# Erste Schritte

# Programmierung Honeywell HR20

# mit

# Avr Studio 4.13
# LCD Visualisierung
# WinAVR 20071221

| | |
|---|---|
| Version: | 1.0 |
| Datum: | 19.02.2008 |

## Historie:

| Version | Datum | Bemerkung |
|---|---|---|
| 1.0 | 19.02.08 | Erstellung |

# Index

# 1 Einführung

Das Heizkörperthermostat HR20 von Honeywell ist ein elektronisches Thermostat, welches Heizkörperventile ansteuert um die Raumtemperatur zu regeln. Die vorhandene Hardware wird genutzt und mit der folgenden Anleitung ist es möglich die Software für den Mikrocontroller zu programmieren und zu simulieren.

# 2 Vorraussetzung

Die Vorraussetzung für die Programmierung sind, dass die folgenden Programme ordnungsgemäß installiert sind:

- Entwicklungsumgebung: AVR Studio 4 mit Service Pack 2 (V4.13. build 571) [1]
- Compiler: WinAVR V20071221 [2]
- LCD Simulation: AVR LCD Visualizer [3]
- Bei Bedarf auch bereits vorhandenen Sourcecode [4]

# 3 Projekt anlegen

Als erstes muss ein Projekt um AVR Studio angelegt werden. Hierzu verfährt man wie folgt:

1) AVR Studio 4 starten
2) Aus dem Project Menu *New Project* auswählen

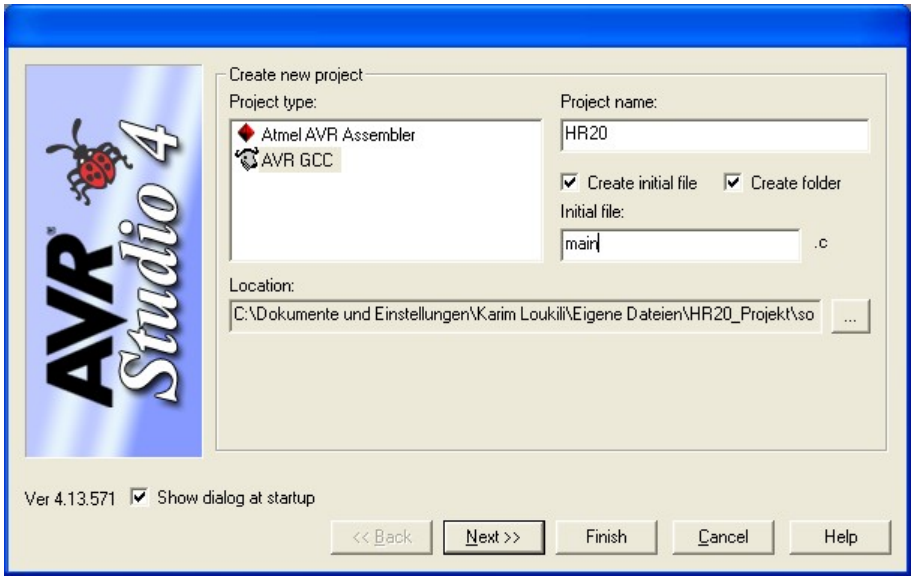


**Abbildung 1: Create new project**

Um ein C Projekt zu erzeugen, muss *AVR GCC* selektiert werden. Bei *Project name* ist ein Name anzugeben (z.B.: HR20) und falls eine leere Datei erzeugt werden soll, muss im Feld *Initial file* der Name angegeben und der Haken bei *Create initial file* gesetzt sein.

Unter *Location:* wird der Pfad gesetzt, wo das Projekt angelegt werden soll. Wenn die Auswahl *Create folder* markiert ist, wird unter diesem Pfad ein Unterverzeichnis mit dem Projektnamen angelegt.

Sind schon Quelltexte vorhanden (z.B. aus dem SVN[4]), wird nur der Projektname gesetzt und das Zielverzeichnis angegeben. Das AVR Studio muss in dem Fall kein *Initial File* anlegen

3) Prozessor wählen
   mit *Next>>* bekommt man die Gelegenheit, die Debug Plattform und den Mikrocontroller des Zielsystems festzulegen.

