Praktikum Mikrocontrollertechnik SS 2015
Versuch 1

Stand: 13. 4. 2015

Einführung in die C-Programmierung Atmel AVR. Die Programmiersprache C ist alt und von Grund auf häßlich. Es ist aber die am meisten verwendete Programmiersprache.

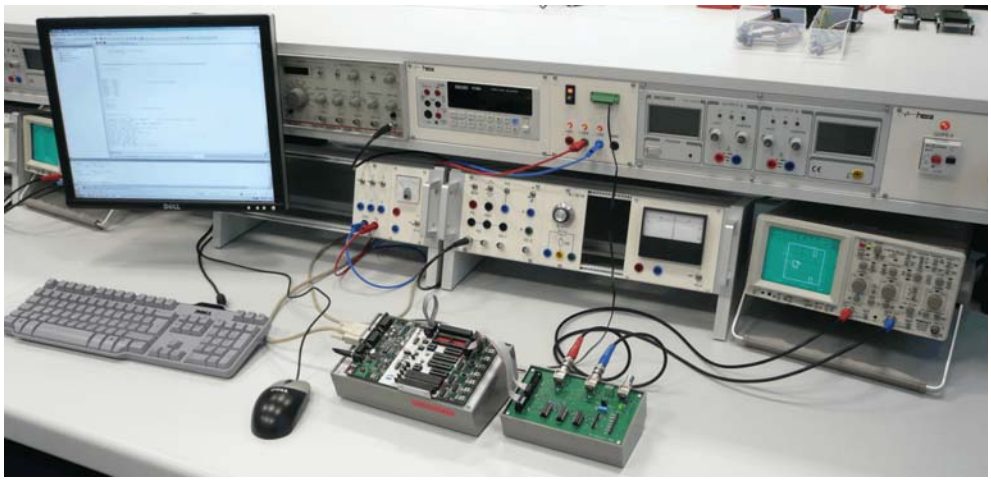
Aufgaben:

1. Kennenlernen des grundsätzlichen Entwicklungsgangs.
2. Darstellen von Zeit mittels Software.
3. Elementare Anwendungsprogrammierung.

Ausrüstung:

- AVR Studio 4 (4.19),
- WinAVR C-Compiler (AVR GCC),
- Starterkit AVR STK 500,
- Siebensegmentanzeige 09/13.

Zu den Geräten siehe die jeweilige Kurzbeschreibung, zum Starterkit auch die Originaldokumentation der Fa. Atmel.



Aufgabe 1: Den Entwicklungsgang kennenlernen

Jedes Vorhaben ist ein Projekt. Wenn Sie etwas Neues beginnen wollen, müssen Sie im AVR Studio ein neues Projekt einrichten.

1. Software starten.
Die Anwendung heißt AVR Studio 4.

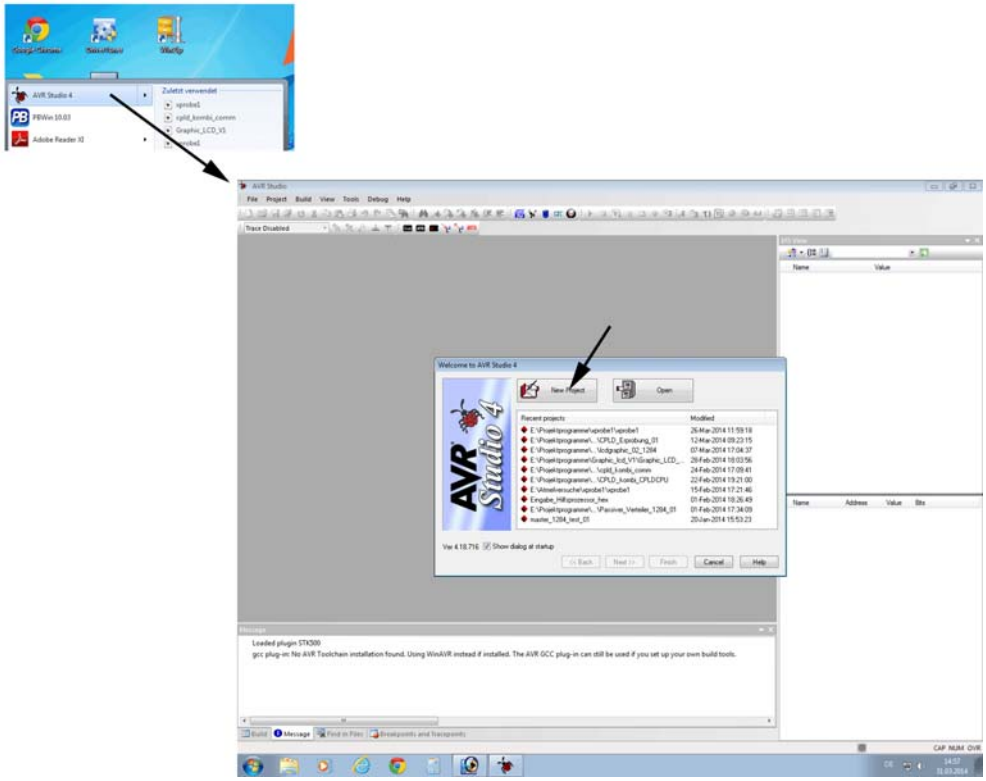


Abb. 1 Das AVR Studio wurde gestartet. Wir wollen eine neues Projekt einrichten.

2. Ein neues Projekt anlegen.

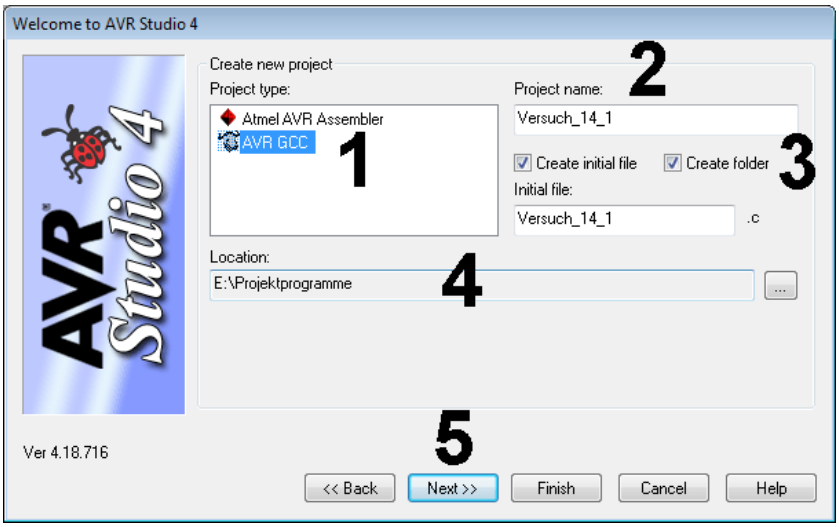


Abb. 2 Das Projekt benennen und anlegen. 1 - es soll in C geschrieben werden. 2 - Namen eintragen. 3 - diese Kontrollkästchen müssen beide aktiv sein. 4 - bitte ein vernünftiges Verzeichnis auswählen. Das Projekt nicht auf dem Desktop oder auf der Netzfestplatte ablegen... 5 - weiter.

