Professionelle Softwareentwicklung für Mikrocontroller

DAvE - Digitaler Applikationsingenieur

Inbetriebnahme des C167CR Starter Kits

Wilhelm Brezovits

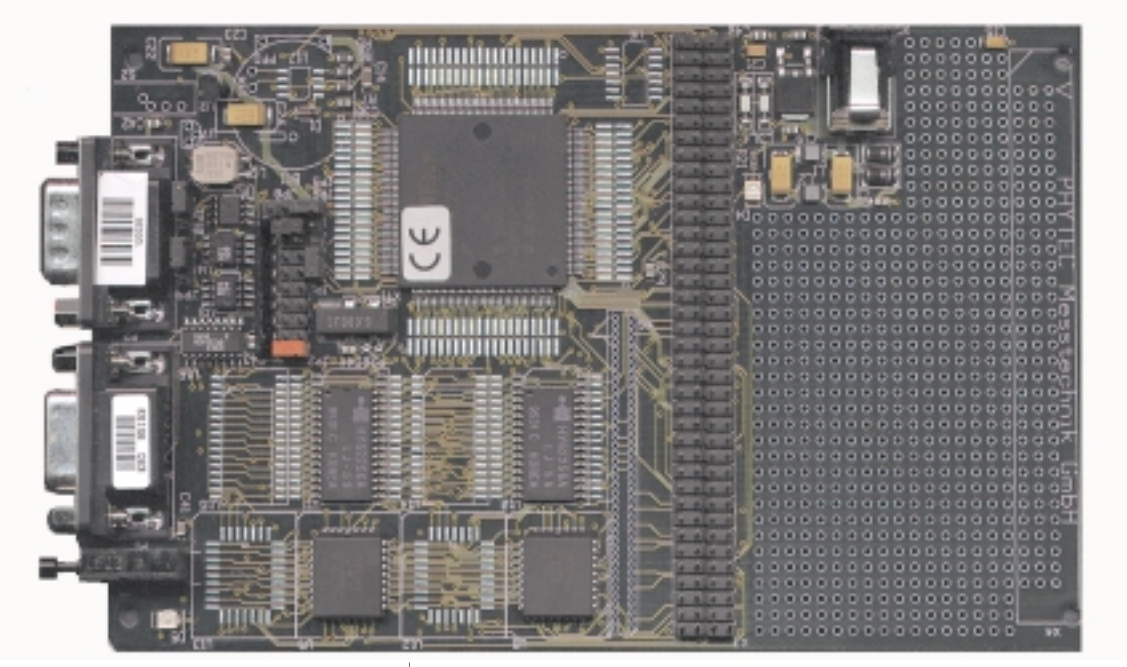
Dieser Artikel soll zu einem Erfolgserlebnis beitragen.

Als Erfolgserlebnis definieren wir eine laufende Applikation am Starterkit nach „Spannung ein".

Alle notwendigen Schritte sollen hier beschrieben werden.

Zuerst ist eine Startup-Datei zu generieren.

Die vom Compilerhersteller mitgelieferte Startup-Datei ist natürlich für die Hardware, auf welcher unsere Applikation laufen soll, anzupassen. Wir generieren diese Datei mit DAvE.



I. DAvE - Digitaler Applikationsingenieur

DAvE ist ein Arbeitskollege. Er ist eine CD-ROM. Er ist ein kostenloses aber wertvolles Support-Tool.

DAvE bietet eine einzigartige, intuitive Benutzeroberfläche, mit der per Mausklick der ausgewählte 8-,16- oder 32-Bit-Mikrocontroller einfach konfiguriert werden kann.

Durch Drücken der rechten Maustaste erhält man jederzeit Hilfe.

So landet man z. B. bei der Eingabe eines Hex-Wertes durch Drücken der rechten Maustaste im Eingabefeld einer Binäreingabe.

Weiters erreicht man durch Drücken der rechten Maustaste in einem umrandeten Feld zusätzliche Hilfestellungen. Handelt es sich z. B. um die Konfiguration eines Bits oder eines Bitfeldes, so erhält man eine Darstellung des zugehörigen Registers, oder man springt auf die richtige Seite im Manual.

DAvE generiert automatisch den richtigen C-Code und ein komplettes Dateisystem, welches mit dem File Viewer betrachtet werden kann.

Zwischen „`// USER CODE BEGIN`" und „`// USER CODE END`" besteht die Möglichkeit, eigenen, applikationsspezifischen Source-Code einzufügen. Dieser bleibt bei einer Änderung der Konfiguration natürlich erhalten!

