



EMTI (Embedded TINY program language)

Dieses Dokument beschreibt die ersten Schritte zur Einrichtung und erfolgreichen Anwendung des EMTI- Compilers.

- Installation und Konfiguration von AVR-Studio
- Installation und Konfiguration des EMTI-Compilers
- Eingabe, Compilation und Simulation eines Testprogramms.

Hinweis: Die Benutzung des EMTI-Compilers erfolgt auf eigenes Risiko.

Inhaltsverzeichnis

Schritt 1: Installation von AVR-Studio.....	3
Schritt 2: Der erste Start von AVR-Studio.....	5
Schritt 3: Installation und Konfiguration des EMTI-Compilers.....	8
Schritt 4: Erstes Testprogramm erstellen.....	9
Schritt 5: Testprogramm compilieren.....	10
Schritt 6: Testprogramm assemblieren.....	11
Schritt 7: Testprogramm simulieren.....	13

Hinweis:

- Die im Dokument genannten Marken gehören ihren Eigentümern.
- Die abgebildeten Screenshots wurden teilweise beschnitten, um die Lesbarkeit in diesem Dokument zu verbessern.

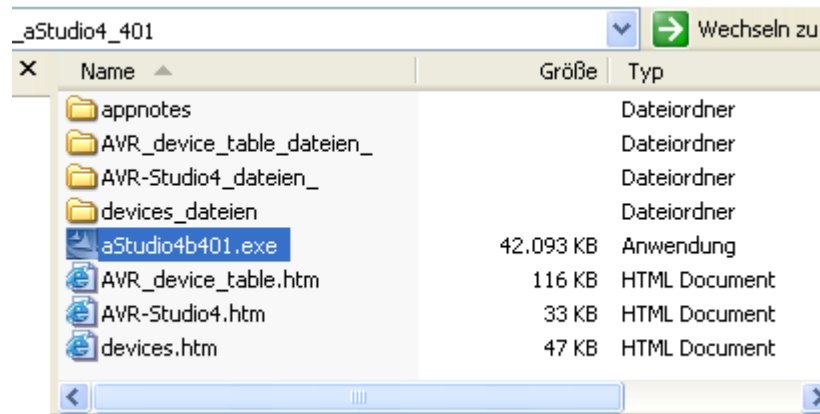
Das Programm AVR-Studio ist Eigentum der Firma ATMEL. Es kann über folgende Webseite als Download kostenlos bezogen werden:

www.atmel.com

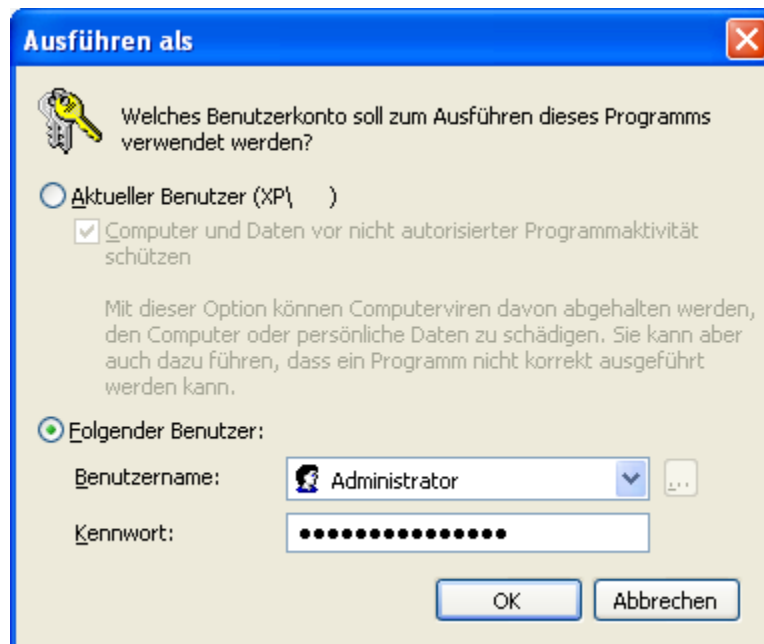


Schritt 1: Installation von AVR-Studio

Installationsprogramm starten



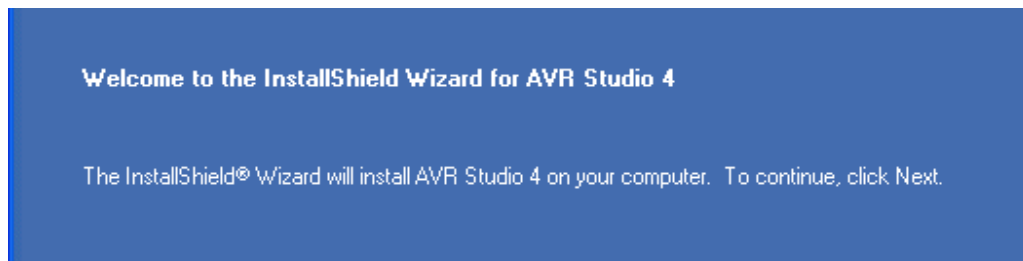
Die Installation von Software erfordert Admin-Rechte. Daher startet man das Installationsprogramm als Administrator (Rechtsklick auf den Programmnamen, im Kontextmenü "Ausführen als ..." anklicken).



Dann den Namen des Administrators auswählen und das zugehörige Kennwort eingeben.

Sicherheitshinweis: Arbeiten Sie grundsätzlich immer mit eingeschränkten Benutzerrechten.

Anschließend startet das Installationsprogramm.



Nacheinander erscheinen weitere Bildschirmmasken, die den Installationsvorgang steuern und Informationen abfragen.