3. Den Mikrocontroller auswählen.

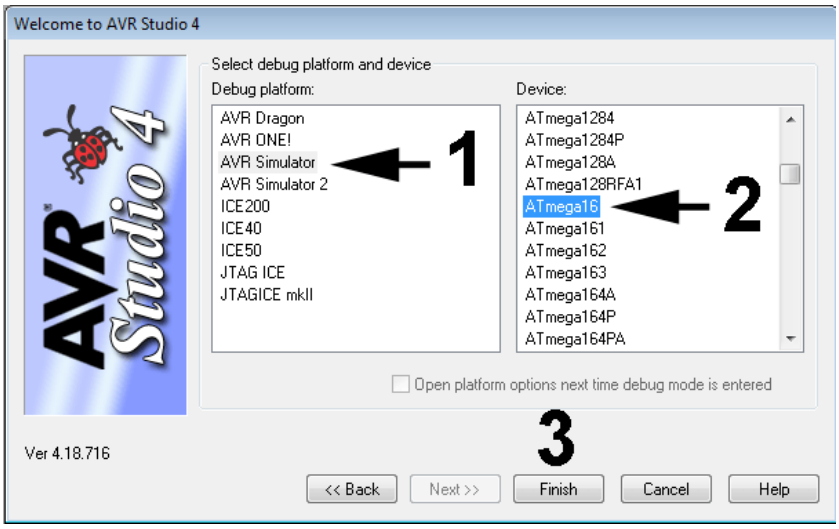


Abb. 3 Fehlersuchplattform und Mikrocontroller auswählen. 1 - wir wählen den AVR Simulator zum Verfolgen der Programmabläufe (obwohl wir ihn nicht brauchen und unsere Fehler zu Fuß suchen wollen). 2 - unser Mikrocontroller ist der Atmega 16. Er ist leistungsfähig genug und ausreichend ausgestattet, aber nicht zu kompliziert. 3 - Projekterstellung beenden.

4. Ein Musterprogramm übernehmen und übersetzen.

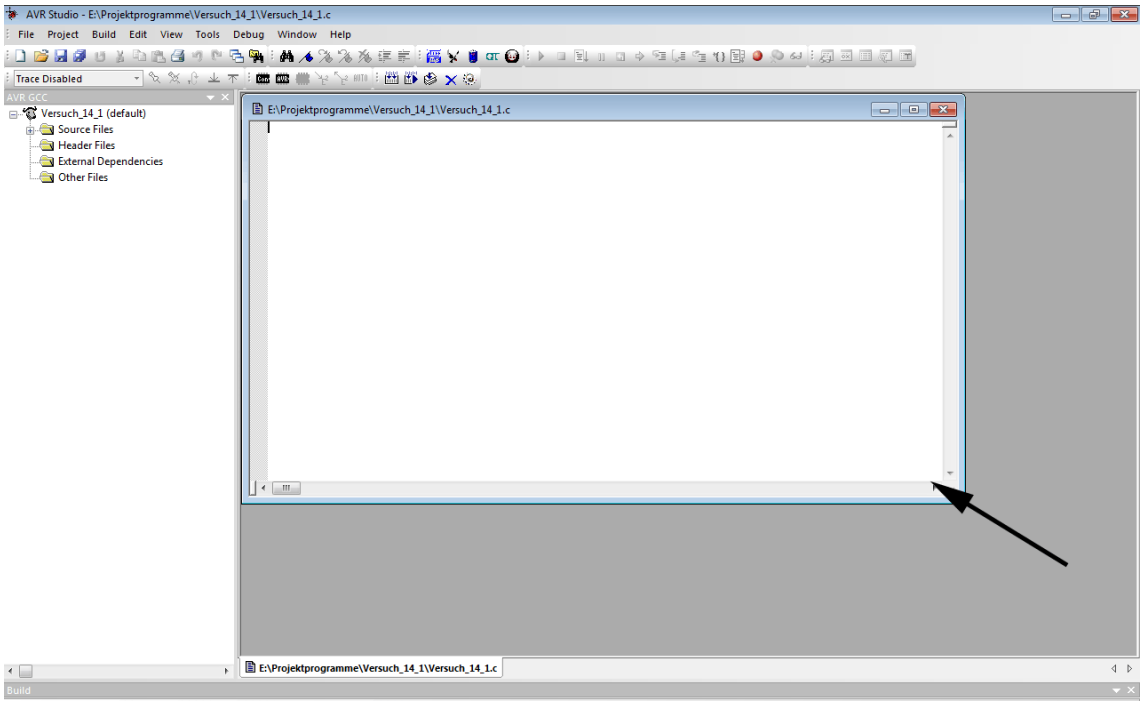


Abb. 4 Das Projekt ist geöffnet. Der Pfeil zeigt auf das (leere) Eingabefenster des Programmeditors.

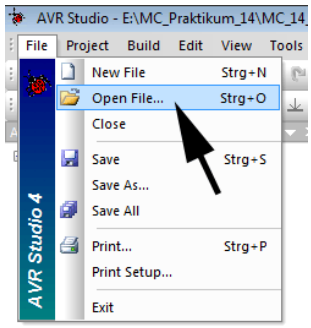


Abb. 5 Wir wollen ein Musterprogramm übernehmen.

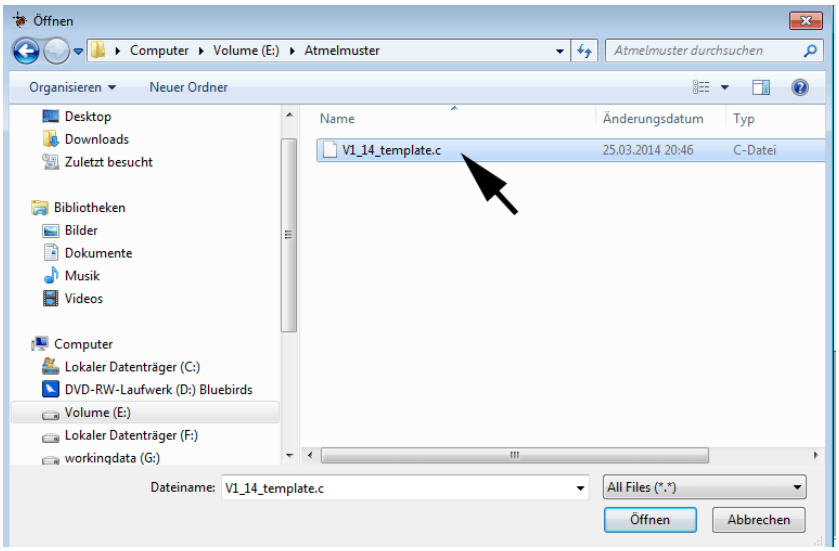


Abb. 6 Das Musterprogramm auswählen und öffnen. Es heißt "V1_15_template.c".