Hinweis (Vorgehensweise bei der Softwareentwicklung)

Das von DAvE generierte Dateisystem (`START.ASM`, `*.c` und `*.h` Dateien) ist in der Compileroberfläche (Keil µVision oder Tasking EDE) als Projekt anzulegen.

In der Compileroberfläche können weitere Dateien dem Projekt hinzugefügt werden.

Applikationsspezifischer Code darf in der Compileroberfläche mit dem Texteditor in den von DAvE generierten Dateien nur zwischen „// USER CODE BEGIN" und „// USER CODE END" eingefügt werden. Ist die Konfiguration des Mikrocontrollers zu ändern, so sollte man das Projekt in der Compileroberfläche schließen, das Projekt mit DAvE öffnen, die Änderung der Konfiguration des Mikrocontrollers durchführen, das Dateisystem von DAvE neu generieren lassen, das Projekt schließen und in der Compileroberfläche wieder öffnen und dort applikationsspezifisches Programm weiter erstellen, u.s.w..

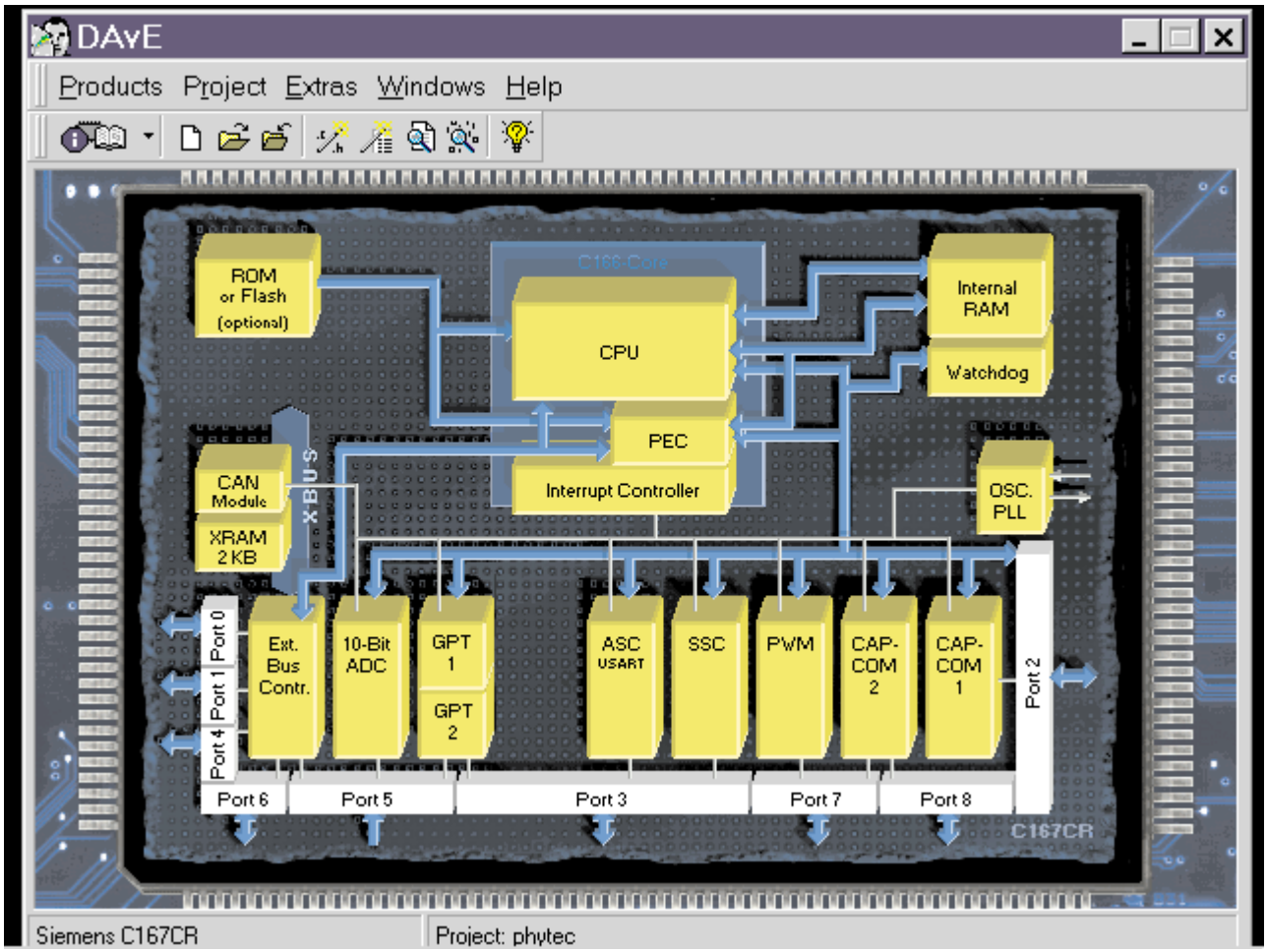
Es ist denkbar, dass Compileroberfläche und DAvE irgendwann ein gemeinsames Tool sein werden.

