AVR-8-bit-Mikrocontroller
Gruppe 200 - Einsetzen von AVR-Tools
Teil 202 - ISP-Programmieradapter

Teil 201 - Experimentierboards

Teil 202 - ISP-Programmieradapter

Teil 203 - AVR-ALE-Testboard

Teil 204 - AVR Studio

Teil 205 - Assembler und AVR Studio

Teil 206 - C-Compiler und AVR Studio

Teil 207 - Editor - UltraEdit

Hinweis

Externe Anschaltungen und Hardware-Erweiterungen werden in der **Gruppe 400 - ASM-Projekte** und in der **Gruppe 600 - C-Projekte** detailliert beschrieben.

AVR-8-bit-Mikrocontroller
Gruppe 200 - Einsetzen von AVR-Tools
Teil 202 - ISP-Programmieradapter

Vorbemerkung

Nichts ist vollkommen - und nichts ist endgültig! So auch nicht dieses Tutorial! Deshalb bitte immer erst nach dem neuesten Datum schauen. Vielleicht gibt es wieder etwas Neues oder eine Fehlerbereinigung oder eine etwas bessere Erklärung. Wer Fehler findet oder Verbesserungen vorzuschlagen hat, bitte melden (**info@alenck.de**).

Immer nach dem Motto: Das Bessere ist Feind des Guten und nichts ist so gut, dass es nicht noch verbessert werden könnte.

Bild-, Beispiel-, Form- und Tabellen-Nummern sind nach folgendem Schema aufgebaut, damit bei Einfügungen/Löschungen nicht alle Nummern wieder geändert werden müssen (hier bunt dargestellt):

Darstellungsart Abschnitt-LfdNummer: Beschreibung	**allgemeines Schema**
• **Bild 5.1.4-02: Daten-Adress-Raum**	Benummerung eines Bildes
• **Beispiel 5.1.4-03: EEPROM-Speicherung**	Benummerung eines Beispiels
• **Form 5.1.3-01: Die `main`-Funktion**	Benummerung einer Formdarstellung
• **Tabelle 5.1.4-01: Schlüsselwörter vom CVAVR**	Benummerung einer Tabelle

Gravierende Änderungen gegenüber der Vorversion

1.
Völlig neue Strukturierung in **Gruppen** und **Teile**, um das Tutorial umfassend ordnen zu können. Die **Abschnitte** in den **Teilen** sind weitgehend erhalten geblieben.

Gruppenbezeichnung	Kurzbezeichnung
Gruppe 100: Technologie der AVR-8-Bit-Mikrocontroller	**Technologie**
Gruppe 200: Einsetzen von AVR-Tools	**Tools**
Gruppe 300: Arbeiten mit AVR-Assembler **3xx_Programm_yyyyy**	**ASM-Programmierung** **ASM-Programm-Beispiel**
Gruppe 400: AVR-ASM-Projekte **4xx_Projekt_yyyyy**	**ASM-Projekte** **ASM-Projekt-Bezeichnung**
Gruppe 500: CodeVisionAVR C-Compiler **5xx_Programm_yyyyy**	**C-Programmierung** **C-Programm-Beispiel**
Gruppe 600: AVR-C-Projekte **6xx_Projekt_yyyyy**	**C-Projekte** **C-Projekt-Bezeichnung**

xx steht für die laufende Nummer innerhalb des **Teils**, in dem das Programm bzw. das Projekt erscheint und **yyyyy** steht für die Programm- bzw. Projekt-Kurz-Bezeichnung.

2.
Notwendige Änderungen auf Grund Neuinstallation von **Windows 7**.

3.
Windows 7 machte eine Installation von **CodeVisionAVR V2.60** als Vollversion notwendig. Daraus leiten sich auch viele Änderungen im Detail für die C-Programmierung (**Gruppe 500**) ab.

4.
Neu-Installation von **AVR Studio Vers. 4.19** unter **Windows 7**

5.
Zur Demonstration des Tools **AVR Studio** ist in **Gruppe 200** eine Trennung in **Teil 205 - Assembler und AVR Studio** und **Teil 206 - C-Compiler und AVR Studio** vorgenommen worden.

6.
ASM- und **C-Projekte** werden jeweils in eigenen Gruppen gesammelt (**Gruppe 400** für Assembler- und **Gruppe 600** für C-Projekte).