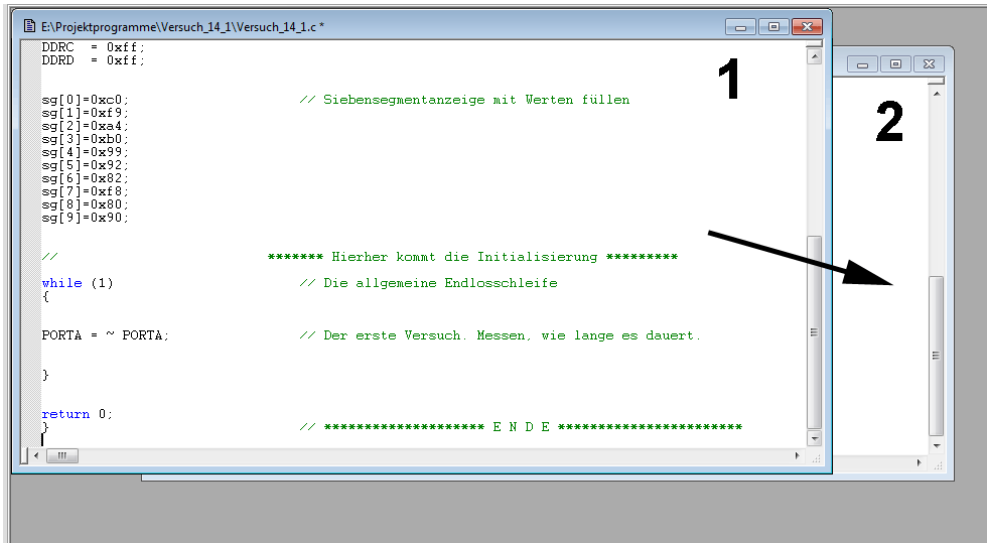


Abb. 7 Das Musterprogramm wird ins Projekt übernommen. 1 - Musterprogramm; 2 - Projektprogramm. In 1 alles markieren. Kopieren. in 2 einfügen. 1 schließen.

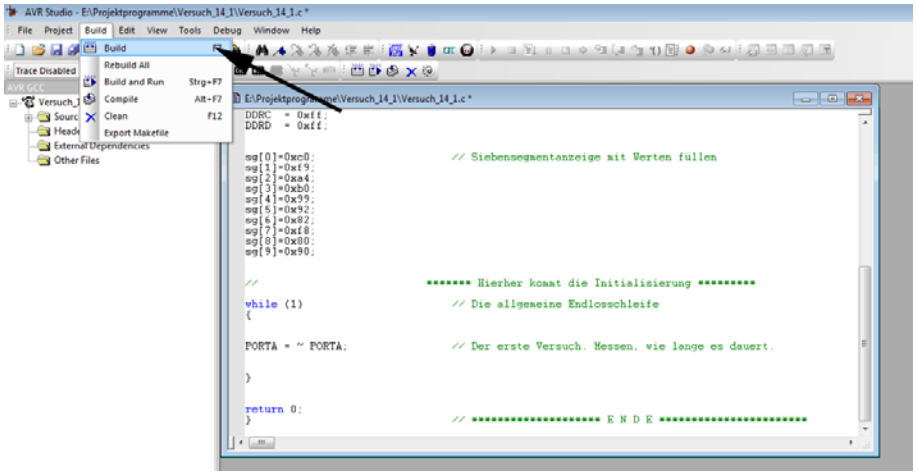


Abb. 8 Jetzt wird das Programm übersetzt. Auswahl “Build” – “Build”.

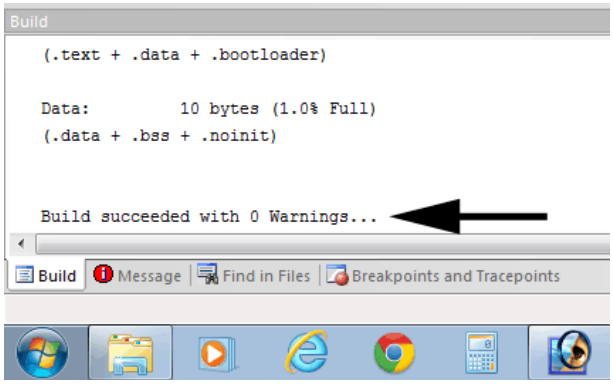


Abb. 9 Diese Nachricht (Pfeil) ist entscheidend. Jetzt kann der Mikrocontroller programmiert werden.

5. Den Mikrocontroller programmieren.

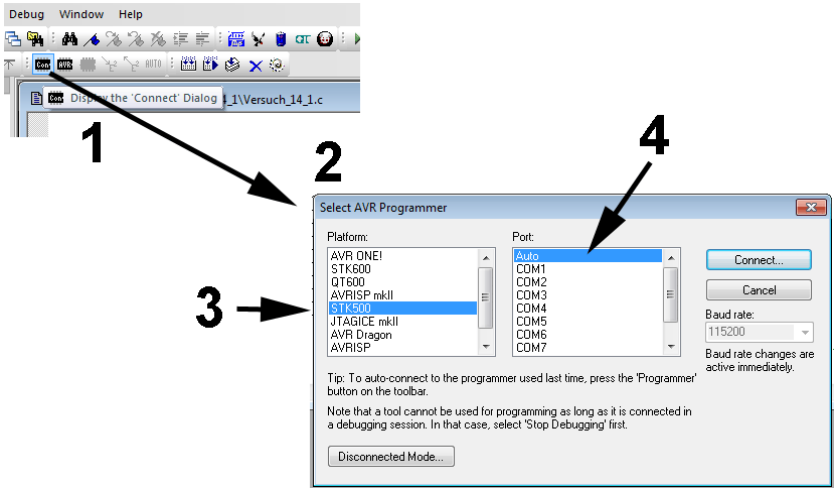


Abb. 10 Der Programmer wird ausgewählt und aktiviert. 1 - mit dem Programmer verbinden. 2 - der Auswahldialog. 3 - es ist das Starterkit STK 500. Es ist an eine serielle Schnittstelle angeschlossen. 4 - wir lassen das Programm die Schnittstelle selbst finden.

