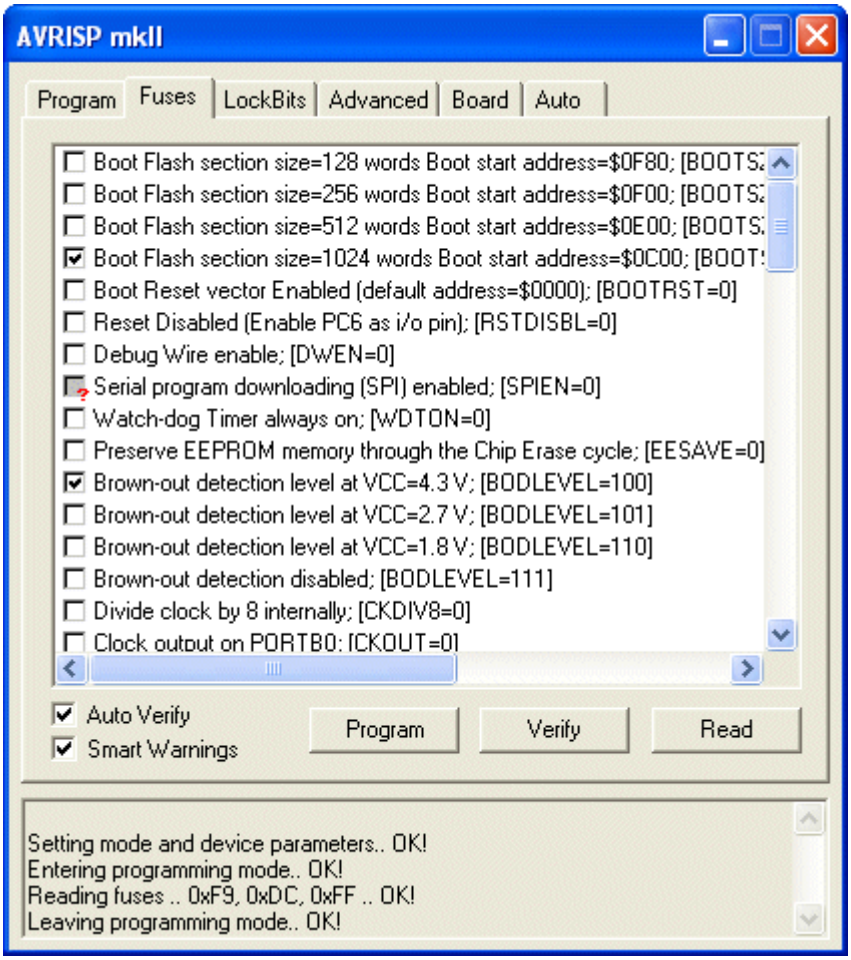


Schritt 6: Fuses ändern

Der ATmega88 besitzt zahlreiche Einstellungen, die über die Fuses erfolgen. Die korrekten Fuses sieht man im Startartikel in Heft 4/08 im Bild 10. Die wichtigste Einstellung betrifft den Oszillator. Wählen Sie den „Ext. Crystal Osc. Freq 8.0- MHz“, um den 16-MHz-Quarz auf der Platine zu verwenden. Die Startup-Time spielt keine wichtige Rolle, verwenden Sie einfach die längste (unterste) Einstellung.



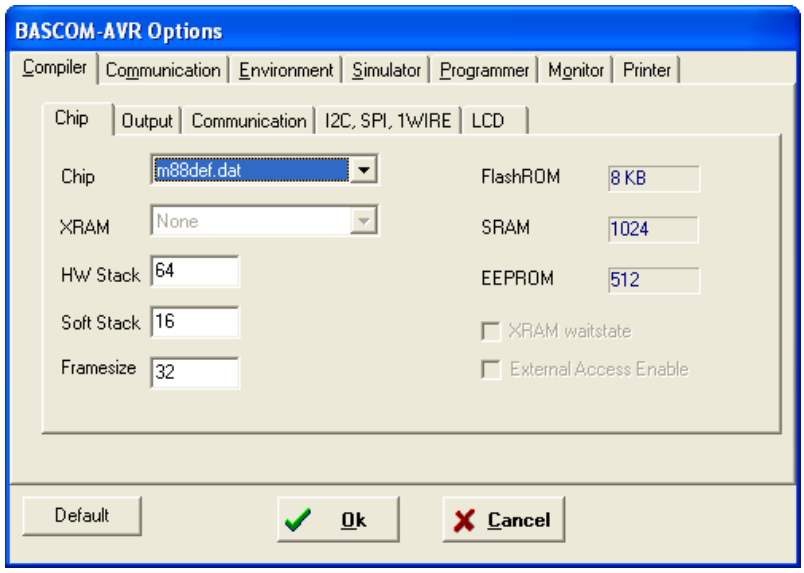
Möglich sind auch Einstellungen des internen RC-Oszillators mit 1 MHz, 2 MHz, 4 MHz und 8 MHz. Damit hat man jeweils eine entsprechend geringere Arbeitsgeschwindigkeit, aber auch eine geringere Frequenzgenauigkeit. Vermeiden Sie unbedingt die Einstellungen Externer Oszillator (Ext Clock) und Externer RC-Oszillator (Ext. RC Osc.), denn damit wäre der ATmega erstmal tot. Das Programmieren der Fuses geht gerade noch, danach aber nichts mehr, weil der Controller dann ganz ohne Taktsignal dasteht, das er auch für die Programmierung braucht. Wenn das doch einmal passiert, muss man einen externen Takt am Pin XTAL1 zuführen, um den Kontakt zum Programmer wiederherzustellen und die Einstellung zu korrigieren.