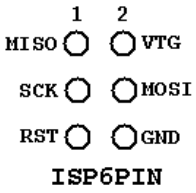
2 ISP-Programmieradapter

Der ISP-Programmieradapter ist eine Schaltung, die die PC-Schnittstelle (parallel, seriell und/oder USB) und die ISP-Schnittstelle zum Experimentierboard bzw. AVR-Anwendungssystem verbindet. Eine Übersicht über mögliche Varianten an ISP-Programmieradaptern findet man hier bei www.mikrocontroller.net.

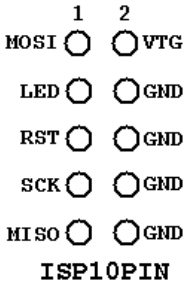
Für das Flashen sorgt ein in allen AVR-Chips eingebautes Interface, über das der Inhalt des Flash-Speichers sowie des eingebauten EEPROM's beschrieben und gelesen werden kann. Das Interface arbeitet seriell und braucht grundsätzlich drei Leitungen:

- **SCK**: Ein Taktsignal, das die zu schreibenden Bits in ein Schieberegister im AVR eintaktet (**S**erial **C**lo**k**) und zu lesende Bits aus einem weiteren Schieberegister austaktet,
- **MOSI**: Befehle und Datenbits vom ISP-Adapter zum Mikrocontroller (Experimentierboard) und
- **MISO**: Datenbits vom Mikrocontroller (Experimentierboard) zum ISP-Adapter.

Da die drei Pins "von Haus aus" zur Steuerung von Anwendungen genutzt werden - sie werden also nicht nur zur Programmierung verwendet - wechseln sie nur dann in den Programmiermodus, wenn das Reset-Signal am AVR (auch: **RST** oder Restart genannt) auf logisch **Null** liegt. Ist das nicht der Fall, so dienen die drei Pins als Ein-/Ausgabe-Ports (z.B. beim ATmega88: **PB3**, **PB4** und **PB5**). Wer die drei Pins mit dieser Doppelbedeutung benutzen möchte und das Programmieren des AVR in der Schaltung selbst vornehmen möchte, muss z.B. einen Multiplexer verwenden oder Schaltung und Programmieranschluss durch Widerstände voneinander entkoppeln. Was nötig ist, richtet sich nach dem, was die wilden Programmierimpulse mit dem Rest der Schaltung anstellen können.



Nicht notwendig, aber bequem ist es, die Versorgungsspannung für das Experimentierboard und den ISP-Programmieradapter gemeinsam vorzusehen und dafür zwei weitere Leitungen einzuplanen. **GND** versteht sich von selbst, **VTG** bedeutet **V**oltage **T**ar**g**et und ist die Betriebsspannung des Zielsystems. Damit wären wir bei der 6-Draht-ISP-Programmierleitung. Die **ISP6PIN-**Verbinder haben die nebenstehende, von ATMEL standardisierte Pinbelegung.



Und wie das so ist mit Standards: Immer gab es schon welche, die früher da waren, die alle verwenden und an die sich immer noch (fast) alle halten. Hier ist der Standard des 10-poligen Steckverbinders. Er hat noch zusätzlich einen **LED**-Anschluss, über den die Programmiersoftware mitteilen kann, dass sie mit dem Programmieren beschäftigt ist. Auch nicht schlecht, mit einer roten LED über einen Widerstand gegen die Versorgungsspannung ein deutliches Zeichen dafür zu setzen, dass die Programmiersoftware ihren Dienst versieht.

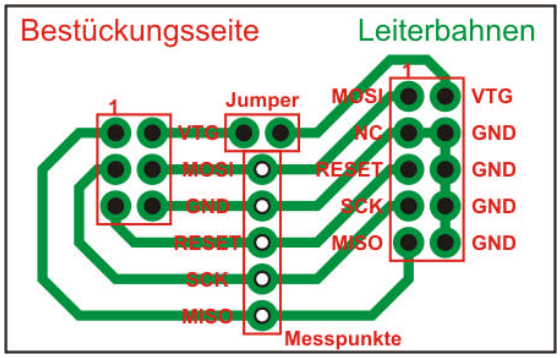


Bild 2-01: ISP-Schnittstellen-Adapter von ISP6PIN auf ISP10PIN und umgekehrt

(Bildvergrößerung)