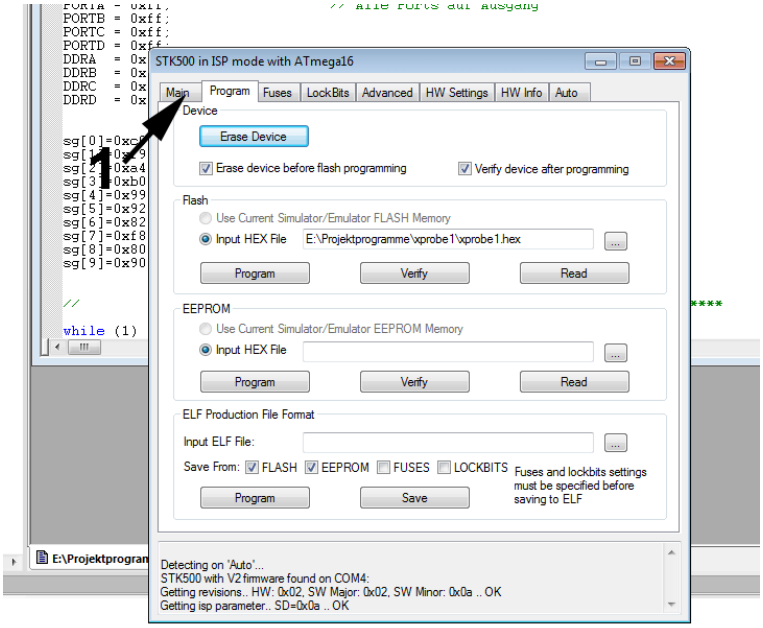


Abb. 11 So meldet sich der Programmer. 1 - wir sollten aber nicht gleich programmieren, sondern erst einmal hier nachschauen, ob auch alles stimmt.

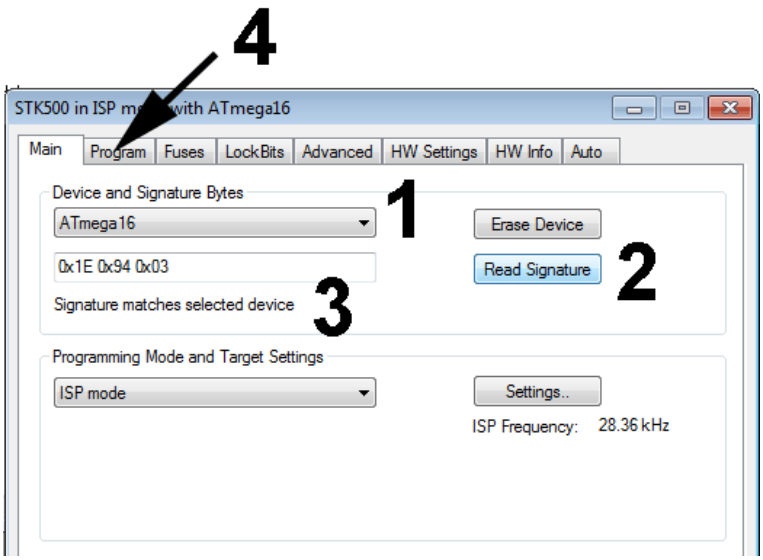


Abb. 12 Vertrauen ist gut, Kontrolle ist besser... 1 - ist der richtige Controllertyp ausgewählt? 2 - zur Kontrolle die Signatur auslesen. 3 - diese Aussage ist entscheidend. Der Schaltkreis auf dem Starterkit ist tatsächlich vom ausgewählten Typ. 4 - zurück zum Programmieren.

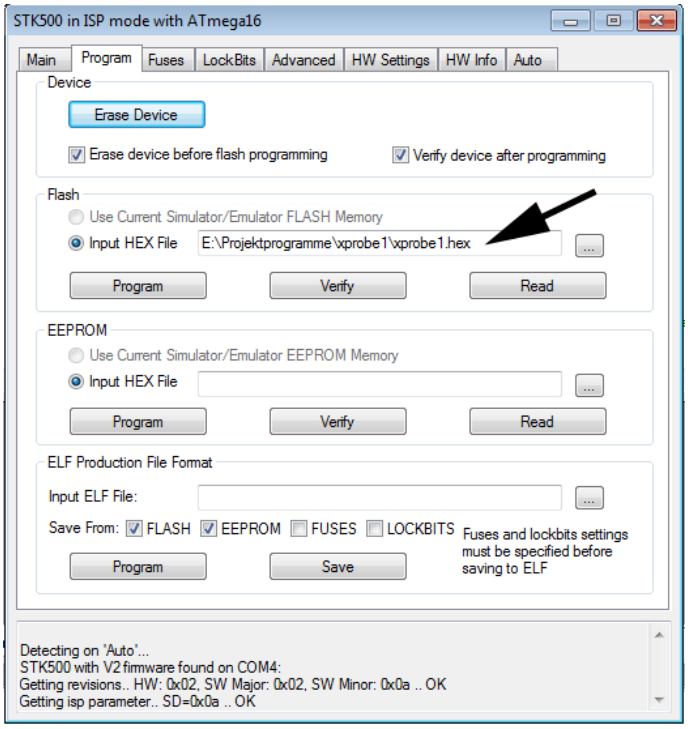


Abb. 13 Das Programm muß in den Flash. Hierzu müssen wir zunächst die Programmdatei auswählen. Der Programmer hat keinen automatischen Datenverbund mit der Programmentwicklung.