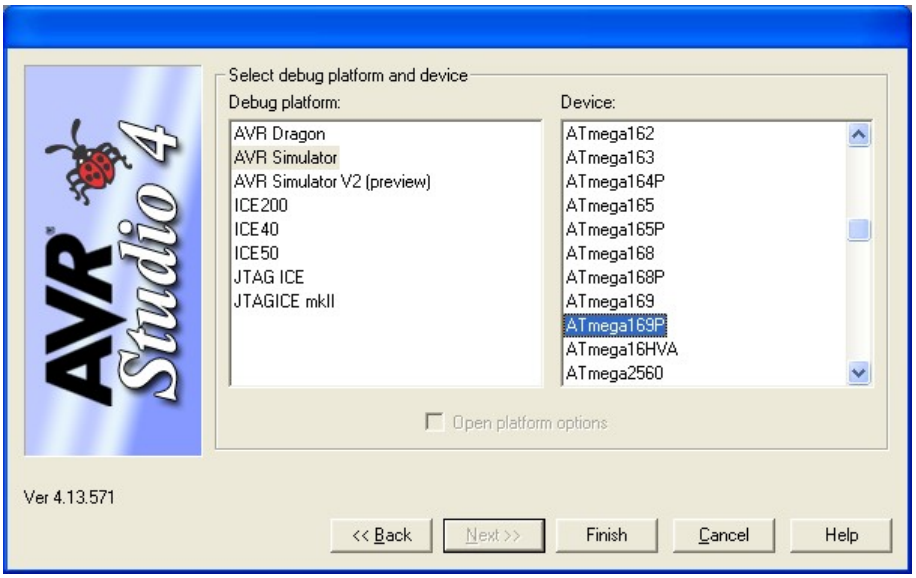


**Abbildung 2: Select debug platform and device**

Auf dem Zielsystem (HR20E) läuft ein *ATmega169P* der unter *Device:* ausgewählt wird. Bei der *Debug plattform* kann man zunächst die *AVR Simulation* angeben. Später wählt man aus dem Menü *Debug\Select Platform and Device…* die einzusetzende Debug Platform (ICE, JTAG, Dragon…).

4) Projektkonfiguration einstellen

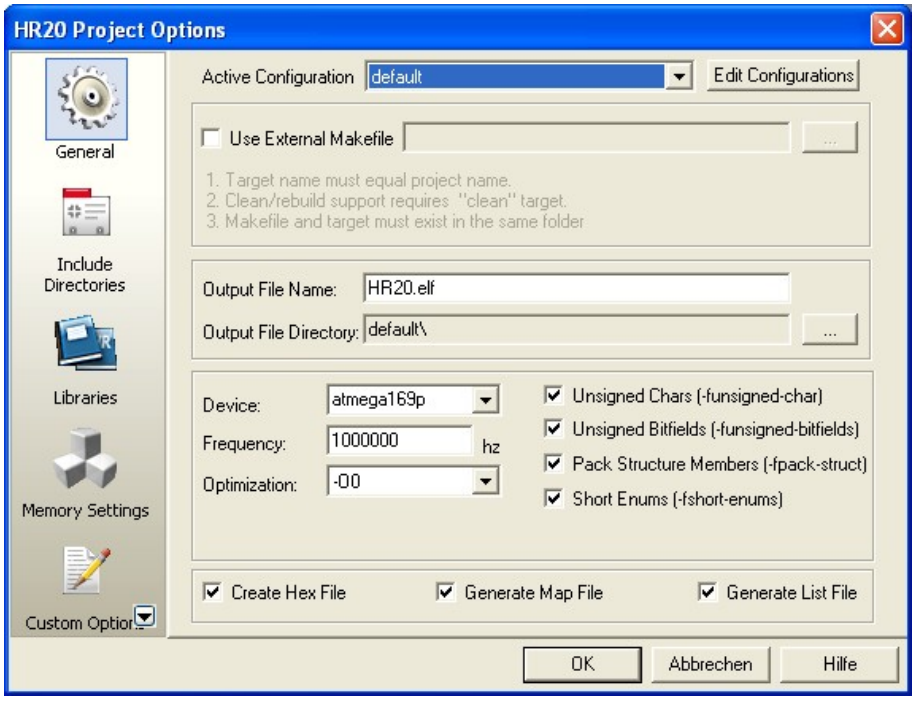


**Abbildung 3: Project options**

Aus dem Menü *Project\Configuration Options…* stellt man unter *Device:* noch einmal Controller ein, trägt darunter die Taktfrfequenz (4000000 für 4MHz) ein und schaltet am Besten vorerst die Optimierung ab (*-O0*), damit man beim Debuggen keine Probleme mit wegoptimierten Code bekommt. In kritischen Codesequenzen kann man aber auf Optimierung angewiesen sein. In der Regel bietet der Compiler für solche Fälle ein Pragma an, dass man auf Abschnitte anwenden kann um die Optimierung zu erzwingen.