Hinweis (zum erfolgreichen Arbeiten mit DAvE)

Um erfolgreich mit DAvE zu arbeiten, sollte man die Architektur des Bausteins kennen (User`s Manual), ANSI C Programmierung beherrschen (speziell extern Deklarationen) und mit den mikrocontrollerspezifischen Erweiterungen des Sprachumfanges von ANSI C (zusätzliche Datentypen, Steuerworte und intrinsic/built in-Funktionen) zwecks Zugriff auf die Architektur des Bausteins vertraut sein.

Sonstiges über DAvE

- DAvE ist auch die Dokumentation der Konfiguration des Mikrocontrollers bei Programmstart.
- DAvE hilft bei der Auswahl eines Mikrocontrollers.
- Nach dem Selektieren der Anforderungen liefert DAvE eine Liste aller Mikrocontroller, welche die Anforderungen erfüllen (das günstigste Produkt steht dabei an erster Stelle).
- DAvE sollte regelmäßig über das Internet aktuell gehalten werden. Updates stehen unter *http://www.infineon.com/DAvE.html* zur Verfügung.



II. Starterkits

Starterkits bieten die Möglichkeit, auf günstige Art und Weise einen bestimmten Mikrocontroller und/oder eine Toolkette näher kennenzulernen.

Sie bestehen aus Hard- und Software.

Die Hardware ist ein einschaltfertiges Board (Spannungsstabilisierung, Mikrocontroller - alle Pins zugänglich, Programmspeicher - meist Flash-EEPROM, Datenspeicher - SRAM und Pegeltreiber für RS232). Mit Hilfe mitgelieferter Software kann einfach per Menü im Boot-Mode der Onboard - oder Onchip Programmspeicher programmiert werden. Dazu muss natürlich eine Intel-Hex-Datei vorhanden sein. Diese kann mit der mitgelieferten „Demo"-Software erstellt werden.

Das Wort „Demo" deshalb, da die Compilerpakete eingeschränkt sind (z.B. generierte Codegröße).

Ist man bereits glücklicher Besitzer einer Toolkette (z.B. Keil oder Tasking-Compiler), so bieten die Starterkits die Möglichkeit, rasch ein neues Derivat der Mikrocontrollerfamilie kennenzulernen ohne zuerst eigene Hardware entwickeln zu müssen.

III. Generierung der Startup-Datei mit Hilfe von DAvE für die Keil und Tasking Toolkette

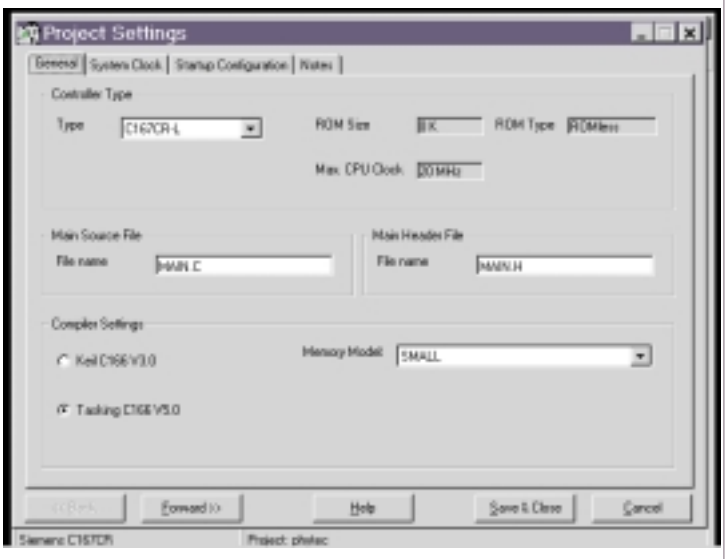
Nach Reset oder „Spannung ein" beginnt der Mikrocontroller mit dem Abarbeiten der internen Resetsequenz. Diese initialisiert alle Special Function Register auf bestimmte Werte. Danach beginnt der Mikroprozessor im Mikrocontroller mit der Abarbeitung des Programms ab Adresse 0.