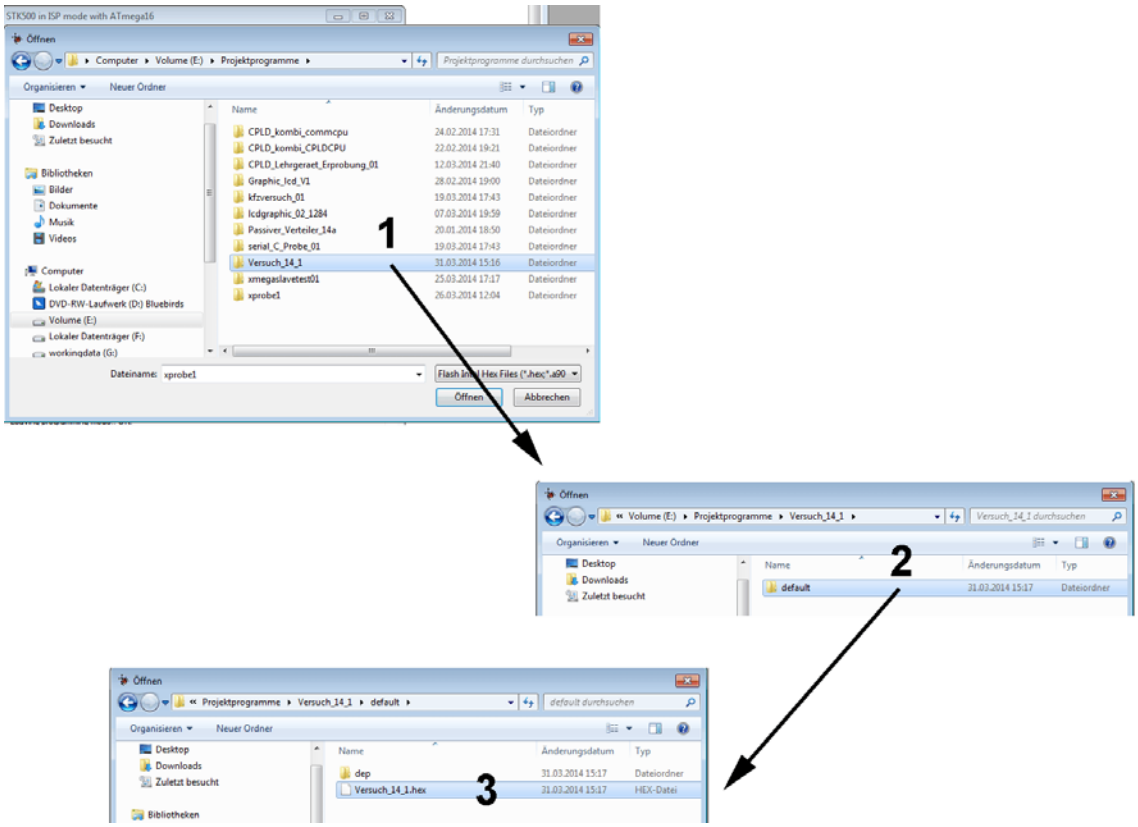


Abb. 14 Programmierdateiauswahl. 1 - das Verzeichnis des Projekts öffnen. 2 - das Unterverzeichns “default” öffnen. 3 - wir brauchen die Datei mit der Endung .hex. Auswählen.

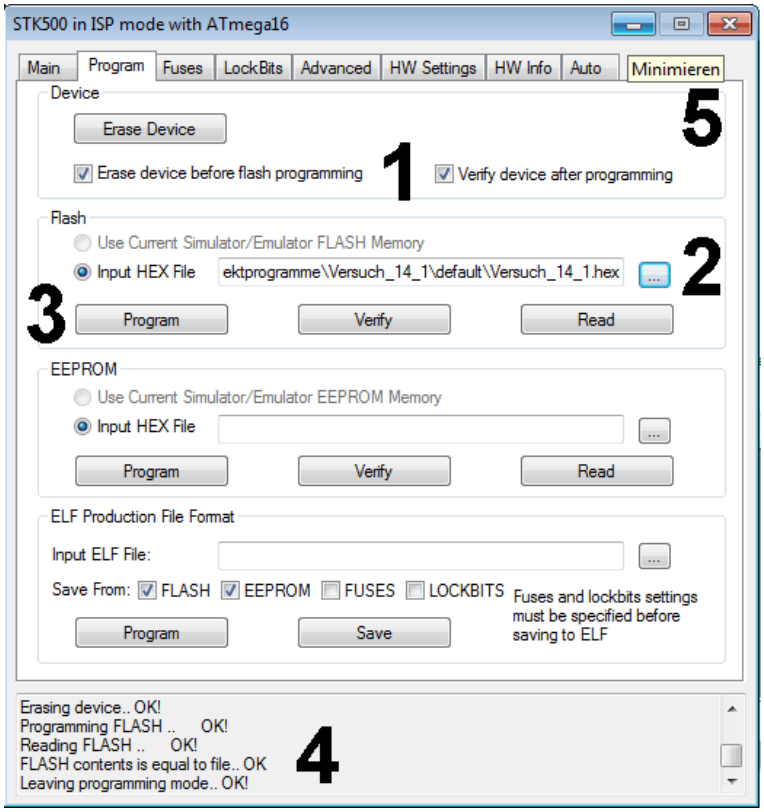


Abb. 15 Programmieren. 1 - diese beiden Kontrollkästchen müssen aktiv sein. 2 - die Programmierdatei ist ausgewählt. 3 -Programmieren auslösen. 4 - die Erfolgsmeldungen. Fertig. 5 - Programmierdialog nicht schließen, sondern minimieren (um ihn später wieder aufrufen zu können).

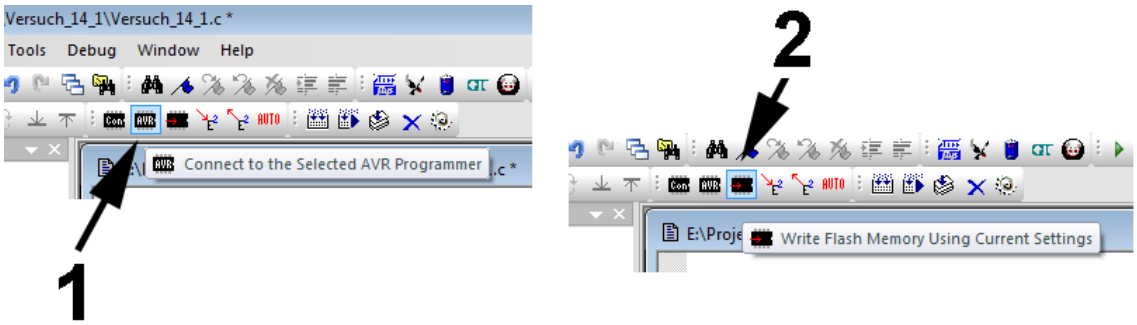


Abb. 16 Programmerauswahl. 1 - mit dem ausgewählten Programmer verbinden. 2 - den Flash gemäß den aktuellen Einstellungen neu programmieren.

Aufgabe 2: Den Aufbau eines C-Programms verstehen.