Bildmaske	Beschreibung	Aktion
Welcome to the InstallShield Wizard	Startbild	Next
License Agreement	Bestätigung der Lizenzvereinbarungen Muss mit "I accept" bestätigt werden.	Next
Choose Destination Location	Auswahl des Installationspfads. Keine Änderung erforderlich	Next
Select Features	Für diverse Debugger kann ein USB-Treiber installiert werden. Man kann den Treiber später noch nachinstallieren, falls man ihn benötigt. Bei Verwendung des AVRISP MKII wird kein Treiber benötigt.	Next
Ready to Install	Installationsvorgang starten	Install
Setup Status	Information über den Fortschritt	
InstallShield Wizard Complete	Die Installation ist beendet.	Finish

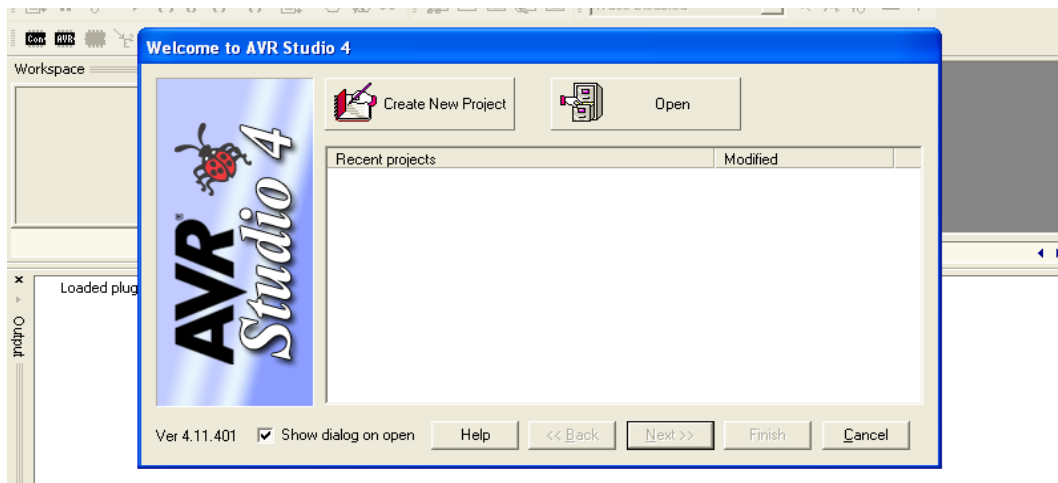
Nach der Installation gibt es im Startmenü einen neuen Eintrag:



AVR-Studio startet nach einem Klick auf "AVR Studio 4".

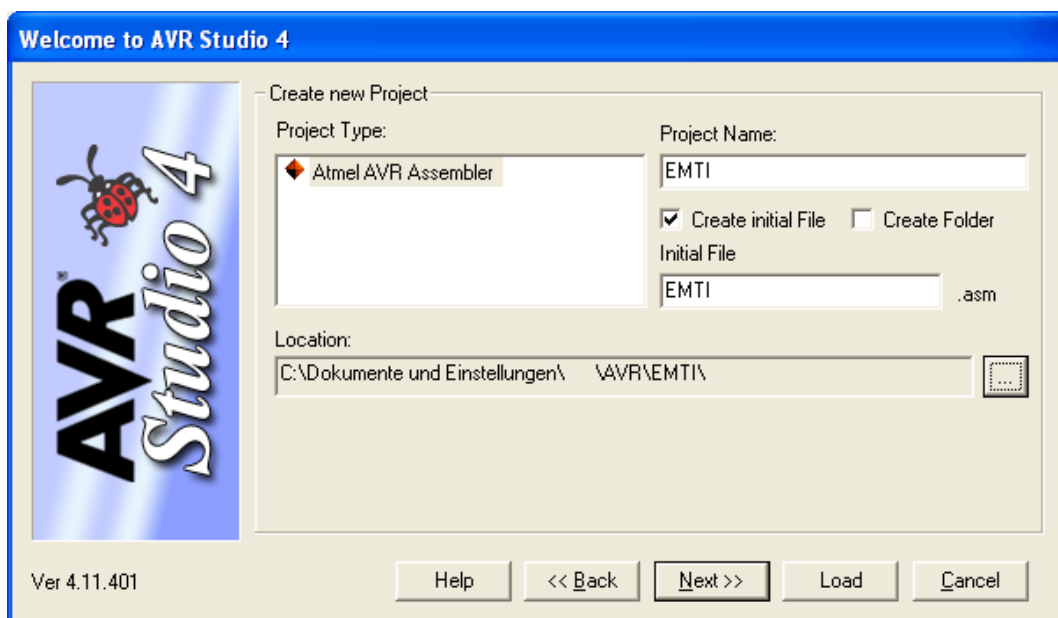
Schritt 2: Der erste Start von AVR-Studio

Beim Start erscheint ein Dialogfeld, mit dem man ein bestehendes Projekt öffnen oder ein neues Projekt anlegen kann. Beim ersten Start müssen wir ein neues Projekt anlegen, und klicken deshalb auf "Create New Project".



Als nächstes ist der Projekttyp festzulegen. Nach der Installation von AVR-Studio gibt es zunächst nur den Projekttyp "Atmel AVR Assembler" zur Auswahl. Bei Bedarf lässt sich später ein C-Compiler (GCC) hinzufügen.