Tabelle 2-01: Pin-Beschreibung des ISP

Signal	Name	6-Pin	10-Pin	I/O	Beschreibung
MISO	**M**aster **I**n - **S**lave **O**ut	1	9	ISP-Input	Kommunikationsverbindung vom zu programmierenden AVR-System (Slave) zum In-System-Programmer (Master).
VTG	**V**oltage **T**ar**G**et Power	2	2	ISP-Input **oder** Output	Um die Programmierung von AVR-Systemen unterschiedlicher Spannungen zu ermöglichen, kann der In-System-Programmer über diesen Pin auch die jeweilige Spannung vom AVR-System übernehmen. Alternativ kann das zu programmierende AVR-System ohne eigene Stromversorgung auch vom In-System-Programmer programmiert werden, wenn dieses eine eigene Stromversorgung besitzt.
SCK	**S**erial **C**loc**K**	3	7	ISP-Output	Die Frequenz des Programmier-Taktes wird vom In-System Programmer (Master) erzeugt.
MOSI	**M**aster **O**ut - **S**lave **I**n	4	1	ISP-Output	Kommunikationsverbindung vom In-System Programmer (Master) zum AVR-System (Slave), welches programmiert werden soll.
RESET	Target AVR **M**icro**C**ontroller**U**nit **RESET**	5	5	ISP-Output	Um die In-System Programmierung zu ermöglichen, muss der Reset des zu programmierenden AVR-Systems aktiviert sein, d.h. auf **LOW** gesetzt sein. Das wird dadurch erreicht, dass die Steuerung des Reset im AVR-System vom In-System-Programmer vorgenommen wird.
GND	Common **G**rou**ND**	6	4, 6, 8, 10	-	Der In-System-Programmer und das zu programmierende AVR-System müssen dasselbe Masse-Potential haben.
NC	**N**o **C**onnection	-	3	-	Der Pin hat keine Verbindung.

In der von ATMEL bevorzugten ISP10PIN-Belegung auf dem Wannenstecker ist der erwähnte Pin 3 für eine zusätzliche LED nicht vorgesehen sondern unbeschaltet (NC).

2.1 ISP - Bezogen auf die verschiedenen Schnittstellen

Zur anderen Seite soll der ISP-Programmieradapter an den PC angeschlossen werden, damit die im Computer editierten und assemblierten/kompilierten Programme in den Speicher des Mikrocontrollers auf dem Experimentierboard übertragen werden können.

Dabei können die parallele, die serielle oder die USB-Schnittstelle des PC verwendet werden. Für einen neu vorzusehenden ISP-Programmieradapter sollte die USB-Schnittstelle des PC verwendet werden, da die reine serielle Schnittstelle ein Auslaufmodell ist und der Parallelport des PC - sofern noch vorhanden - meist vom Drucker verwendet wird.

2.1.1 Programmer für die serielle Schnittstelle (z.B. AVR910)

Die serielle Schnittstelle (RS-232) wird im www.mikrocontroller.net ausführlich beschrieben. Sie wird - trotz USB - ebenfalls noch bei ISP-Adaptern eingesetzt. Eine elegante Anpassung der RS-232-Schnittstelle, die mit -12 V/+12 V -Signalen arbeitet, auf die von den AVRs benötigten TTL-Pegel wird mit dem RS-232-Treiber-Chip MAX232A erreicht. Die Version MAX232A hat den Vorteil, dass für die benötigten Kapazitäten für die Pegel-Verdoppelung/Invertierung 0,1 µF ausreichen.

Die Firmware für den Programmer AVR910 basiert auf der Application Note AVR910 von ATMEL. Die Original Applikation ist allerdings schon etwas älter, und die Software des AT90S1200 wird von Atmel schon länger nicht mehr gepflegt.

Hinweis: Sollte eine serielle Schnittstelle im PC nicht mehr zur Verfügung stehen (in den meisten Notebooks sind bereits keine seriellen Schnittstellen mehr vorhanden), kann auch ein USB/RS-232-Adapter verwendet werden, um diese Programmer (weiter) zu verwenden.

2.1.2 Programmer für die parallele Schnittstelle

Ein weiteres Auslaufmodell für den ISP-Programmieradapter ist die parallele Schnittstelle. Einige Schaltungsbeispiele kann man hier finden: Programmieradapter für Atmel AVR Mikrocontroller.

2.1.3 Programmer für die USB-Schnittstelle

Der **U**niversal **S**erial **B**us als Schnittstelle des ISP-Adapters zum PC ist zu bevorzugen, da dieser Bus, der heutzutage an jedem neuen PC zu finden ist und langsam aber sicher die RS-232- und Parallelport-Anschlüsse ersetzt, schneller und universeller ist.

