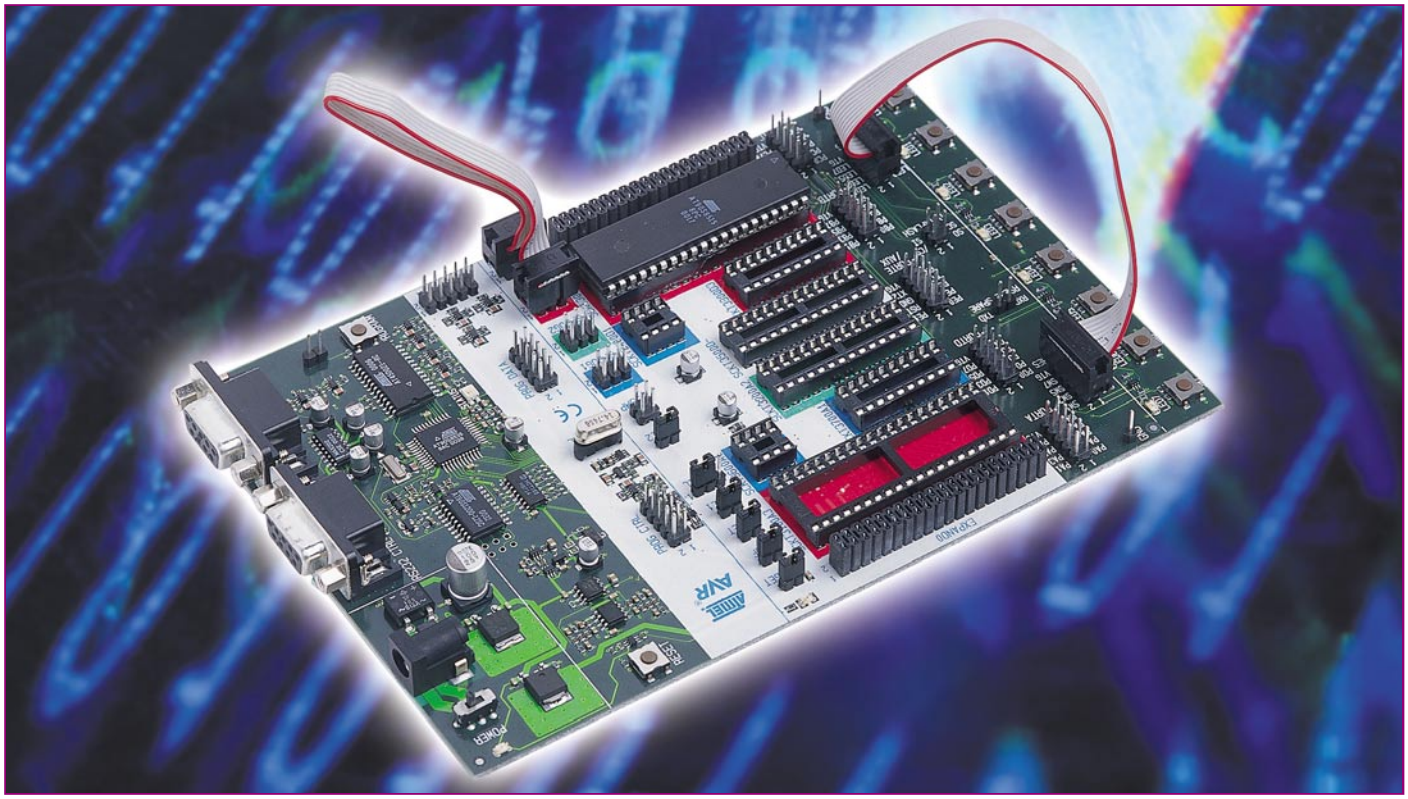

AVR-Grundlagen Teil 5

Nach der Realisierung des ersten Beispielprojektes mit Hilfe des AVR-Starterkits verschaffen wir uns jetzt eine Übersicht über einige gängige C-Compiler für die AVR-Mikrocontroller, die in vielen Anwendungsfällen die Programmierung einfacher gestalten und somit Projekte schneller ans Ziel bringen.