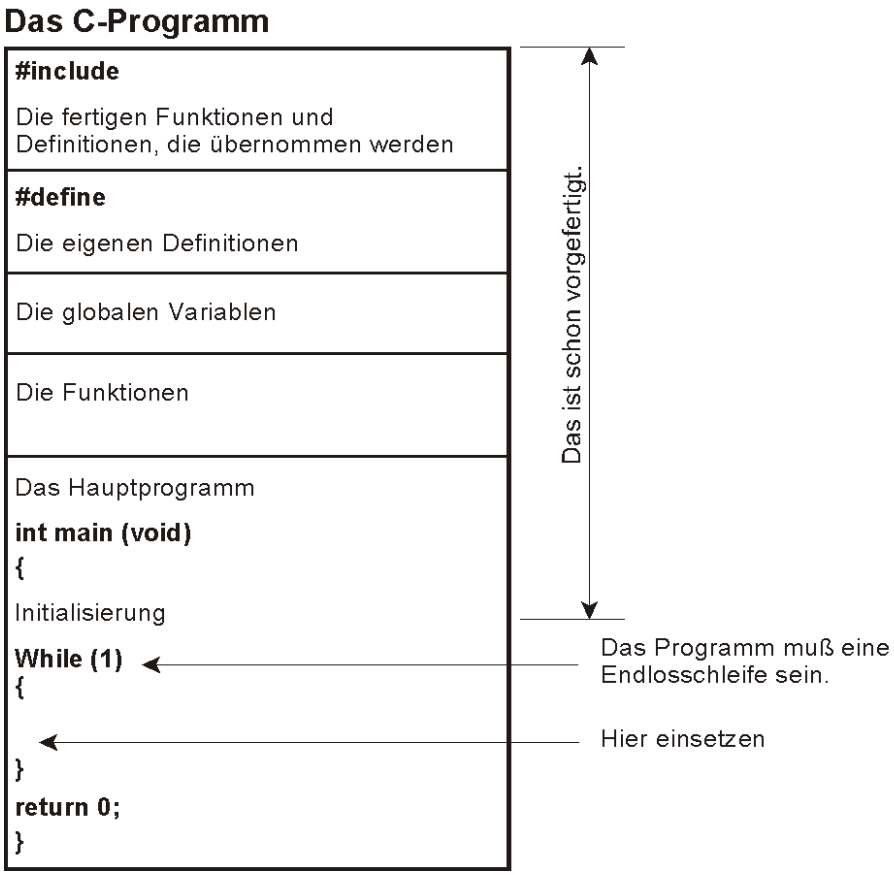


Abb. 17 Die grundsätzliche Struktur eines C-Programms.

Aufgabe 3: Programmlaufzeiten messen.

Eine beliebige Bitposition von Port A mit dem Oszilloskop verbinden (Tastkopf). Wir lassen das Musterprogramm laufen. In der Endlosschleife wird der Port A zyklisch umgeschaltet. Es müssen symmetrische Rechteckimpulse angezeigt werden. Damit kann die Durchlaufzeit der Schleife gemessen werden. Eine Halbperiode = ein Schleifendurchlauf. Zeit messen.

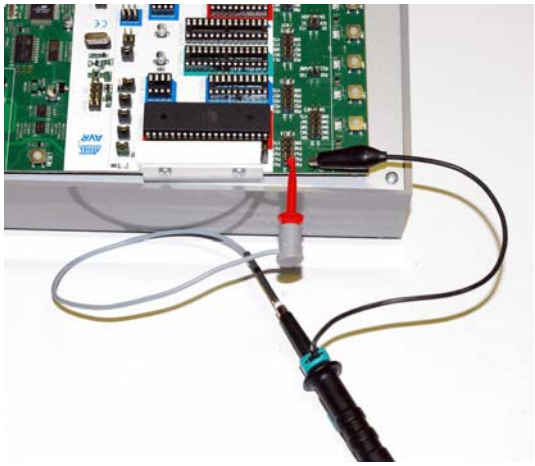


Abb. 18 Zeitmessung mit Oszilloskop.

Die Durchlaufzeit hängt vom Compiler ab. Die Konfigurationseinstellungen des Projekts aufrufen.

Welchen Maschinencode hat der Compiler erzeugt? Das ist aus dem sog. List File ersichtlich (Dateiendung .lss).

Der Maschinencode hängt von der gewählten Optimierungsstufe ab. O0 = keine Optimierung. Os = höchste Optimierung. Mit beiden Einstellungen probieren. List Files ansehen.

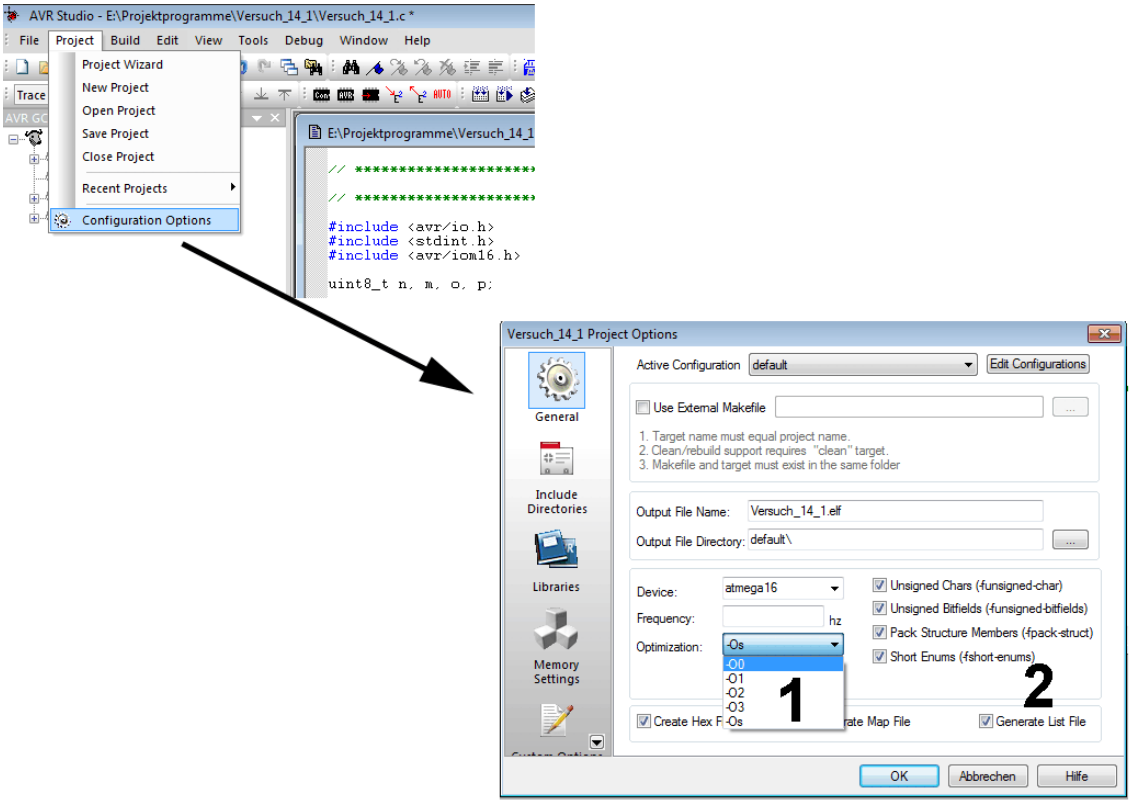


Abb. 19 Die Konfigurationseinstellungen des Projekts. 1 - Optimierung; 2 - List File erzeugen / nicht erzeugen.