# 4 Dateien dem Projekt hinzufügen

Nun, da ein Projekt angelegt ist, kann man die bestehenden Quelltexte (z.B.: aus dem SVN[4]) wie folgt dem Projekt hinzufügen:

Im Kontext Menü des *Source Files* Verzeichnises werden mit *Add Existing Source File(s)* bestehende Datei hinzugefügt. Hier fügt man die herunter geladenen .c Dateien aus.

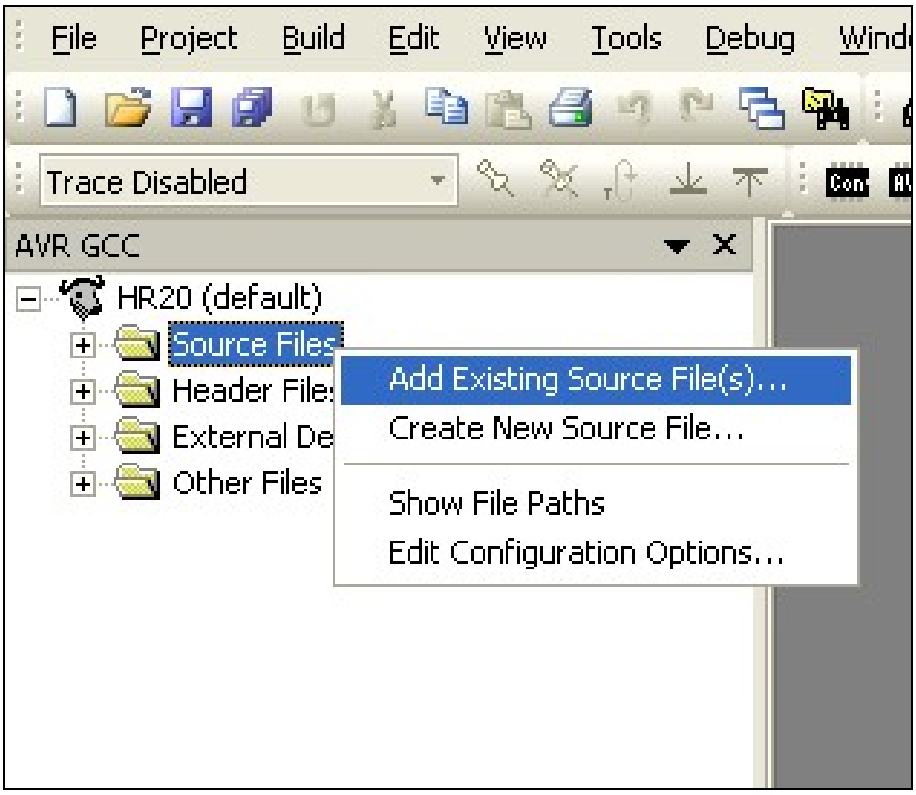


**Abbildung 4: Add Existing Source Files(s)**

Die Header-Dateien des Projektes werden im Header Files Kontextmenü dazu geladen.