Einfacher programmieren

Wie bereits im ersten Teil der Artikelserie erwähnt, hat man von vornherein den Befehlssatz der AVR-Mikrocontroller für den Einsatz entsprechender C-Compiler optimiert. Da die maschinennahe Programmierung mit dem Assembler vielen Anwendern sehr schwer fällt und da ohnehin der Einsatz einer Hochsprache das Erstellen einer Software stark vereinfacht, wollen wir an dieser Stelle zwei gängige C-Compiler für den AVR-Mikrocontroller vorstellen.

Setzt man einen solchen Compiler ein, wird das Programm für den Prozessor in der von den Meisten wohl als einfacher empfundenen Programmiersprache „C" verfasst. Dies hat den Vorteil, dass der Quelltext des Programmes nicht nur sehr viel leichter zu erarbeiten, sondern auch die Kontrolle effizienter durchführbar ist. Multiplikationen, Divisionen und andere mathematische Routinen werden einfach, wie man es von der Programmiersprache C vom PC her kennt, implementiert - hingegen muss man im Assembler jede Routine mühsam selbst erstellen.

Funktion

Was ein Compiler eigentlich macht, lässt sich leicht erklären: Er übersetzt (to compile = übersetzen, kompilieren) den C-Quellcode in den entsprechenden Assemblercode, der dann direkt in die vom Prozessor verarbeitbaren Maschinenbefehle umgesetzt wird. Das heißt, dass für jede in C programmierte Funktion die entsprechende Assemblerfunktion integriert wird. Der erzeugte Assemblercode wird in Listen-Dateien mit der Endung „.lst" gespeichert, sodass man jeden einzelnen Schritt per Editor genau nachverfolgen kann.

Ein großer Vorteil bei der Verwendung einer Hochsprache ist der, dass die gesamte Verwaltung der Variablen im Speicherbereich vom Compiler übernommen wird, sodass Fehler in diesem Bereich leichter vermeidbar sind.

Compiler

Es gibt sehr viele C-Compiler, die für die AVR-Mikrocontrollerfamilie von Atmel einsetzbar sind. Eine Übersicht darüber findet man unter [1] im Internet. An dieser Stelle soll anhand von zwei ausgewählten Produkten gezeigt werden, wie unterschiedlich die Übersetzungsprogramme gestaltet sein können. So hat jeder die Wahl, den passenden Compiler entsprechend den eigenen Anforderungen selbst auszuwählen.

Der GCC-AVR-Compiler

Der GNU-C-Compiler für AVR ist eine Portierung des bekannten GNU-C-Compilers und hat zunächst den großen Vorteil, dass er als Freeware im Internet zur Verfügung steht [2]. Dieser Compiler ist jedoch nicht unbedingt für den Anfänger geeignet, da man alle Einstellungen manuell vornehmen muss. Zur Vereinfachung des Einstiegs in die Programmierung mit dem GCC-AVR-Compiler finden sich ebenfalls unter [2] diverse Testprogramme (gcctest), mit denen man sehr schnell Erfolge erzielen kann.

In der folgenden kurzen Anweisung wollen wir beispielhaft die Installation und die