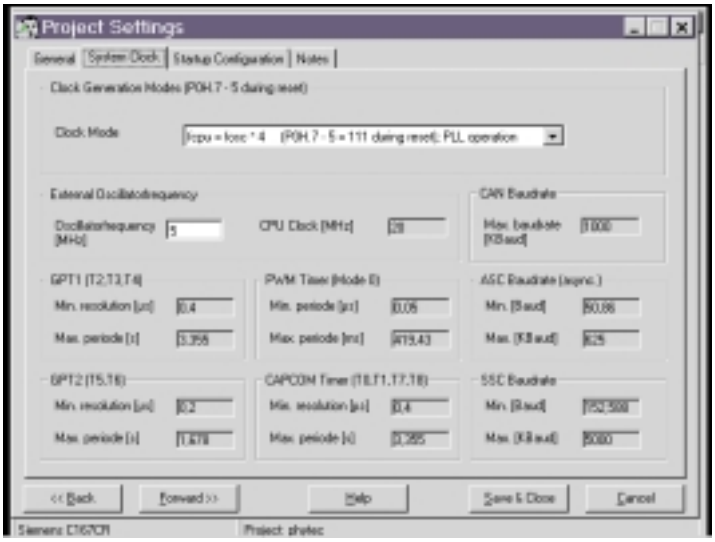
Da bei Adresse 0 die Interrupteinsprungtabelle (Vektortabelle) beginnt befindet sich hier ein Sprungbefehl zum Startup-Code. Der Startup-Code initialisiert den Mikrocontroller (Bus-Timing, Stack-Größe, ...) und schafft die Voraussetzungen für Hochsprachen (Variableninitialisierung, User-Stack, Floating-Point-Stack, ...). Der letzte Befehl des Startup-Code ist ein Sprung zum Hochsprachen-Hauptprogramm `void main (void)`.

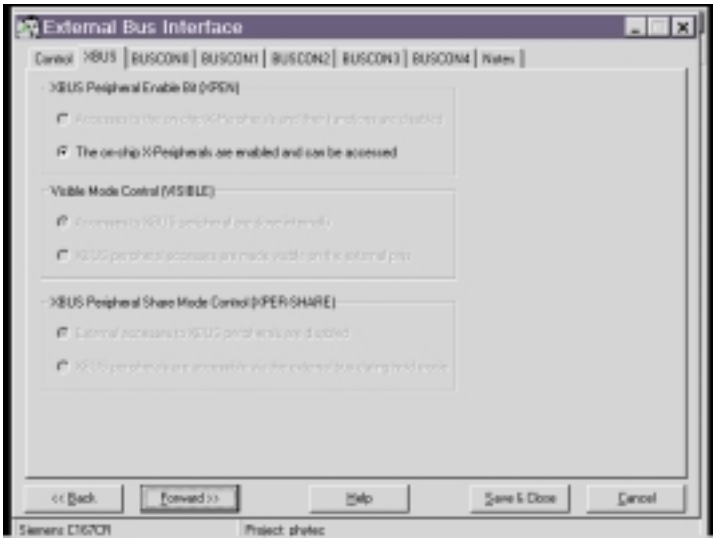
DAvE generiert eine Startup-Datei mit dem Namen `START.ASM`.

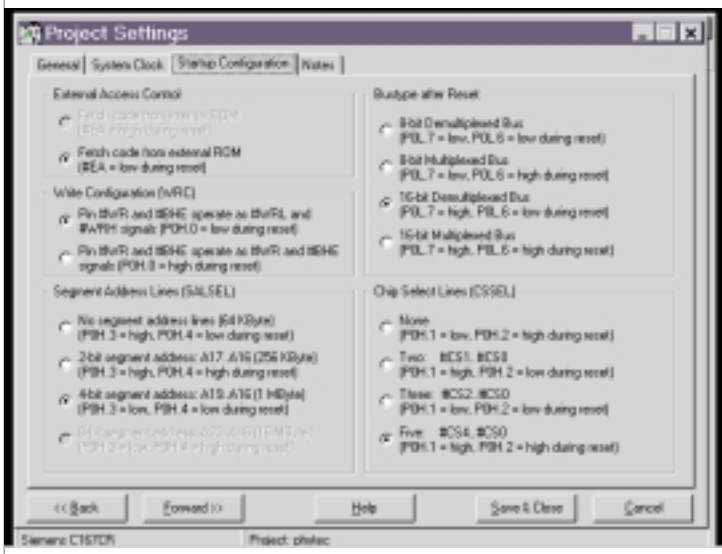
Zuerst wird mit DAvE ein Projekt (phytec) erstellt (***Project, New***). (***Screenshot oben***)