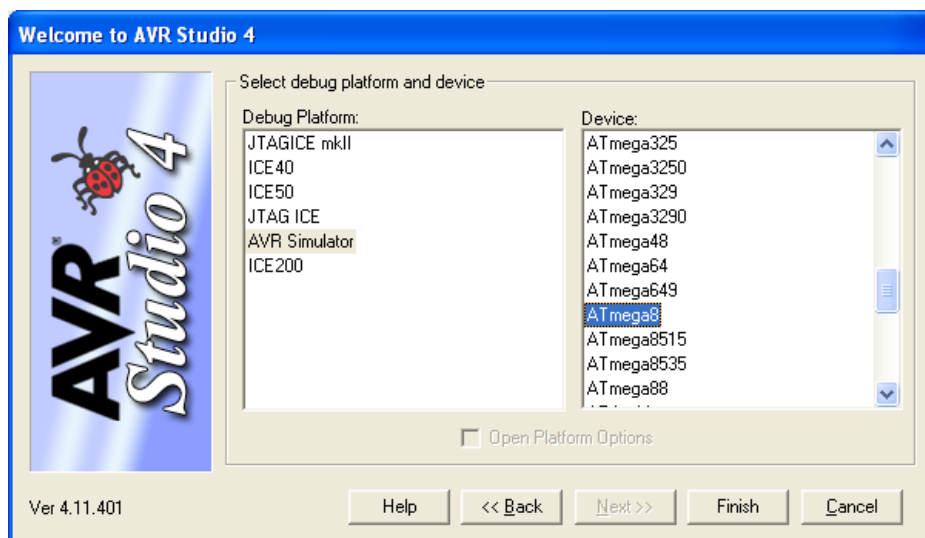
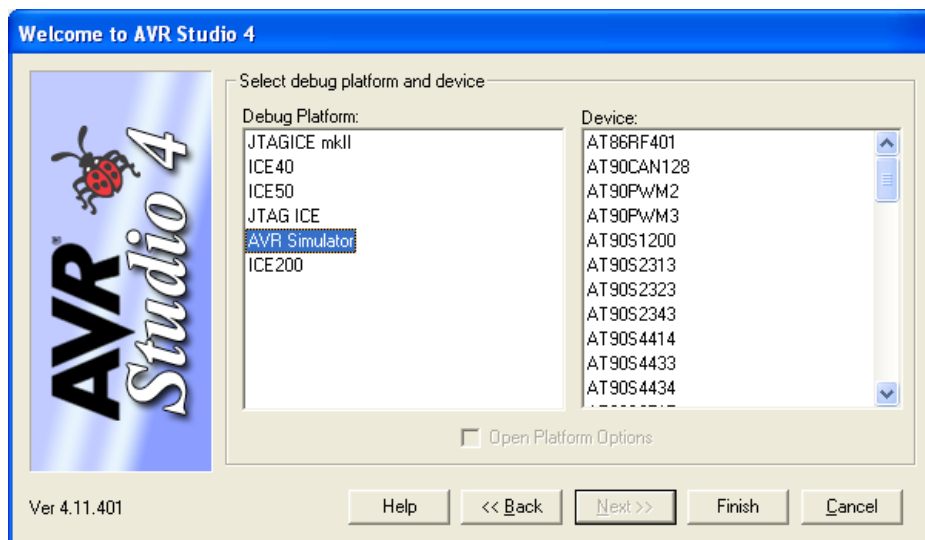
Als Projektnamen wählen wir "EMTI". Das Projektverzeichnis (Location) ändern wir und legen im Benutzerverzeichnis den Ordner "AVR" und darin den Ordner "EMTI" an. Die Auswahl bestätigen wir mit "Next".



Anschließend ist eine Debug-Plattform auszuwählen. Bei Verwendung des AVRISP MKII, der keine Debugger-Funktion enthält, wählt man den "AVR Simulator".

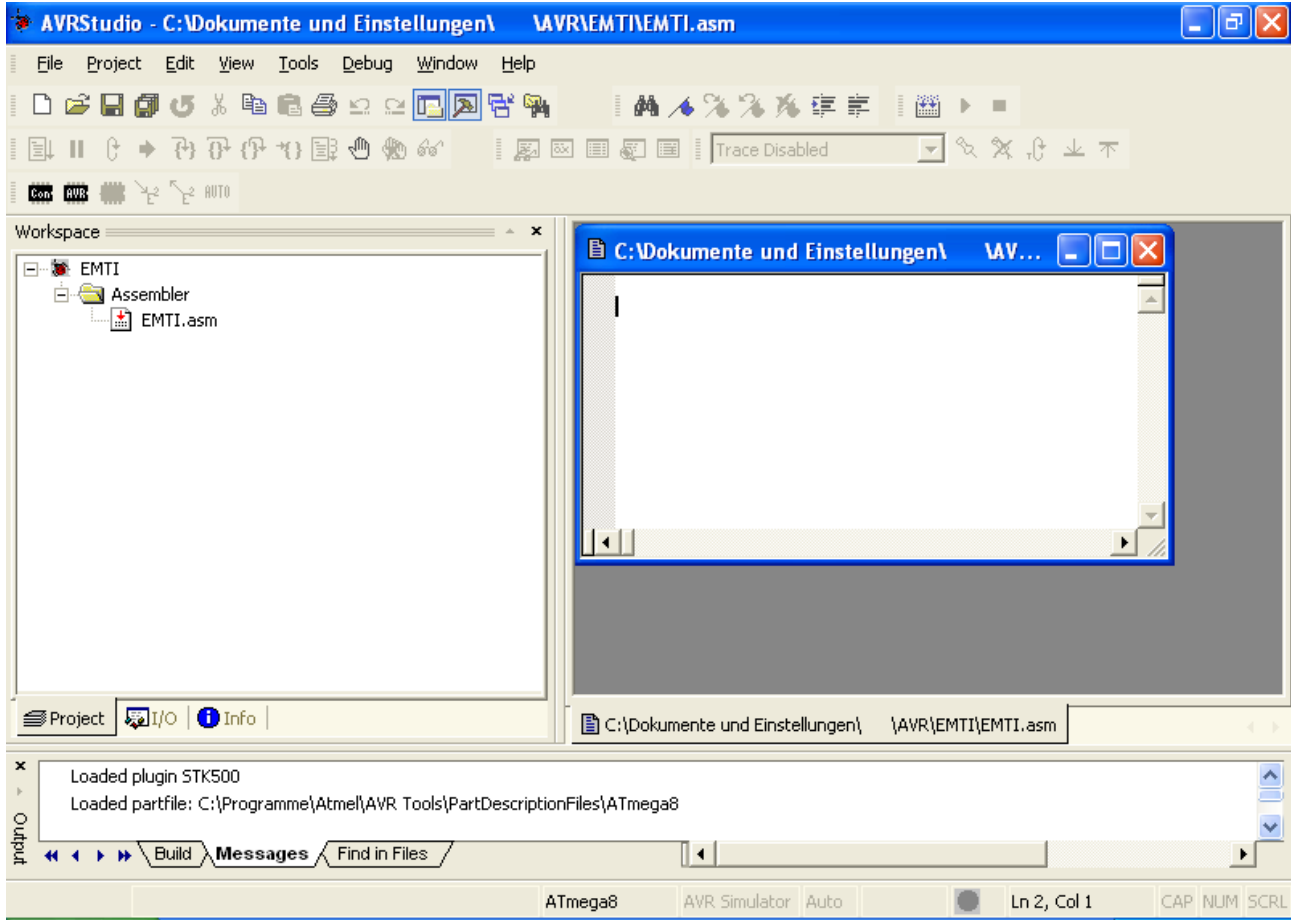
Wer keine Hardware als Zielsystem einsetzt, und die Beispiele nur am Computer ausprobieren möchte, wählt ebenfalls den Simulator.