Noch schlimmer ist es, wenn Sie den Reset-Pin versehentlich als Port umwidmen (Reset Disabled). Dann hat man zwar einen Port (PC6) mehr, aber eine weitere Programmierung ist nicht mehr möglich! Eine sinnvolle Anwendung gäbe es nur dann, wenn der Port PC6 unbedingt gebraucht wird und wenn man sicher ist, dass das Programm nie wieder geändert werden muss.

Schritt 7: BASCOM verwenden

Laden Sie sich die freie Version von BASCOM-AVR von MCS electronics (www.mcselec.com). Auf der Elektor-Webseite www.elektor.de/071035 finden Sie einen direkten Link zum Download. Nach der Installation kann man ein neues Projekt beginnen. Die wichtigste Einstellung ist die Wahl des korrekten Controllers.



Bevor man aber richtig loslegt, sollte man sich mit der Hilfedatei vertraut machen, denn hier sieht man am schnellsten, welche Möglichkeiten das Programm bietet. Außerdem lohnt es sich, die mitgelieferten Beispielprojekte anzusehen, auch wenn sie für andere AVR-Typen geschrieben wurden.

Schritt 8: Ein vorhandenes Programm ändern

Der einfachste und sicherste Weg zu lernen besteht darin, ein vorhandenes Projekt in kleinen Schritten zu verändern. Auch das kleinste Programm bietet nämlich viele mögliche Fehlerquellen. Und wer einmal über mehrere Fehler gleichzeitig gestolpert ist, der weiß, wie lange man sich bei der Fehlersuche im Kreis drehen kann. Geht man aber von einem getesteten Programm aus, kann man sich mit einzelnen kleinen Änderungen vorantasten. Wenn dann plötzlich nichts mehr geht, muss nur die letzte Änderung genauer untersucht werden.

Laden Sie zum Beispiel das erste Bascom-Beispiel (LED1.bas) aus Heft 4/08. Kompilieren Sie es zuerst einmal ohne Änderungen. Es muss ein neues Hexfile mit dem aktuellen Datum im Projektverzeichnis zu finden sein. Dieses laden Sie dann mit dem AVR-Studio in den Controller. Es sollte dann genauso arbeiten wie das fertige, von der Elektor-Seite geladene Hex-File.

Ändern Sie dann den Quelltext an einer leicht verständlichen Stelle, verkürzen Sie zum Beispiel die Wartezeit (Waitms 20 statt Waitms 100). Das Lauflicht sollte dann deutlich schneller laufen. Als nächstes können Sie das Programm für andere Portanschlüsse umschreiben oder zum Beispiel so erweitern, dass die LEDs abwechselnd hin und zurück laufen.

Leisten Sie sich absichtlich einmal einen Tippfehler. Beim Kompilieren entsteht dann eine Fehlermeldung. Klicken Sie auf die Fehlermeldung, dann wird genau die fehlerhafte Zeile im Quelltext markiert.

Schritt 9: Ein eigenes, ganz neues Programm

Wenn Sie eine ganz neue Aufgabe beginnen wollen, bietet es sich an, trotzdem erst einmal das Gerüst eines vorhandenen Programms zu verwenden.