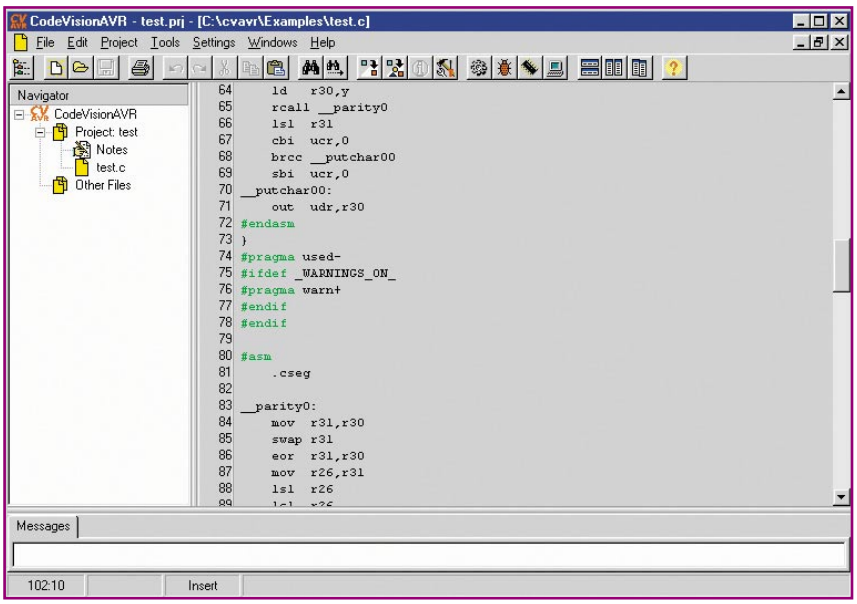

Bild 1: Entwicklungsumgebung CodeVision AVR C-Compiler

Kompilierung eines der Testprogramme schrittweise erklären.

Zuerst wird der AVR-GNU-C-Compiler über das mit heruntergeladene Installationsprogramm installiert. Dann startet man im Installationsverzeichnis die Datei „run.bat“ um die Umgebungsvariablen für das System zu setzen. Jetzt werden die Testprogramme mit einem entsprechenden Entpacker in das gewünschte Verzeichnis (z. B. „C:\“) entpackt.

Im Anschluss daran wechselt man in eines der entpackten Verzeichnisse (z. B. „gcctest1“), wo drei Dateien zu finden sein sollten: Der Quelltext des Programms hat die Dateiendung „*.c“; dann folgt das fertig vorkompilierte Ergebnis „*.rom“ sowie eine Datei namens „makefile“. Letztere enthält die Anweisung für die Übersetzung des Programms. Diese Datei ist leicht für eigene Projekte anpassbar, da für jede wichtige Zeile der zugehörige Kommentar eingefügt wurde. Das Kompilieren des Programms erfolgt durch den Aufruf der Datei „make.exe“, welche durch das Setzen der Umgebungsvariablen direkt verfügbar ist. Beim Übersetzen entsteht eine „*.obj“-Datei (z. B. gcctest1.obj), die ihrerseits direkt vom AVR-Studio lesbar ist.

Mittels des AVR-Studios und des Starterkits STK500 lässt sich das Programm ohne Probleme in den Mikrocontroller programmieren. Dazu öffnet man die Datei über das Menü „File → Open“, wählt sie im erscheinenden Dialogfenster entsprechend aus und bestätigt den Vorgang mit dem „OK“-Button. Jetzt wird über „Tools → STK500“ der Dialog zum Programmieren gestartet. Bevor man jedoch das Programm in den Mikrocontroller programmieren kann, sind noch einige Einstellungen vorzunehmen.

Zuerst wird unter „Device“ der Typ des verwendeten Controllers (beispielsweise AT90S8515), dann der entsprechende Programmiermodus (Programming mode) angewählt, und schließlich, bevor das Programmieren des Controllers über den „Program“-Button erfolgt, erfolgt die Auswahl der aktuellen Datei als Quelle („Use Current Simulator/ Emulator Flash Memory”).

Ein Nachteil dieses Compilers ist allerdings, dass er keine Windows-Oberfläche besitzt, sondern allein in der DOS-Ebene arbeitet.

Der CodeVision AVR C-Compiler

Der zweite C-Compiler, den wir hier betrachten wollen, ist der „CodeVision AVR C-Compiler”, den man in einer eingeschränkten Version unter [3] aus dem Internet herunterladen kann. Die hauptsächliche Einschränkung der sonst frei verfügbaren Version ist die Limitierung der übersetzbaren Codelänge.