Als Controller-Typ (Device) wählen wir den ATmega8. Der Controller-Typ lässt sich bei Bedarf später jederzeit ändern.

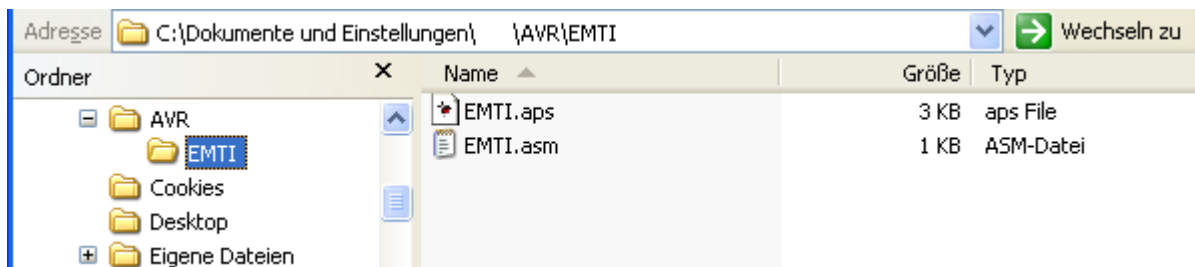


Die Auswahl bestätigen wir mit "Finish".

Wenn alles funktioniert hat, erscheint die Oberfläche des AVR-Studio und ist zur Eingabe eines Programms bereit. Dazu dient das Quelltextfenster rechts im Bild.



Im Projektverzeichnis wurden automatisch die Dateien EMTI.aps und EMTI.asm angelegt:

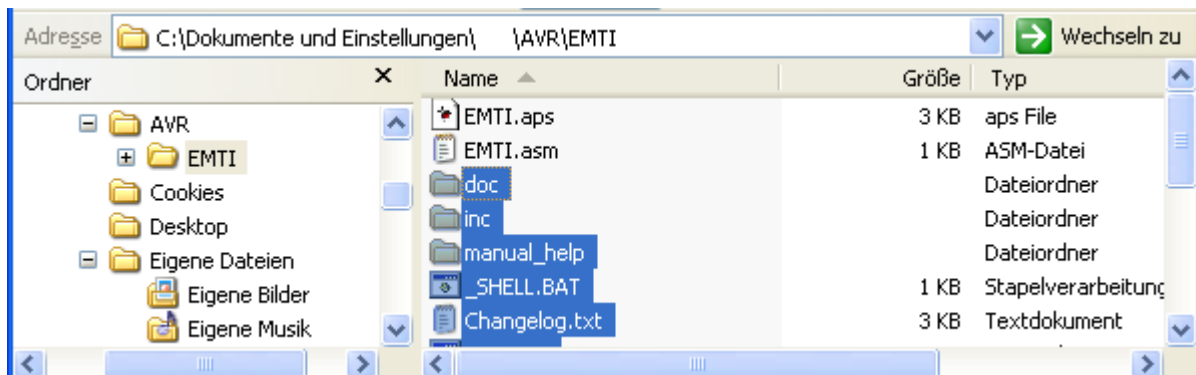


EMTI.aps
EMTI.asm

enthält Konfigurationsdaten zum Projekt
enthält den Assembler Quelltext und ist anfangs leer

Schritt 3: Installation und Konfiguration des EMTI-Compilers

Zur Installation des Compilers genügt es, die Dateien aus dem Download-Archiv (ZIP-Datei) zu entpacken und im zuvor angelegten Projektordner "Benutzername\AVR\EMTI" zu speichern.



Anschließend erstellen wir eine Verknüpfung auf die Datei "_SHELL.BAT" auf dem Desktop und nennen sie "EMTI SHELL".



Ein Doppelklick öffnet ein Eingabefenster, mit dem der Compiler gesteuert wird. Um zu testen, ob der Compiler funktioniert, geben wir folgendes Kommando ein:

```
emti /h
```

Die Ausgabe ist abhängig von der Compiler-Version und sollte in etwa so aussehen:

```
CA\ EMTI SHELL
C:\Dokumente und Einstellungen\ \AVR\EMTI>cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Dokumente und Einstellungen\ \AVR\EMTI>emti /h

*****
EMTI Embedded-Tiny-Compiler
Version 0.04, Build 2010-11-14 (Y-M-D)
(unstable)
Copyright by Sven K. (oog) - www.proog.de
Based on the tutorial 'Let's Build a Compiler' by Jack Grenshaw

Preview Version - Without any warranty

usage: emti <input-filename> [<output-filename>]
*****

C:\Dokumente und Einstellungen\ \AVR\EMTI>
```


Schritt 4: Erstes Testprogramm erstellen

Wir erstellen im Projektverzeichnis eine Textdatei mit dem Namen "blink.txt". Per Doppelklick öffnet sich ein Editor, in dem wir folgendes Programm eintragen:

```
Def cpu="mega8"  
Def KeepRam=1  
  
Begin  
  Pin "B0".ConfigOut  
  
  Repeat  
    Pin "B0".High  
    Pin "B0".Low  
  Again  
End
```

Das Testprogramm speichern wir anschließend ab.