Dann stimmt schon mal der Prozessor und die Quarzfrequenz. Als nächstes ist es sinnvoll, Teile aus den Beispielen der Hilfe in das eigene Programm zu kopieren. Wenn Sie zum Beispiel die serielle Schnittstelle verwenden oder einen Timer für PWM-Ausgaben einsetzen wollen, finden Sie die entscheidenden Hinweise in der Hilfe.
Bei der Verwendung der seriellen Schnittstelle beachten Sie bitte, dass das Testboard keinen Pegelwandler besitzt. Am USART-Anschluss K5 liegen also TTL-Signale. Diese passen zum eingangs erwähnten und in diesem Heft beschriebenen USB-Seriell-Kabel von FTDI. Wenn Ihr Computer eine serielle Schnittstelle hat, können Sie auch einfach das IC MAX232 (extern) verwenden, um auf normale RS232-Pegel für die Verbindung zwischen USART und PC zu kommen. Eine weitere Möglichkeit, vor allem wenn nur in Richtung PC gesendet werden soll, ist der Einsatz des Software-UART in Bascom. Mit diesem können Sie die Polarität wählen, sodass eine direkte Verbindung zur RXD-Leitung des PC möglich wird.

Schritt 10: Umgang mit Ports

Der Umgang mit den Ports ist zwar nicht besonders schwierig, aber man kann trotzdem sehr leicht kleine Fehler machen. Wenn zum Beispiel ein Port für Ausgaben benutzt werden soll, muss er vorher in Ausgaberichtung umgeschaltet werden (Config Portc = Output). Für Eingaben kann es sinnvoll sein, den Port zwar in Leserichtung zu belassen, aber die internen Pull-up-Widerstände einzuschalten, indem man Einsen in das zugehörige Portregister schreibt (Portb = 255).

```
'Time of reaction measured on PB0, output at 6 LEDs on PortC

$regfile = "m88def.dat"
$crystal = 16000000

Config Portc = Output
Config Portb = Input
Portb = 255              'Pullups
Config Portd = Input
Portd = 255              'Pullups

Dim Leds As Byte
Dim Timeout As Word
Leds = 1

Do
  Portc = 255
  Timeout = Rnd(1000)
  Timeout = Timeout + 500
  Waitms Timeout
  Portc = 0
  Timeout = 0
  Do
    Timeout = Timeout + 1
    Waitms 10
  Loop Until Pinb.0 = 0 Or Timeout = 127
  Leds = Timeout
  Portc = Leds
  Waitms 1000
Loop
```

Nach einem Reset oder Neustart sind alle Ports zunächst Eingänge ohne Pullups. Sie verhalten sich also wie hochohmige CMOS-Eingänge. Mit einem Oszilloskop erkennt man diesen Zustand leicht, wenn man gleichzeitig mit der Messspitze auch seinen Finger anschließt. Dann sieht man nämlich deutlich ein Netzbrummen, das durch einen niederohmigen Portpin unterdrückt würde. Offene Eingänge können ein Risiko für die Software darstellen, wenn man Ports abfragt und dabei zufällige Zustände liest.
Beim Lesen von Portzuständen muss das Leseregister PINx abgefragt werden, nicht aber das Ausgaberegister PORTx. Nicht nur einmal ist es vorgekommen, dass sich jemand gewundert hat, wenn er ein High-Signal an einem Portpin messen konnte, das zugehörige Portregister aber Nullen enthielt. Liest man zum Beispiel den Inhalt von PORTB, dann findet man hier die zuletzt ausgegebenen Bits. Liest man dagegen PINB, dann erhält man die tatsächlichen Zustände an den Pins.

Besondere Aufmerksamkeit brauchen die Ports PB3 (MOSI), PB4 (MISO) und PB5 (SCK), die zugleich die ISP-Schnittstelle darstellen. Wenn kein Programmer angeschlossen ist, hat man hier ganz normale Ports. Während der Entwicklung muss man jedoch beachten, dass der Programmer parallel angeschlossen ist. Offene Eingänge sind dann nicht ganz so hochohmig. Außerdem können von außen angeschlossene Ausgänge oder niederohmige Belastungen (<1 kΩ) den Programmiervorgang stören.

Tipp am Schluss:

Dieses PDF wird bei Bedarf aktualisiert. Hinweise darauf und weitere Tipps und aktuelle Informationen finden Sie auf der Elektor-Website im Forum:

www.elektor.de/forum

Dort wurde ein eigenes CC2-Elektor-AVR-Projektforum eingerichtet, das von unserem Moderator (Kürzel: Tip) mit Unterstützung der ATM18-Entwickler und Autoren betreut wird.

Für den direkten Kontakt mit der Elektor-Redaktion verwenden Sie bitte das Kontaktformular auf der Elektor-Website:

www.elektor.de/kontakt

Wählen Sie als Betreff bitte „Redaktion“ oder „Technische Frage“ aus.

Ihre Elektor-Redaktion