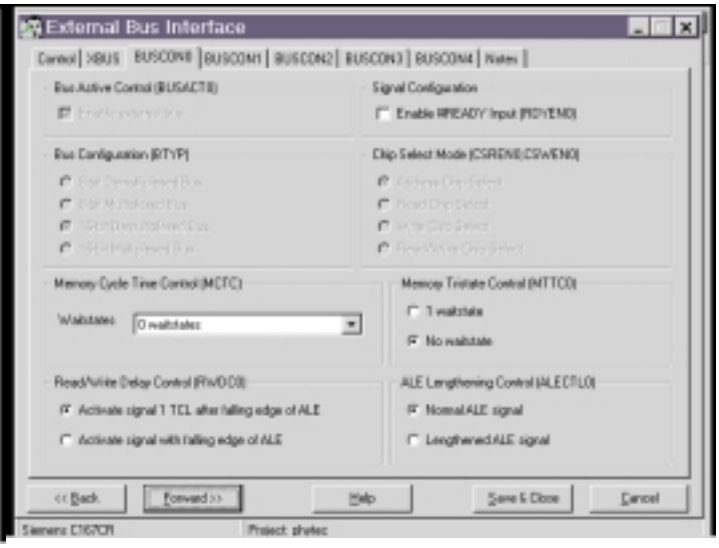
Bei den Project Settings stellen wir die verwendete Toolkette (Keil oder Tasking) ein (***Screenshot unten***). Beide Compiler sind zwar

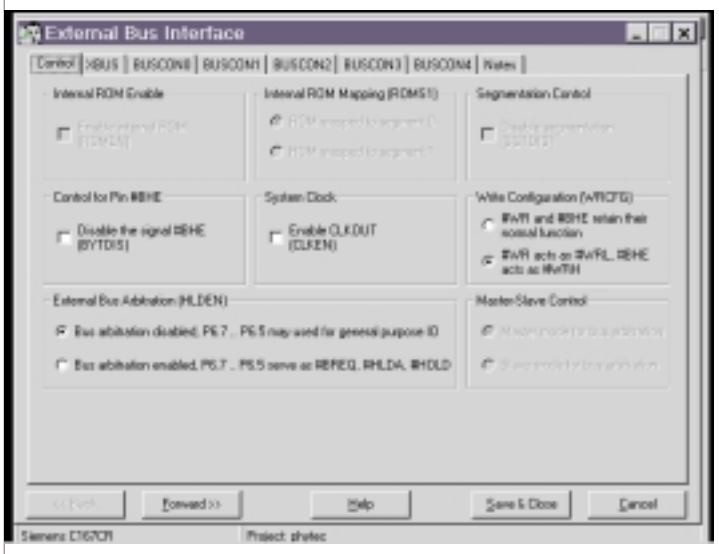


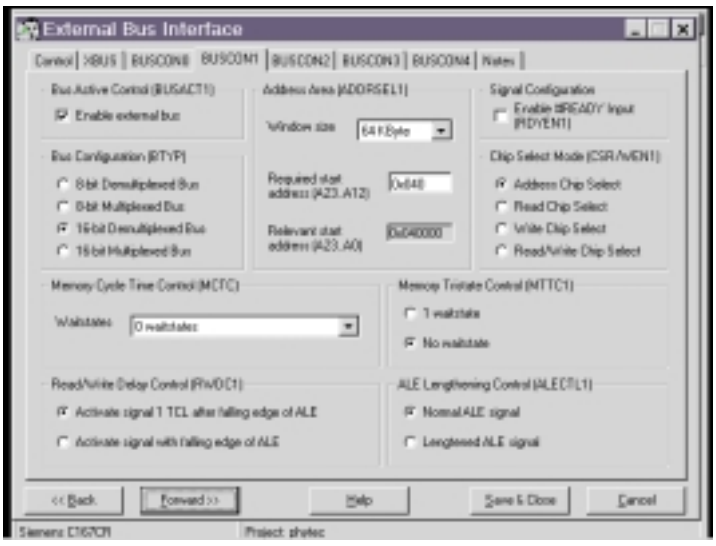












ANSI-C-Compiler unterscheiden sich aber in der Syntax der mikrocontrollerspezifischen Erweiterungen.

Durch das Einstellen der Taktfrequenz erhalten wir eine Übersicht über die zeitlichen Eigenschaften der On-Chip Peripherals.

In Übereinstimmung mit dem Phytec KitCON-167 Hardware-Manual wird die Startup(Einschalt)-Konfiguration und der External Buscontroller für BUSCON0 - CS0 - 256 KByte FLASH Bank1 U8/U9 - 00:0000h bis 03:FFFFh und BUSCON1 - ADDRSEL1 - CS1 - 64 KByte RAM Bank1 U10/U11 - 04:0000h bis 04:FFFFh konfiguriert (Chipselectkonfiguration).