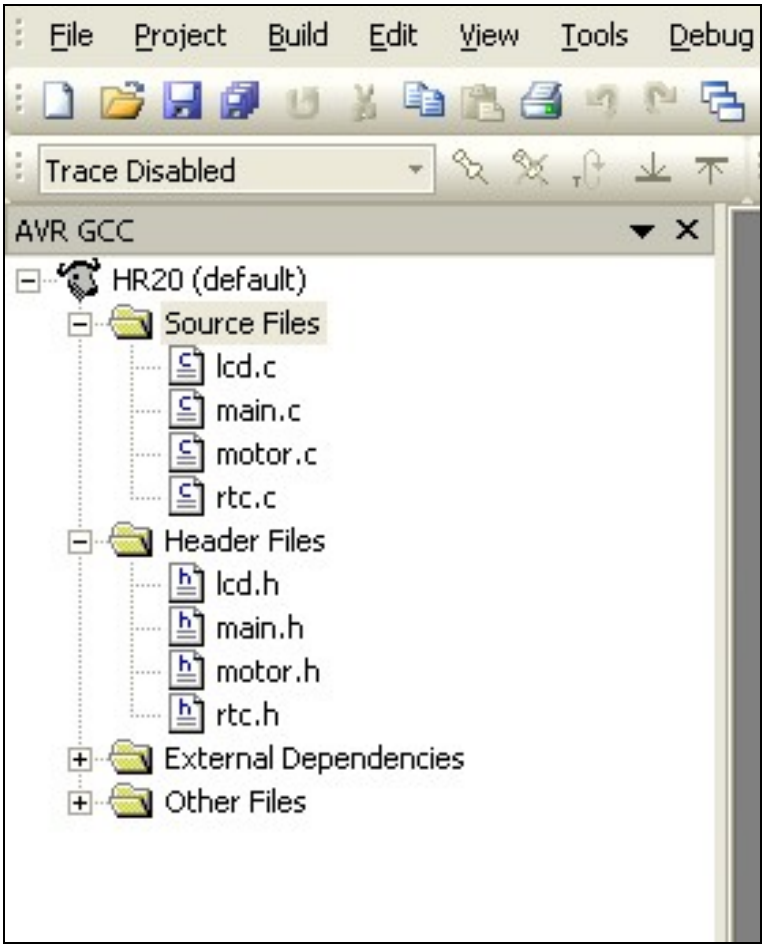


**Abbildung 5: Source files**

Zum Projektstand vom 17.02.2008 reicht obige Konfiguration ausgehend von den Defaults nach der Installation, um einen erfolgreichen Build auszuführen und die SW im Simulationsmodus zu starten. Im weiteren Projektverlauf, können hier weitere Einstellungen nötig werden.

# 5 Simulation mit dem LCD Visualizer

Zur Simulation muss das AVR Visualizer Plugin installiert sein.

Als erstes muss man mit dem AVR LCD Editor ein LCD Display File mit allen nötigen Dateien erzeugen. Eine Beschreibung dazu findet man unter Tool\AVR LCD Visualizer\Help.

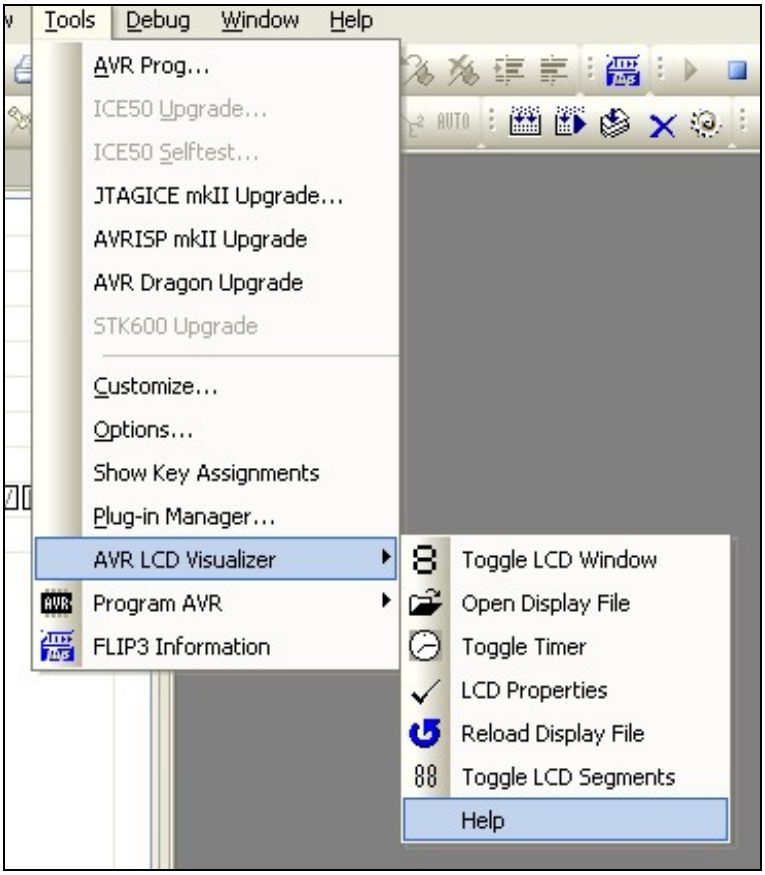


**Abbildung 6: LCD Visualizer Help**

Wenn man die bestehenden Sourcen aus dem SVN [4] benutzt, dann befindet sich in dem Quellverzeichnis auch ein Verzeichnis *LcdEdit* in dem sich bereits die Konfigurationsdaten für das HR20 befinden.