Zur Erklärung:

Def CPU legt fest, dass ein Programm für den ATmega8 compiliert werden soll.

Def KeepRam=1 verhindert, dass der Programmspeicher des ATmega beim Programmstart mit Nullen überschrieben wird. Somit entsteht beim späteren Simulieren des Programms mit AVR-Studio keine unnötige Wartezeit.

Das eigentliche Programm steht zwischen den Schlüsselworten "Begin" und "End".

Pin "B0".ConfigOut schaltet die Richtung von Pin B0 als Ausgang.

Repeat .. Again ist eine Endlos-Programmschleife. Alle Befehle dazwischen werden fortlaufend wiederholt.

Repeat markiert den Anfang der Programmschleife.

Pin "B0".High schaltet Pin B0 auf High-Pegel.

Pin "B0".Low schaltet Pin B0 auf Low-Pegel.

Again springt zurück zu der mit Repeat festgelegten Stelle. Durch den fortlaufenden Wechsel von High und Low an Pin B0 entsteht ein Blinken.

Hinweis: Der EMTI-Compiler unterscheidet nicht zwischen Groß- und Kleinschreibung. Man kann alle Befehle groß oder klein schreiben. Dies gilt auch für selbst definierte Namen (Konstante, Variable, Prozeduren usw.).

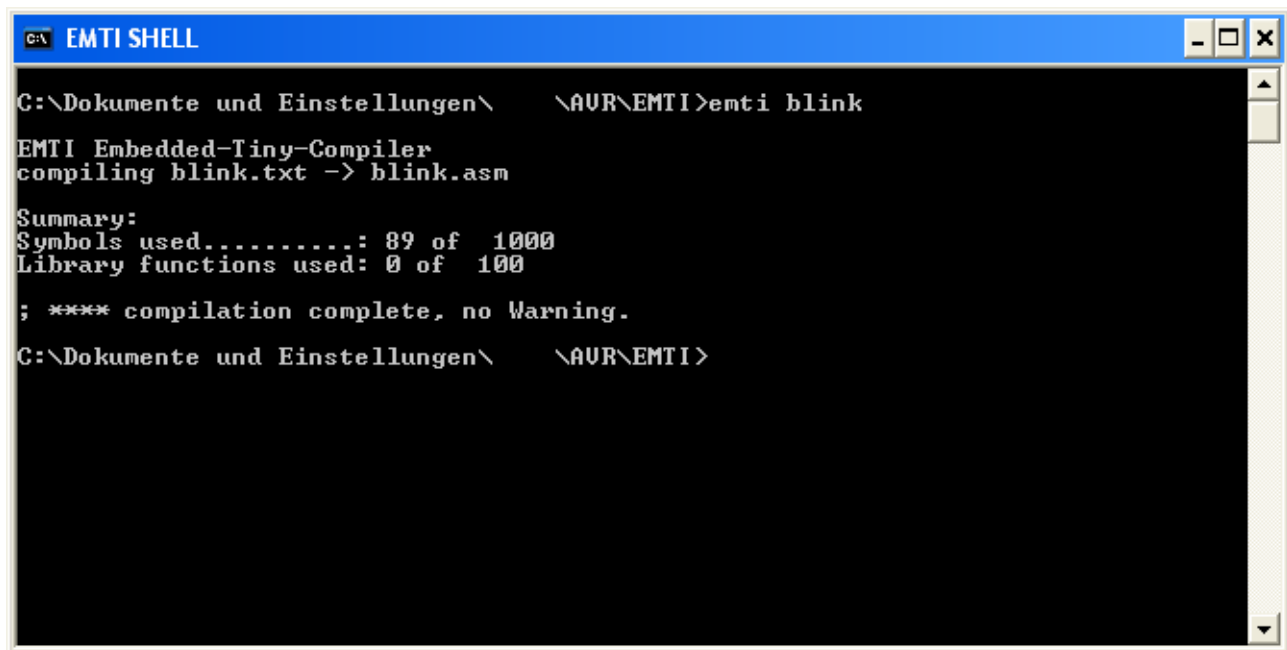
Schritt 5: Testprogramm compilieren

Um das Testprogramm zu compilieren, ist in der EMTI SHELL folgender Befehl einzugeben:

```
emti blink
```

Der EMTI Compiler liest nun den Programm-Quelltext "blink.txt" ein, compiliert ihn und erzeugt als Ausgabe die Datei "blink.asm".

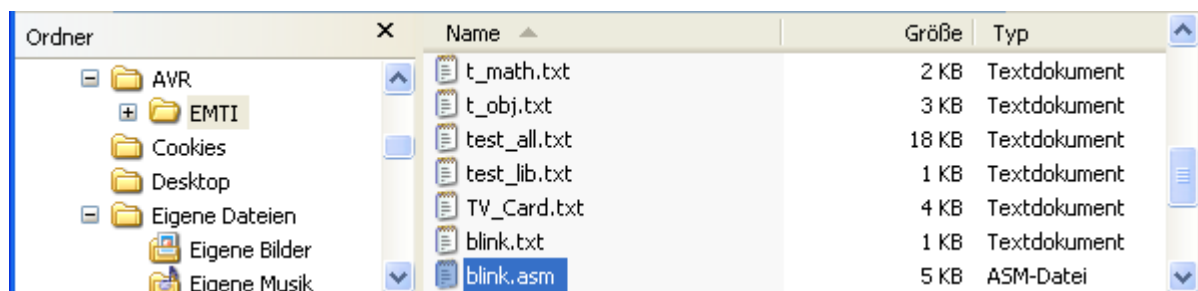
Sollte sich in das Programm "blink.txt" ein Fehler eingeschlichen haben, so erscheint eine Fehlermeldung mit Hinweis, in welcher Programmzeile der Fehler liegt.



```
C:\Dokumente und Einstellungen\ \AVR\EMTI>emti blink
EMTI Embedded-Tiny-Compiler
compiling blink.txt -> blink.asm

Summary:
Symbols used.....: 89 of 1000
Library functions used: 0 of 100

; **** compilation complete, no Warning.
C:\Dokumente und Einstellungen\ \AVR\EMTI>
```