Bei 55ns Speichern gilt: 0 Waitstate, RW-Delay, no Tri-state, short ALE, 16-Bit-Demultiplexed und Umschaltung von A0/BHE# zu Write LOW (WRL#) und Write HIGH (WRH#) - siehe auch Karl-Heinz Mattheis Buch Seite 94 und Kapitel 4 erhältlich als CD-ROM MC-Tools 17 – Arbeiten mit C166-Controllern unter *http://www.otmar-feger.de/*.

Mit *< Project - Generate Code >* initiieren wir die Source-Code-Generierung, die mit *< Project - File Viewer >* betrachtet werden kann.

Hinweis

DAvE generiert für die Keil und für die Tasking Toolkette eine Startupdatei mit dem Namen `START.ASM`.

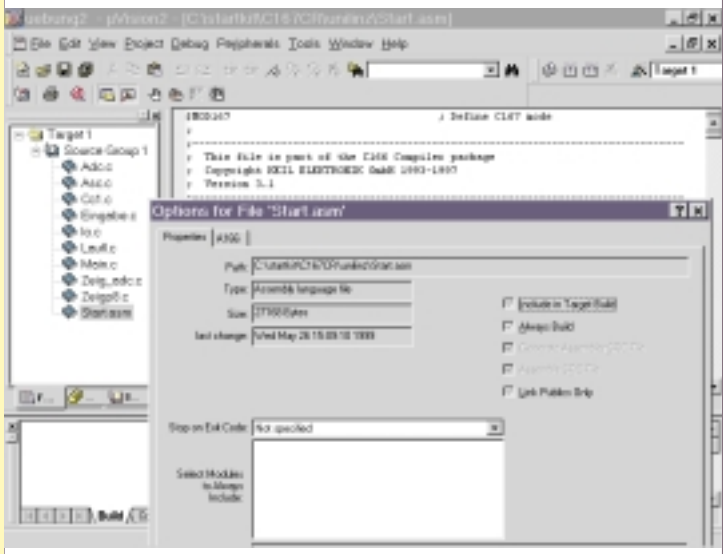
a Hinweis für die Keil-Toolkette

Die Initialisierung des externen Buscontrollers (BUSCON0 bis BUSCON4) wird von DAvE in der Startupdatei `START.ASM` gemacht und mit `*** INSERTED ***` gekennzeichnet.

Die von DAvE generierte Startupdatei `START.ASM` für unser C167CR-Starterkit wird danach folgendermaßen in die Keil Entwicklungsumgebung µVision eingebunden:

µVision1 - Keil Entwicklungsoberfläche, Einbindung der Startupdatei

Wichtig dabei ist, dass „***Include in Link/Lib***“ angekreuzt ist!



µVision2 - Keil Entwicklungsoberfläche, Einbindung der Startupdatei

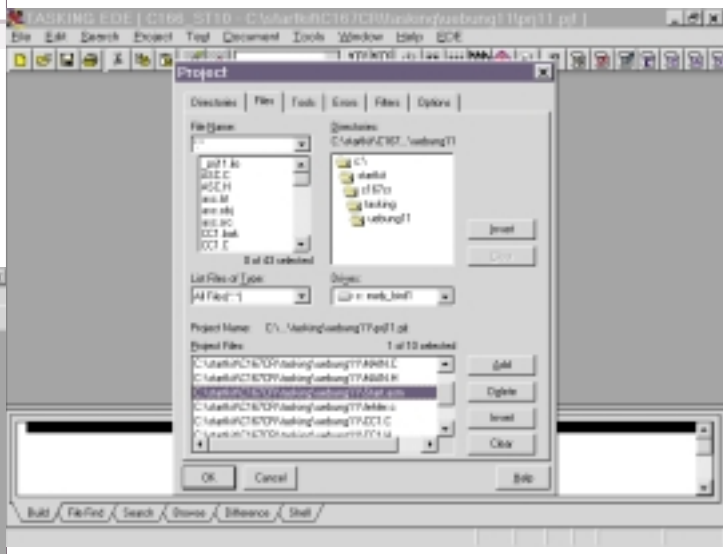
Weitere Hinweise zur Konfiguration der Compileroberfläche findet man am Unterrichtsserver der HTL-Klagenfurt unter *http://www.htblmo-klu.ac.at/lernen/*.

b Hinweis für die Tasking-Toolkette:

Die Initialisierung von BUSCON0 wird von DAvE in der Startup-Datei `START.ASM` gemacht und mit `*** INSERTED ***` gekennzeichnet.

Die Initialisierung von BUSCON1 bis BUSCON4 wir von DAvE in der Datei `main.c` innerhalb der `Project_Init()`-Funktion gemacht.