Übertragungsgeschwindigkeiten:

- Low Speed: 1,5 MBit/s (USB 1.1 und 2.0)
- Full Speed: 12 MBit/s (USB 1.1 und 2.0)
- High Speed: 480 MBit/s (nur USB 2.0)

Die in Datenblättern gerne verwendete Angabe "USB 2.0 Full Speed" darf man also nicht wörtlich verstehen, das sind trotz USB 2.0 nur 12 MBit/s.

Zu beachten ist, dass es beim USB (im Gegensatz etwa zu RS-232) zwei Arten von Controllern gibt: Host- und Devicecontroller. Der Hostcontroller bezeichnet dabei die steuernde Seite und ist z.B. in PCs zu finden. Devicecontroller sind in den zu steuernden USB-Geräten, z.B. USB-Druckern, zu finden. Diese Unterscheidung ist ziemlich wichtig, weil die meisten USB-Lösungen für Mikrocontroller USB-Devices darstellen und man deswegen dort weder Webcams noch USB-Speichersticks anschließen kann. Mit einer Ergänzung des Standards (USB On-The-Go) gibt es die begrenzte Möglichkeit, dass Geräte Host-Funktionalität zur Kommunikation mit ausgewählten Peripheriegeräten erhalten.

Benutzt man ein USB-Device am PC, dann braucht man auch noch passende Treiber. Die aktuellen Betriebssysteme bringen meist eine Reihe von Treibern für Standardanwendungen (z.B. USB-Drucker, USB-Scanner usw.) mit. Gibt es keinen passenden Standardtreiber, dann muss man eben einen erstellen. Dazu werden bei manchen Chips kostenlose Treiber mitgeliefert, bei anderen muss man sie kaufen oder (aufwendig) selbst erstellen.

Für einen **USB/RS-232-Adapter** wird im Gerätemanager von Windows ein weiterer Kommunikationsanschluss (**COMxx**) installiert, d.h. dieser Adapter wird mit einem besonderen Treiber zu einer virtuellen seriellen Schnittstelle. Nicht wundern: der **Assistent für das Suchen neuer Hardware** erscheint **2-mal** hintereinander!

2.2 CC2-AVR-Programmer alias USBprog

Der im ATM18-Projekt und auch hier eingesetzte **CC2-AVR-Programmer** basiert auf dem **USBprog** mit einer "echten" USB-Schnittstelle, also kein USB/RS-232-Adapter. Er wurde so umgestaltet, dass ein Benutzer ohne weitere Vorkenntnisse des USB-Programmers damit seine Mikrocontroller programmieren kann. Das USB-Datenübertragungs-Protokoll wurde dabei von **Benedikt Sauter**, dem Erzeuger des Programmers, so angepasst, dass er sowohl mit dem **AVR Studio,** mit dem **CodeVisionAVR C-Compiler** als auch mit **AVRDUDE** arbeitet.

Für weitere Informationen wird auf die Seite http://www.embedded-projects.net/ (Open-Source Projekt USBProg) von **Benedikt Sauter** verwiesen. Hier werden nur die notwendigen Informationen komprimiert zusammengestellt, um die Handhabung des Programmers zu erläutern.