Selbst programmieren

Um die folgenden Aufgaben zu lösen, müssen Sie selbst programmieren. Die Programmabläufe fügen Sie in mit dem Editor des AVR Studio in die geöffnete C-Quelldatei ein.

Achtung:
Der Compiler achtet auf die Groß- und Kleinschreibung. Die Atmel-AVR-Definitionen (Ports, Steuerbits usw.) werden stets groß geschrieben, reservierte Bezeichner der Programmiersprache hingegen klein. Korrekt geschriebene reservierte Bezeichner werden blau wiedergegeben, Kommentare grün. Kommentarzeichen: // (zeilenweise) oder /* */ (ganze Blöcke). Die Semikolons nach den Anweisungen nicht vergessen!

Richtig	Falsch
while (1)	WHILE (1)
PORTA	porta
RXEN	Rxen

Programmbeispiele

Programmbeispiele finden Sie auf der C-Quelldatei “V1_15.examples.c”. Sie steht im selben Verzeichnis wie das Musterprogramm. Sie können diese Datei im Editor öffnen, und Programmstücke in die C-Quelldatei Ihres Projekts übernehmen. Manchmal sind zudem noch Änderungen erforderlich, beispielsweise in der Initialisierung der E-A-Ports. Es versteht sich von selbst, daß Sie auch alternative eigene Lösungen ausprobieren dürfen.

Aufgabe 4: Zeit darstellen.

Wir stellen die Projektkonfiguration auf O0 = keine Optimierung. Das Ziel ist eine Funktion millisecs(), der ein 16-Bit-Wert als Parameter übergeben wird. Die Funktion wartet so viele Millisekunden ab, wie dieser Wert angibt (Warteschleife).

Hierzu schreiben wir zunächst eine Funktion milli(), die genau eine Millisekunde wartet. Erprobung: durch Einfügen in die Endlosschleife. Die Durchlaufanzahl (eine Konstante) in der Funktion so lange ändern, bis der Schleifendurchlauf näherungsweise eine Millisekunde dauert.

Mit dieser Funktion die Funktion millisecs() schreiben und erproben. millisecs (n) ist einfach ein n-maliges Aufrufen von milli().

Wenn alles klappt, ausprobieren, was sich ergibt, wenn die Optimierung auf Volldampf (Os) geschaltet wird. Dann aber wieder zurück zu O0 (jetzt wissen wir nämlich, warum...).

Programmbeispiel 01.

Aufgabe 5: Lauflicht in eine Richtung.

Die LEDs des Starterkits mit Port C verbinden. Zyklische aufeinanderfolgende Erregung in eine Richtung. Die LEDs sind aktiv Low.

Abb. 20 Die LEDs an Port C.

Programmbeispiel: 02.

Aufgabe 6: Lauflicht in beide Richtungen (hin und her).

Zyklische aufeinanderfolgende Erregung der LEDs, zunächst in die eine Richtung, dann zurück.

Programmbeispiel: 03.

Aufgabe 7: Gebäudesystemtechnik (1).

Das elektrische Licht ein- und ausschalten. Tasten an Port D anschließen. Port D auf Eingang. Mit einer der Tasten (z. B. Taste 7) soll die gesamte LED-Reihe ein- und ausgeschaltet werden (erste Betätigung: ein, zweite Betätigung: aus usw.). Die Tasten sind aktiv Low.

Abb. 21 LEDs an Port C, Tasten an Port D.

Programmbeispiel: 04.

Aufgabe 8: Gebäudesystemtechnik (2).

Das Ein- und Ausschalten funktioniert nicht vollkommen sicher. Mal klappt's, mal nicht. Das liegt daran, daß die Taste prellt. Jetzt Entprellung (durch Zeitverzögerung) einbauen.

Programmbeispiel: 05.

Aufgabe 9: Gebäudesystemtechnik (3).

Das elektrische Licht nacheinander ein- und ausschalten (Power Sequencer). Am Anfang sind alle LEDs dunkel. Mit einer der Tasten eine Einschaltsequenz auslösen. Zunächst schaltet nur die erste LED, nach etwa 300... 500 ms kommt die zweite hinzu usw., bis schließlich alle acht eingeschaltet haben. Wird dieselbe Taste erneut betätigt, sollen die LEDs in umgekehrter Reihenfolge nacheinander ausschalten.

Programmbeispiel: 06.

Aufgabe 10: Zweistellige Siebensegmentanzeige.

Die Siebensegmentanzeige 09/13 anschließen. Wir verwenden zwei Anzeigemodule, die wir mit den Ports B und C verbinden.

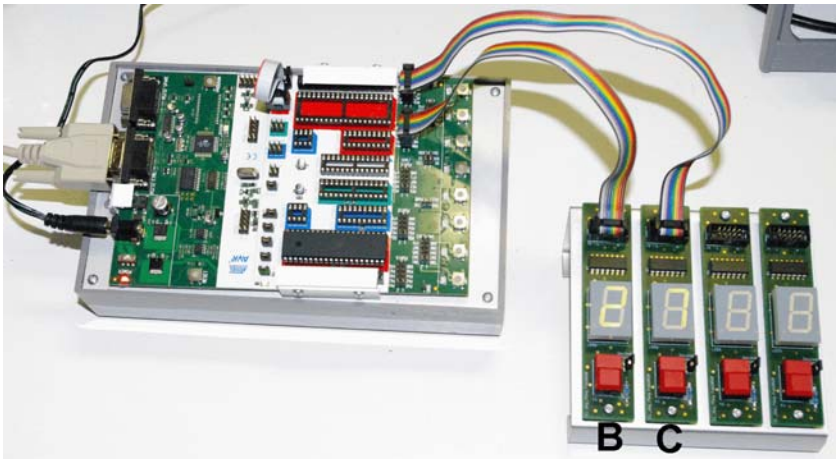


Abb. 22 Starterkit mit Siebensegmentanzeige. LEDs und Tasten sind aktiv Low.