Die von DAvE generierten Dateien `START.ASM` und `MAIN.C` werden danach folgendermaßen in die Tasking-Entwicklungsumgebung EDE (Embedded Development Environment) eingebunden:



EDE - Tasking Entwicklungsoberfläche, Einbindung der Startupdatei

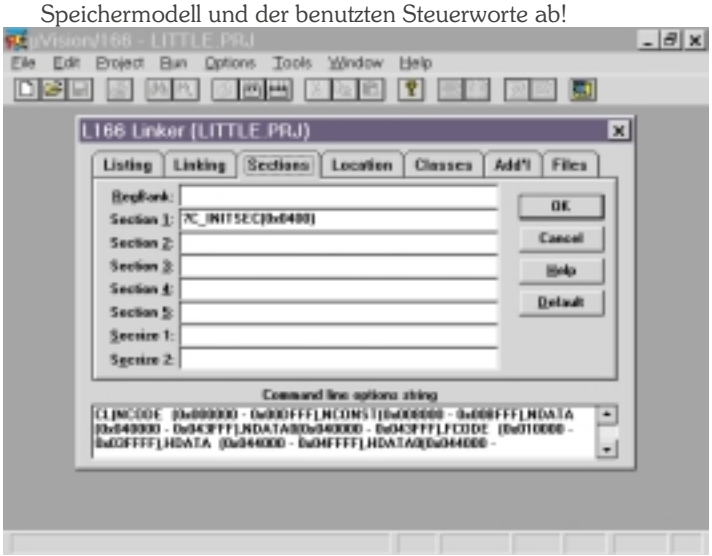
IV. Locater Steuerdatei für das C167CR Starterkit

Die Locater Anweisungen sind notwendig, damit die vom Compiler generierten Segmente (Code, Daten) vom Linker/Locater auf die richtigen physikalischen Speichermedien (ROM, RAM) gemäß der Hardware (C167CR Starterkit) aufgeteilt werden.

a bei der Keil-Toolkette, µVision 1

Die Angaben sind hier natürlich nur als Beispiel gedacht!

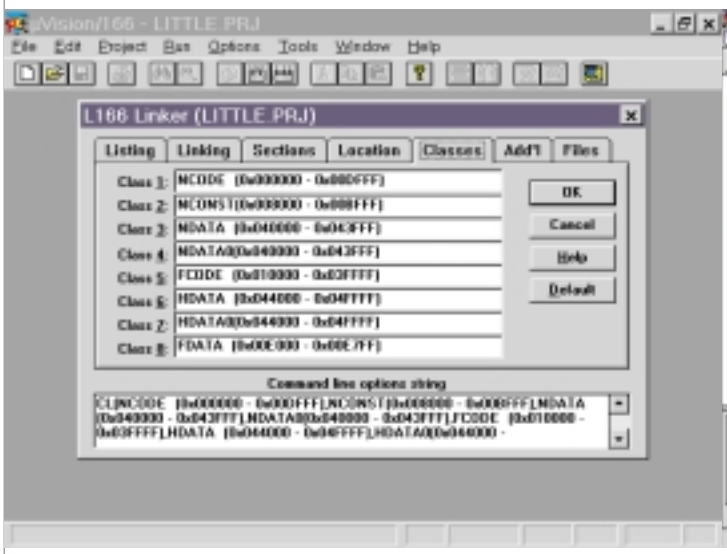
Welche Classes und Sections der Compiler tatsächlich generiert hängt natürlich vom Anwenderprogramm, vom verwendeten Speichermodell und der benutzten Steuerworte ab!



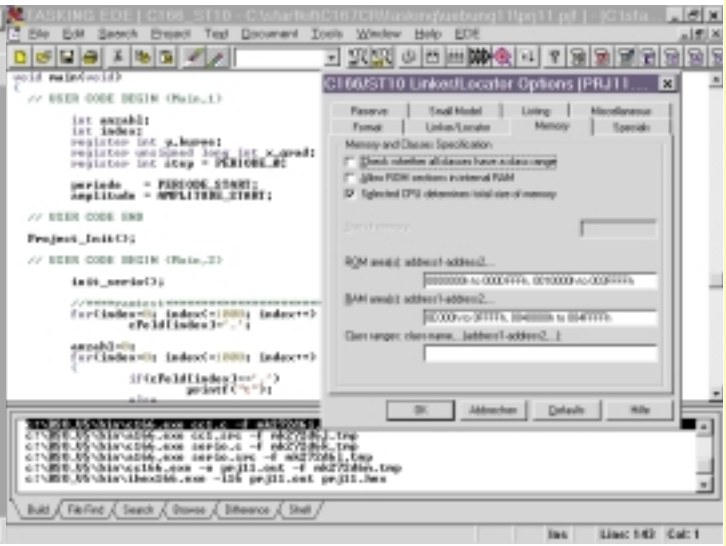
µVision1- Keil Entwicklungsoberfläche, Linker/Locater

b bei der Keil-Toolkette, µVision 2

Bei der µVision2 kann auf einfache Art und Weise angegeben werden, wo sich im Zielsystem