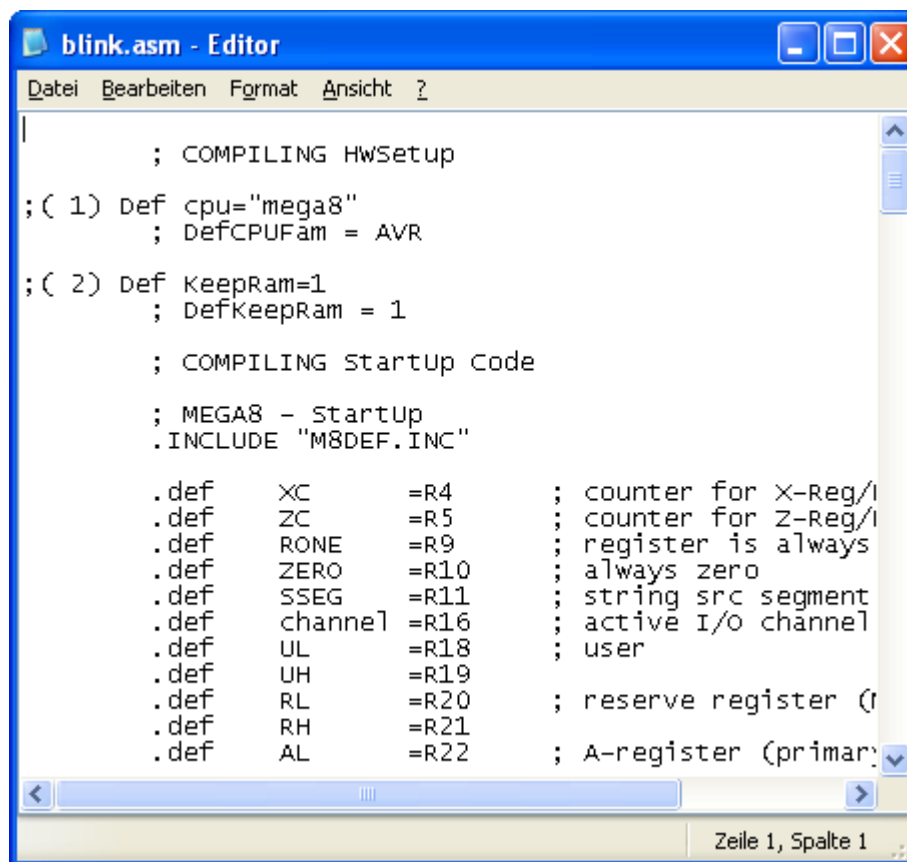
Ordner	Name	Größe	Typ
AVR	t_math.txt	2 KB	Textdokument
EMTI	t_obj.txt	3 KB	Textdokument
	test_all.txt	18 KB	Textdokument
	test_lib.txt	1 KB	Textdokument
	TV_Card.txt	4 KB	Textdokument
	blink.txt	1 KB	Textdokument
	blink.asm	5 KB	ASM-Datei

Hinweis: Die unter "Summary" angezeigten Funktionen zeigen Compiler-interne Ressourcen an und sind für unser Programm bedeutungslos. Die Anzeige wird in späteren Compiler-Versionen verschwinden.

Schritt 6: Testprogramm assemblieren

Um das Testprogramm zu assemblieren, muss es in AVR-Studio übertragen werden.

Eine Möglichkeit ist, den Assemblertext "blink.asm" mit dem Editor zu öffnen, zu kopieren und über die Zwischenablage in AVR-Studio einzufügen.



```
blink.asm - Editor
Datei Bearbeiten Format Ansicht ?

; COMPILING HWSetup
;( 1) Def cpu="mega8"
; DefCPUFam = AVR
;( 2) Def KeepRam=1
; DefKeepRam = 1

; COMPILING Startup Code

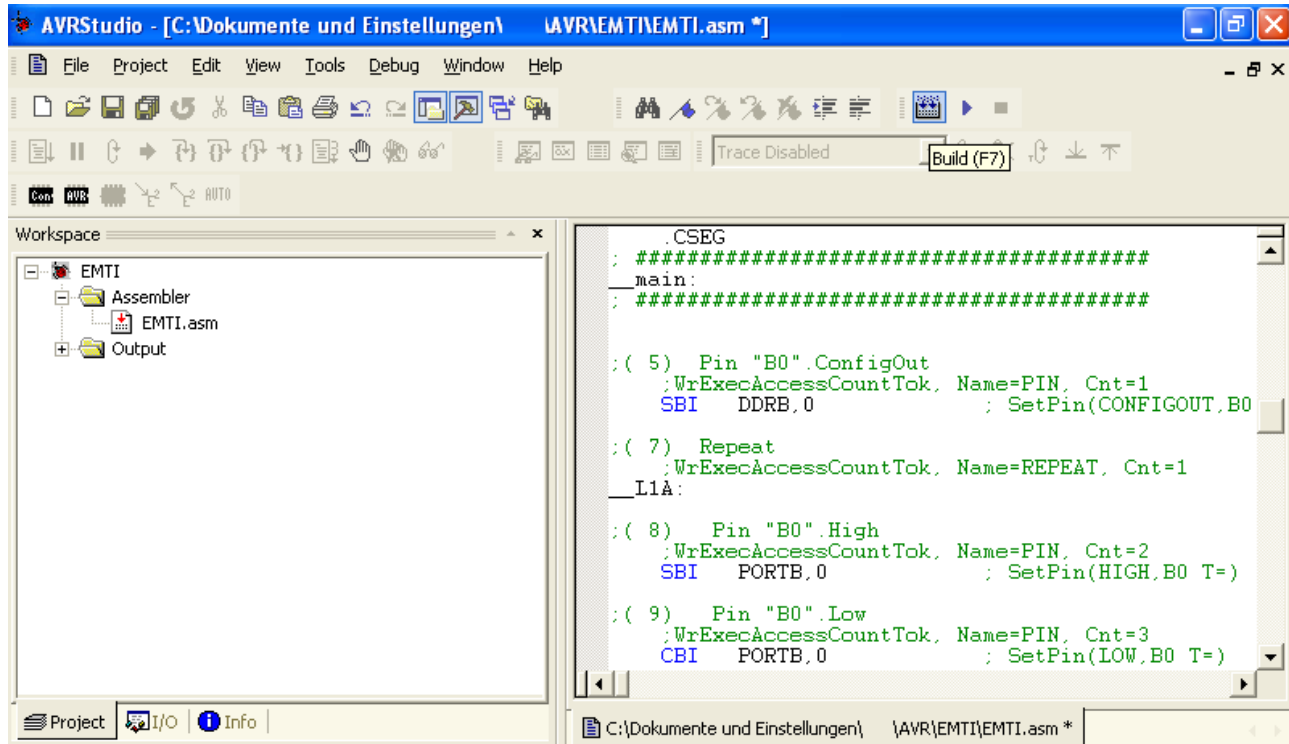
; MEGA8 - Startup
.INCLUDE "M8DEF.INC"

.def XC =R4 ; counter for X-Reg/I
.def ZC =R5 ; counter for Z-Reg/I
.def RONE =R9 ; register is always
.def ZERO =R10 ; always zero
.def SSEG =R11 ; string src segment
.def channel =R16 ; active I/O channel
.def UL =R18 ; user
.def UH =R19
.def RL =R20 ; reserve register (r
.def RH =R21
.def AL =R22 ; A-register (primar
```