Hat man eine LCD Konfiguration, startet man die Simulation im AVR Studio z.B. mit F5. Daraufhin wird das Projekt kompiliert und gestartet, der Debugger bleibt dann am Anfang der main Funktion stehen.

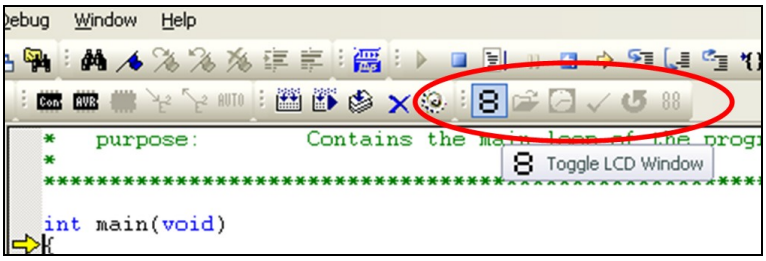


**Abbildung 7: LCD Visualizer Control panel**

Im LCD Funktionsblock (siehe rotes Oval oben) klickt man auf die 8 und danach rechts daneben auf das *Datei Öffnen* Symbol. Für den HR20 ist die bestehende Displaydatei *(…HR20_Projekt\source\LcdEdit\hr20.dis)* zu öffnen.

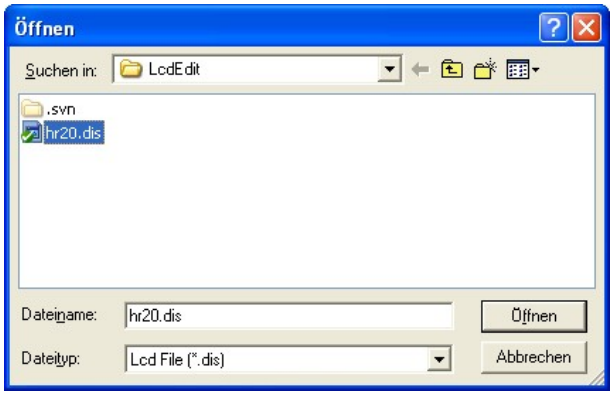


**Abbildung 8: LCD Visualizer Load Configuration**

Das Display wird dann in einem Panel angezeigt. Am besten ist es, wenn man es an eines der umliegenden Fenster des Studios ankert, indem man in die Titelleiste klickt und auf eines der Ankerfelder die dann beim ziehen erscheinen abwirft.

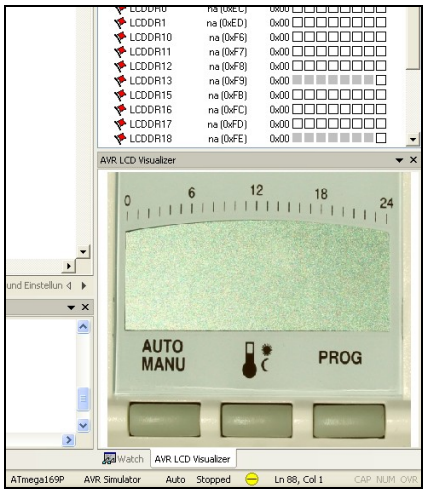


**Abbildung 9: LCD Visualizer Screen**

Hat man im Code alles richtig angesteuert, sieht man im Display die Ausgaben. Damit diese auch bei laufender Programmausführung angezeigt werden, sollte unbedingt der Toggle Timer eingeschaltet werden.

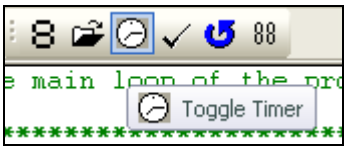


**Abbildung 10: LCD Visualizer Toggle Timer**

Hat man alles richtig eingestellt, sieht das dann so aus:


**Abbildung 11: LCD Visualizer active**

# 6 Referenzen

[1] AVR Studio 4, Version 4.13.528 + Service Pack 2 Build 571
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

[2] WinAVR, Version 20071221
http://winavr.sourceforge.net/

[3] AVR LCD Visualizer
http://atmel.com/dyn/products/tools_card.asp?tool_id=2725
http://atmel.com/dyn/resources/prod_documents/AvrLcd.msi

[4] HR20 SVN
Repository Location: https://OpenSVN.csie.org/hr20
Browse: https://OpenSVN.csie.org/viewcvs.cgi/?root=hr20
TRAC: https://opensvn.csie.org/traccgi/hr20