Die Installation des Programms erfolgt direkt unter MS-Windows. Danach steht eine ansprechende und funktionell durchdachte Entwicklungsumgebung (Abbildung 1) zur Verfügung. Diese bietet, neben den gewohnten Features, noch einen „CodeWizard“ (Abbildung 2), welcher bei der Generierung eines neuen Projektes hilft. Sehr viele Hardwareparameter sind mittels dessen grafischer Oberfläche einfach und übersichtlich konfigurierbar und werden direkt in den Quellcode übernommen, so-

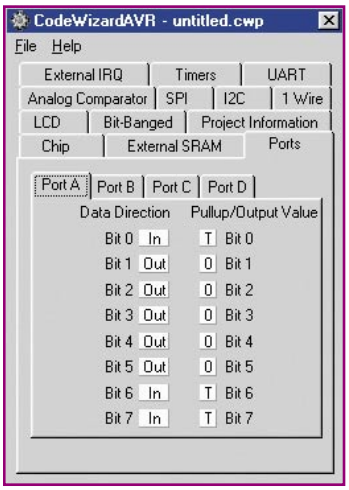

Bild 2: CodeWizard

dass einige Zeit gespart werden kann. Zusätzlich werden diverse Kommentare automatisch erzeugt, die beim Programmieren sehr hilfreich sein können.

Außerdem sind die unterstützten Mikrocontroller direkt über einen zugehörigen Dialog und das entsprechende Tool, z. B. Atmel Starterkit STK 500, programmierbar.

Programmieren in C - die Tücken

Die Programmierung in der Hochsprache C macht die Programmentwicklung sehr viel einfacher und schneller, jedoch ist an einigen Stellen Vorsicht geboten, wenn man das Programm für einen Mikrocontroller erstellt. Denn die meisten Mikrocontroller verfügen nur über eine begrenzte Anzahl von Ressourcen bzw. On-Chip-Peripherie, was besonders beim RAM (Arbeitsspeicher) von großer Bedeutung ist. Bei der Programmierung in C neigt man aufgrund der übersichtlichen Programmstruktur aber oft dazu, große Felder oder komplizierte Datenstrukturen anzulegen, die dann unter Umständen große Bereiche des Datenspeichers belegen. Der Datenspeicher wird jedoch auch für den Stack verwendet, auf dem das Programm, z. B. bei Funktionsaufrufen, die Rücksprungadresse sowie weitere wichtige Daten ablegt. Aus diesem Grunde sollten die Unterfunktionen des Programms nicht in zu vielen Ebenen verschachtelt werden, sodass es nicht wegen des begrenzten RAMs dazu kommen kann, dass der Stack die Daten im Bereich der Variablen überschreibt. Die Folge ist ein nicht mehr weiter arbeitsfähiges Programm. Fehler, die durch einen Stacküberlauf bedingt sind, sind häufig schwer zu lokalisieren, also gilt auch hier die allgemeine Regel, effizient und übersichtlich zu programmieren.

Nachdem wir mit dem Compiler nun ein weiteres wichtiges Werkzeug für das Programmieren der AVR-Controller kennengelernt haben, werden wir im nächsten Teil der Artikelserie ein einfaches Programmiergerät für die gängigen AVR-Mikrocontroller vorstellen, sodass man für deren Programmierung nicht mehr des AVR-Starterkits bedarf.

Internet:

[1] Übersicht C-Compiler
http://www.atmel.com/atmel/products/prod205.htm#COMPILERS

[2] AVRfreaks.net
GCC AVR – Gnu C-Compiler für AVR
http://www.avrfreaks.net/AVRGCC/

[3] CodeVision AVR C-Compiler
http://infotech.ir.ro/html/download.htm