2.2.1 Aufbau

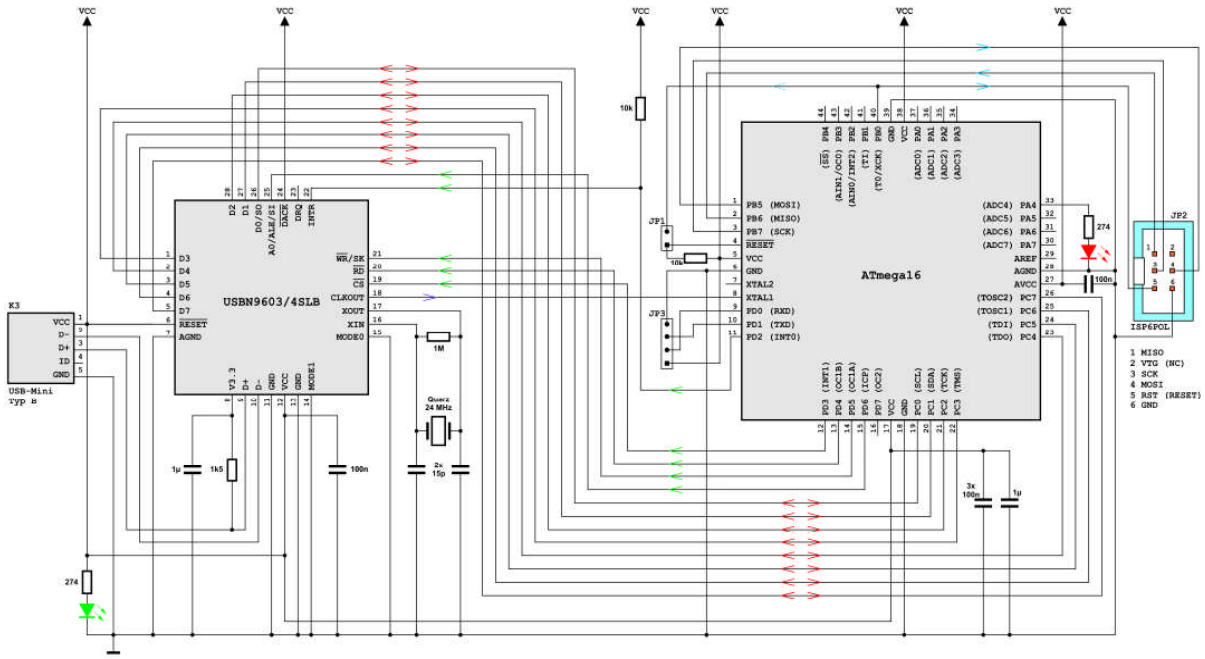


Bild 2.2.1-01: Schalplan des CC2-AVR-Programmers nach Benedikt Sauter (Bildvergrößerung)

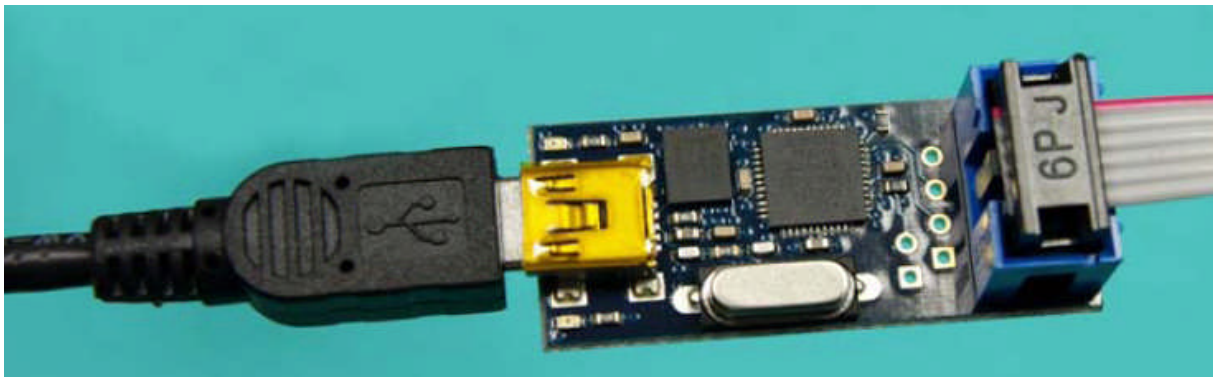


Bild 2.2.1-02: Der CC2-AVR-Programmer einschl. Verbindungskabel (Bildvergrößerung)

Der CC2-AVR-Programmer wurde speziell für das Testboard des ATM18-Projekts entwickelt, so dass er nur im Shop von **Elektor** als Artikel mit der Nummer **080083-71** geführt wird. Er kostet dort einschließlich USB-Verbindungskabel und 6-poligem Flachbandkabel **32,00 €**.

Im Grunde kann man viele AVR-Mikrocontroller mit diesem Adapter programmieren. Offiziell ist er aber nur für den ATMega88 und für einige andere Typen positiv getestet worden. Da der CC2-AVR-

Programmer mit einem **ATMega16** statt eines **ATMega32** bestückt ist, musste auf die Bootloader-Funktion, wie sie im **USBprog** vorhanden ist, verzichtet werden. Sollte das Testboard auch für den Einsatz anderer Mikrocontroller erweitert werden, so ist die Verwendung des USBprog zu empfehlen, der nur 2 € mehr kostet. Er ist flexibler und bietet noch zusätzliche Funktionen.