Jeder der Ports:

7	6	5	4	3	2	1	0
KEY#	G#	F#	E#	D#	C#	B#	A#

Es sind vier Teilaufgaben zu bearbeiten:

10.1. Zyklisch von 0 bis 99 zählen.
Genügend Zeitverzögerung zwischen den Zählschritten vorsehen (z. B. 200 ms), so daß der Zählvorgang visuell verfolgt werden kann. Worauf es besonders ankommt, ist die Wandlung vom Binären (der Zählvariablen in der Schleife) ins Dezimale und von da aus in die Segmentbitmuster. Wir können hierzu die Grundrechenarten ausnutzen; Tricksen müssen wir nicht. Ein deutliches Beispiel für die Vorteile einer "höheren" Programmiersprache...

Programmbeispiel: 07.

10.2. Handzähler (zum Zählen von Autos, Schafen, BVB-Fans usw.).
Rechte Taste zum Zählen (+ 1), linke Taste zum Löschen. Bei 99 mit Zählen aufhören. Probeweise die Entprellung weglassen (auskommentieren). Die Tasten auf den LED-Anzeigen prellen typischerweise gut und lassen so den Prelleffekt deutlich erkennen.

Programmbeispiel: 08.

10.3. Zyklisch von 1 bis 49 zählen (Lottozahlen).
Genügend Zeitverzögerung zwischen den Zählschritten vorsehen (z. B. 200 ms), so daß der Zählvorgang visuell verfolgt werden kann.

Programmbeispiel: 09.

10.4. Lottozahlen ziehen.
Schnell zählen, solange die rechte Taste gedrückt ist. Bei losgelassener Taste hält der Zähler an.

Programmbeispiel: 10.

Aufgabe 11: Tageszeituhr mit vierstelliger Siebensegmentanzeige.

Alle vier Siebensegment-Anzeigemodule an das Starterkit anschließen.

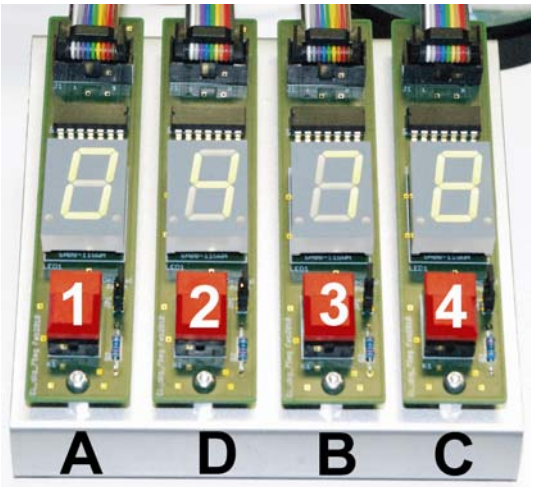


Abb. 23 So werden die Anzeigemodule angeschlossen.

Auch hier sind vier Teilaufgaben zu bearbeiten:

11.1. Tageszeituhr (1). Elementare Zeitzählung von 00:00 bis 23:59.
Erst einmal das Zählen an sich ausprobieren. Die Uhrzeit hat es in sich... Wir wollen hier binär zählen und zwecks Anzeige ins Dezimale wandeln. Eine solche Uhr schaltet alle Minute weiter. Der Zähltakt ist also die Minute. Der Tag hat 1440 Minuten. Also modulo 1440 zählen. 16 Bits genügen.

Wenn wir einen Minutenwert durch 60 teilen, erhalten wir als Quotienten die Stunde und als Rest die verbleibenden Minuten. Beide Werte sind in jeweils zweistellige Dezimalzahlen zu wandeln. Hierzu werden sie durch 10 geteilt. Der Quotient ergibt die höherwertige Dezimalstelle, der Rest die niederwertige.

Nun können wir nicht wirklich mit einem Minutentakt arbeiten, denn dann würde es viel zu lange dauern, den Zählablauf zu beobachten. Deshalb verwenden wir zunächst einen Takt von 100 ms.

Programmbeispiel: 11.

11.2. Tageszeituhr (2). Wählbarer Zähltakt.
Jetzt soll der Zähltakt über die Tasten unter den Siebensegmentanzeigen ausgewählt werden. Um jeweilige Taste nicht dauernd niederhalten zu müssen, übernehmen wir die Zähltaktauswahl in eine Merkvariable **status**, die wir neu definieren. Zudem müssen die Bits 7 der E-A-Ports Eingänge werden.

- Taste 1: Zählen mit 1 ms.
- Taste 2: Zählen mit 10 ms.
- Taste 3: Zählen mit 100 ms.
- Taste 4: Zählen mit 1 s.

Die Statusvariable wird auf einfache und schmucklose Weise gesetzt und dann mit einer **switch**-Anweisung ausgewertet.

Sehen Sie nach, wie lang das Maschinenprogramm ist.

Programmbeispiel: 12.

11.3. Stoppuhr (1). Überprüfen der Zählfunktion.
Die Stoppuhr soll von 00.00 bis 99.99 Sekunden zählen, und zwar mit einer Auflösung von 1/100 s. Um die Zählabläufe überprüfen zu können, behalten wir die Zähltaktauswahl der vorigen Aufgabe bei, stellen aber die Anzeige auf reine Dezimalzählung um (alle Stellen von 0...9).

Programmbeispiel: 13.

11.4. Stoppuhr (2).
Taste 1 dient als Löschtaste, Taste 4 als Start- und Stoptaste. Der Zähltakt dauert jetzt 1/100 s = 10 ms. Das Programm muß alles allein leisten: den Zähltakt herstellen, zählen, wandeln, die Tasten abfragen und die Grundfunktionen einer Stoppuhr erledigen (laufen, anhalten, löschen). Der Zustandsautomat ist ein bewährtes Prinzip, eine solche Aufgabenvielfalt zu beherrschen. Wir verwenden folgende Zustandscodierung:

- 0 = Grundzustand. Ruhe
- 1 = Taste 4 wurde betätigt. Uhr zählt.
- 2 = Uhr zählt. Taste 4 wurde losgelassen.
- 3 = Uhr steht. Taste 4 wurde betätigt. Übergang nach 0, wenn Taste wieder losgelassen.

- 9 = Zählerstand 99.99 = Zählende. Wenn Taste 1 betätigt (Löschen), dann Übergang nach 10.
- 10 = Löschtaste betätigt. Löschen. Wenn beide Tasten losgelassen, Übergang nach 0.

Programmbeispiel: 14.

Das Programmbeispiel veranschaulicht, daß ein solcher Zustandsautomat auf überschaubare, reguläre Weise mit einer **switch**-Anweisung ausprogrammiert werden kann.