Zeile 1, Spalte 1

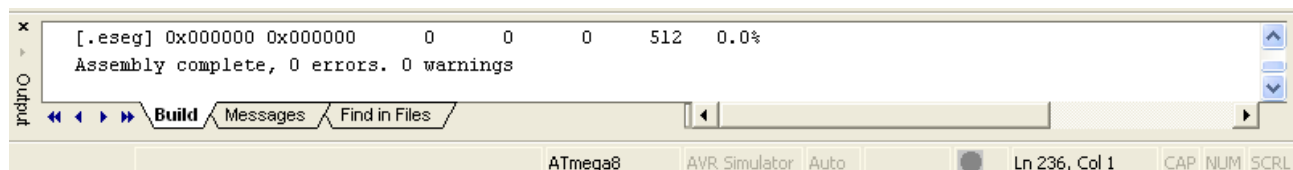
Öffnen wir also den Quelltext mit dem Editor, markieren den gesamten Text (STRG+A) und kopieren ihn in die Zwischenablage (STRG+C).

Dann wechseln wir zum AVR-Studio und platzieren den Cursor in das Quelltextfenster auf der rechten Seite. Dann fügen wir den kopierten Text ein (STRG+V).



Als nächstes assemblieren wir den Quelltext mit "Build". Der entsprechende Button ist auf dem oberen Screenshot markiert. Er liegt mittig über dem Quelltext.

Wenn die Assemblierung erfolgreich verläuft, erscheint im Output Fenster unten die Meldung "0 errors, 0 warnings".