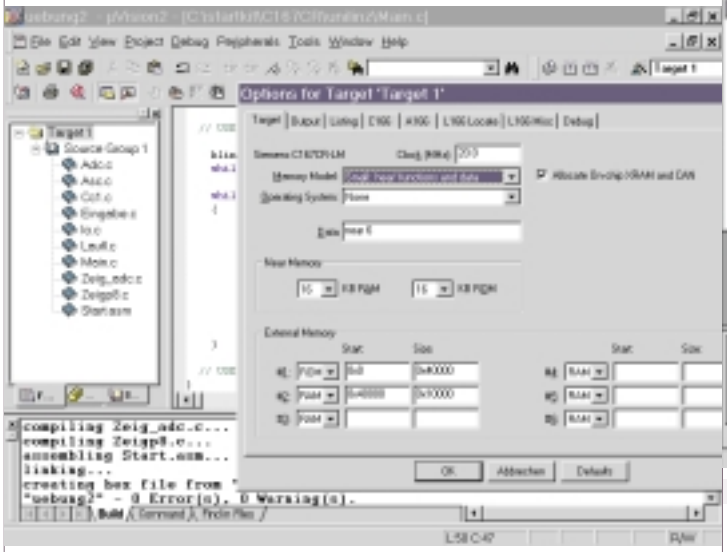


µVision1 Keil Entwicklungsoberfläche, Linker/Locater

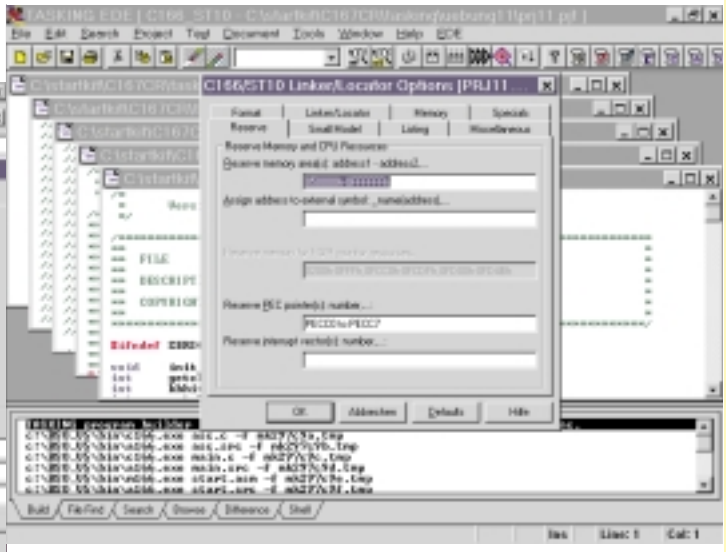


EDE - Tasking Entwicklungsoberfläche, Linker/Locater

Speichertyp „ROM“ und Speichertyp „RAM“ befindet! Diese Angaben sind zur Generierung der hex-Datei (*.h86) notwendig.



µVision2 Keil Entwicklungsoberfläche, Linker/Locater



EDE - Tasking Entwicklungsoberfläche, Linker/Locater

Ach ja, natürlich läuft das hier besprochene Programm „nur“ am Zielsystem „nach Spannung“ ein und am Simulator.

c bei der Tasking-Toolkette

Bei der EDE kann auf einfache Art und Weise angegeben werden, wo sich im Zielsystem Speichertyp „ROM“ und Speichertyp „RAM“ befindet! Diese Angaben sind zur Generierung der Hex-Datei notwendig.

V. Schlussbemerkungen

Ich hoffe, dass diese Darstellungen hilfreich sind.

Feedback zum Artikel bitte an *wilhelm.brezovits@siemens.at* .

Warum nicht mit dem Debugger?

Der Debugger muss die Möglichkeit haben, das Programm zu ändern. Definiert man zum Beispiel einen Breakpoint, so wird der Code an dieser Stelle durch einen Sprung in den Monitor (der mit dem Debugger kommuniziert) ersetzt, damit der Debugger weiss, dass diese Stelle erreicht wurde.

Das bedeutet allerdings: Das Programm steht im „RAM“ und nicht im Zielspeichertyp „ROM“.

Das RAM beginnt im Debuggerbetrieb auf der Adresse 00:0000 H (statt 04:0000 H) – also andere START.ASM und andere Linker/Locater-Steuerung!