Der Programmer ist eingangsseitig mit einer Buchse **USB-Mini Typ B** ausgestattet, die vom PC aus den Programmer mit +5 Volt (VCC) versorgt. Ausgangsseitig stellt ein 6-pol-Wannenstecker **ISP6POL** die Verbindung zum Testboard her. Da der Programmer kompatibel zu AVRISP-mkII ist, kann man bei der Installation von AVR Studio auch gleich die entsprechenden Treiber mitladen (siehe **Teil 04**, Abschnitt **4.1 AVR Studio installieren**). Eine Stromversorgung des Testboards aus der USB-Schnittstelle heraus über den Programmer wurde allerdings unterbunden - Pin 2 des ISP ist nicht beschaltet.

2.2.2 Arbeitsweise

Der Programmer wird - wie im **Bild 1.1-02: Zusammenschaltung von PC, ISP und Experimentierboard** dargestellt - zwischen PC und Testbord geschaltet und über die Treiber des AVR-Chip-Programmer-Typs **AVRISP-mkII** gesteuert. Er unterstützt Programmier-Taktraten im Bereich von 249 Hz bis 2 MHz und kann sowohl vom **AVR Studio** als auch vom **CodeVisionAVR C-Compiler** (CVAVR) aus zum Flashen aufgerufen werden. Wie man die Taktraten im AVR-Studio bzw. im CodeVisionAVR C-Compiler einstellt, wird in den entsprechenden Abschnitten des ASM-Assemblers bzw. des CVAVR-Compilers beschrieben.

Die speziell für das CC2-ATM18-Projekt angepasste Firmware ist in einem ATmega16-Controller gespeichert und steuert den Chip USBN9604SLB für die schnelle Übergabe der Daten vom USB-Port und das Einspeichern des Objektprogramms in den Flash-Speicher des Ziel-Mikrocontrollers über die 6-polige ISP-Schnittstelle. Die grüne LED signalisiert die vom USB kommenden +5 V und die rote LED signalisiert die beim Flashen übertragenen Bits durch ein Flackern während der Übertragung.

2.2.3 Firmware-Änderung

Wie oben schon angedeutet, lässt sich ein AVR-Mikrocontroller nur dann über die ISP-Schnittstelle flashen, wenn die drei Pins **MOSI**, **MISO** und **SLK** in den Programmiermodus geschaltet sind, d.h. wenn am Reset-Pin des AVR logisch **0** also **GND**-Potential anliegt. Im Programmierbetrieb legt der Programmer als aktiver Teil die logische **0** mit seinem Pin **PB0** an den Reset des Ziel-AVR.

Wie wird ein Programmer "programmiert"? Wenn nun der ATmega16 vom Programmer ① selbst zum Ziel einer Programmierung werden soll, damit in ihm eine neue Firmware geflasht werden kann, so ist das auf Anhieb nicht möglich! Für das Flashen ist ein weiterer Programmer ② für die Programmierung und ein weiterer PC für die Versorgung des Ziel-Programmers ① notwendig, da sein Reset-Eingang (Pin 4) mit einem Pull-Up-Widerstand auf logisch **1** gesetzt werden muss. Erst wenn der Jumper **JP1** überbrückt wird, so dass Pin 4 über die ISP-Schnittstelle in den Programmmodus geschaltet werden kann, ist auch über die ISP-Schnittstelle ein Flashen des Programmers ① möglich.

Der Programmer ② wird mit dem USB-Port des PC's für die Programmierung verbunden und der Programmer ① wird mit seinem USB-Kabel an einen beliebigen anderen PC zur Stromversorgung angeschaltet. Die beiden Programmer werden untereinander über ihre ISP-Schnittstellen verbunden. Beim Flashen ist darauf zu achten, dass im AVR Studio der **ATmega16** und ein Programmier-Takt nicht über **250 kHz** eingestellt wird. Die Einstellungen der Fuses sollten nicht verändert werden; ihre Einstellungen sind:

LOW Fuse: `0xE0`
und
HIGH-Fuse: `0xD9`

Welche Firmware soll in den Programmer ① geflasht werden? Die Seite http://www.embedded-projects.net/cc2avrprog ist leider nicht mehr vorhanden. Die Firmware (Text) des CC2-AVR-Programmers (**`cc2programmer02.hex`**) ist aber noch auf der folgenden Seite zu finden:

http://code.google.com/p/usbprog/source/browse/trunk/cc2programmer