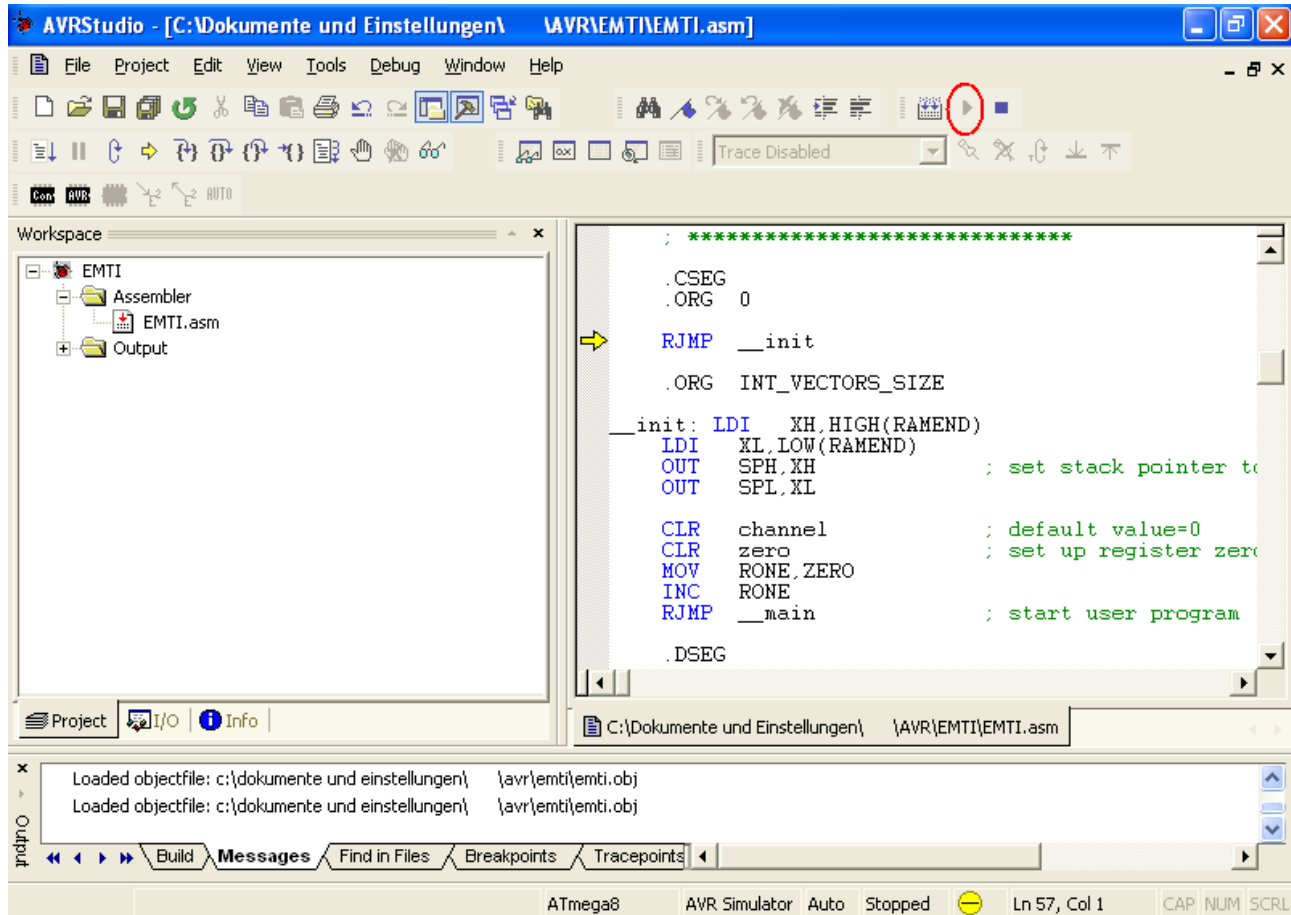
Anmerkung

Da der EMTI-Compiler keine Ressourcenüberwachung durchführt, kann es sein, dass ein zu großes Programm oder zu groß dimensionierte Variablen erst vom Assembler bemerkt werden können.

Sollte dies passieren, kann man einen Mikrocontroller mit mehr Speicher einsetzen oder versuchen, das Programm zu kürzen oder zu optimieren.

Schritt 7: Testprogramm simulieren

Nach erfolgreichem "Build" startet man den Simulator durch Klick auf das kleine Dreieck rechts neben dem Build-Button. Es ist im Screenshot mit einem roten Kreis markiert.



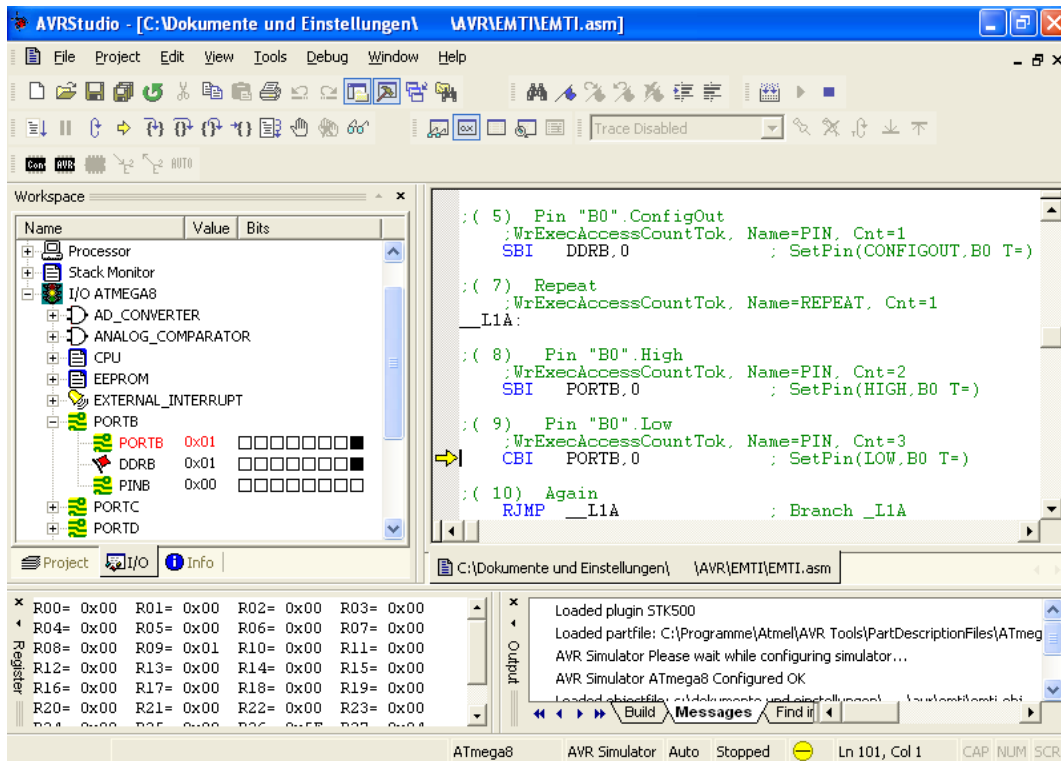
Das Fenster verändert sich. Im Quelltextfenster erscheint ein gelber Pfeil, welcher den Program Counter darstellt. Beim Druck der Taste F10 (oder F11, siehe Online-Hilfe zum AVR-Studio) wird der mit dem Pfeil markierte Befehl ausgeführt.

Um den Programmablauf besser verfolgen zu können, kann man sich die CPU-Register, verschiedene Speicherbereiche sowie I/O-Devices anzeigen lassen.

Die Trennlinien zwischen den Fenstern lassen sich mit der Maus verschieben, um die Bildschirmaufteilung den eigenen Bedürfnissen anzupassen.

Im linken Fenster "Workspace" kann man mit dem Kartenreiter "I/O" die Ansicht für I/O-Devices einschalten. Zur Prüfung, ob Pin B0 korrekt blinkt, ist die Ansicht für PORTB einzuschalten.

Weitere Ansichten lassen sich über das Menü "View" ein- und ausschalten.



Mit der Taste F10 kann man nun das Testprogramm Schritt für Schritt ablaufen lassen. Im Quelltextfenster sieht man die Zeilen aus dem Testprogramm in grüner Schrift, mit Angabe der Zeilennummer in Klammern. Im linken Fenster sieht man die Signale von PORTB.

Die Programmzeile (5) Pin "B0".ConfigOut
wurde übersetzt in SBI DDRB,0

Gut zu erkennen ist auch die Repeat .. Again-Schleife, repräsentiert durch Sprungmarke (__L1A:) und Sprungbefehl (RJMP __L1A).

Da der EMTI-Quelltext als Kommentar angezeigt wird, lässt sich der Programmablauf während der Simulation gut verfolgen.

Ich gratuliere zum ersten erfolgreich ausgeführten EMTI-Programm.

Sven K. (oog)