ERFOLGREICH STARTEN

MIT DEM INFINEON C167-STARTERKIT UND DEM SOFTWARE-ENTWICKLUNGS SYSTEM VON KEIL

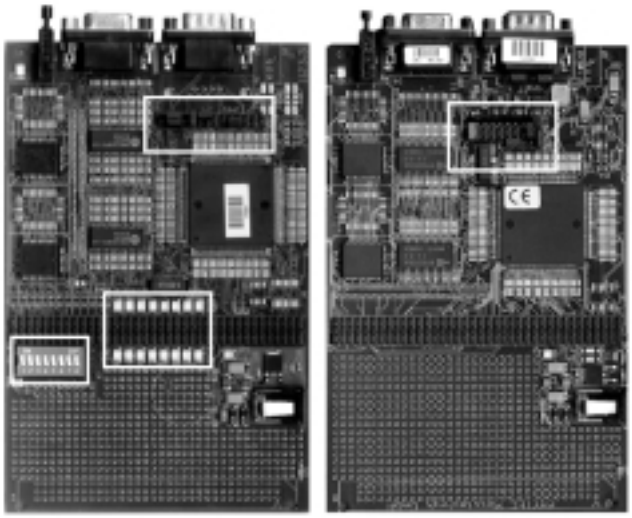
Walter Waldner

1. Einleitung

Mit dem **Infineon C167-Starterkit** erhält man alles, um die Welt der 16-Bit-Mikrocontroller erforschen zu können. Das Kit enthält:

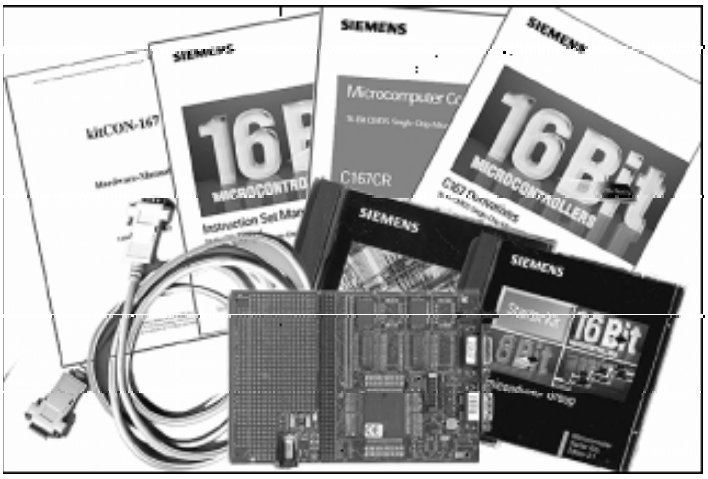
- das kitCON-167 Evaluation-Board von Phytec mit dem Infineon 16-Bit-Mikrocontroller C167CR-LM, 64KB RAM und 256 KB Flash-Memory
- Eine CD-ROM mit Software und Dokumentationen
- Manuals zum C167-Mikrocontroller
- serielles Kabel zur Verbindung des kitCON-Boards mit dem PC

Das Starterkit wurde Mitte 1998 neu aufgelegt. Diese Ausgabe enthält gegenüber der 1997er-Ausgabe ein geringfügig geändertes kitCON-167-Board.



Das neuere kitCON-167-Board (nachfolgend als kitCON-167-1998 bezeichnet) ist im Bild links abgebildet. Es enthält gegenüber dem kitCON-167-1997 zusätzlich 16 Leuchtdioden, die mit Port 2 des Controllers verbunden sind. Auch die Funktion der Jumper wurde etwas geändert. Neu hinzugekommen ist der Schalter SW3, über den unter anderem der Bootstrap-Modus aktiviert wird. Die nachfolgenden Ausführungen sind für beide Board-Varianten gültig. Auf die wenigen Unterschiede zwischen den beiden Platinen wird an gegebener Stelle hingewiesen werden.

Dieser Artikel beschreibt, wie Sie ihre ersten einfachen Programme auf der Starterkit-Hardware zum Laufen bringen - und das, ohne vorher die Assemblersprache des Prozessors lernen zu müssen. Heute werden Mikrocontroller fast ausschließlich in Hochsprache programmiert. Die hardwarenahe Programmiersprache C hat sich in diesem Bereich als sehr geeignet erwiesen und auch für die C166-Familie gibt es eine Reihe von leistungsfähigen Software-Entwicklungsumgebungen für C. Auf der Starterkit-CD-ROM sind einige Demoversionen dieser Werkzeuge enthalten, die unterschiedliche Einschränkungen gegenüber den (meist recht teuren) Vollversionen aufweisen. Meine Wahl fiel auf die Toolkette der Firma KEIL - ein professionelles Entwicklungssystem mit Assembler, Compiler, Linker/Locator, Intel-Hex-Konverter und Debugger/Simulator. Die KEIL-Demoversion erlaubt es, Programme bis zu 4 KB Codegröße zu übersetzen, zu linken und in die Ziel-Hardware zu laden, was für viele Experimente zum Kennenlernen aller Komponenten des C167-Mikrocontrollers völlig ausreichend ist.



Aller Anfang ist schwer - das gilt auch für die erste Inbetriebnahme des Starterkits. Zunächst geht es darum, auf der umfangreichen CD-ROM die richtigen Verzeichnisse und Dateien zu finden und die Keil-Software auf dem PC zu installieren. Danach sind Compiler, Linker und Debugger zu konfigurieren. Schließlich gibt es noch verschiedene Möglichkeiten, das Programm auf dem kitCon167-Board zum Laufen zu bringen.

Dieses Dokument soll Ihnen helfen, sich in dieser Vielfalt zurecht zu finden und nach Ihren ersten Experimenten werden Sie wahrscheinlich meine Meinung teilen, dass das C167-Starterkit ein ganz fantastisches Paket ist, mit dem man hervorragend arbeiten kann, das leistungsfähig und doch einfach zu programmieren ist, wenn man weiß, wie.

2. Erforderliche Vorkenntnisse

Zunächst sollten Sie Teile der Dokumentationen lesen, die Sie mit dem Starter-Kit erhalten haben:

- das kitCON-167-Hardware-Manual
- zumindest die Kapitel “Introduction”, “Architectural Overview”, “Memory Organization”, “Central Processing Unit”, “Parallel Ports” des C167-User-Manuals

3. Die Software-Entwicklungsumgebung von KEIL (“KEIL Tool-Kette”) PK166

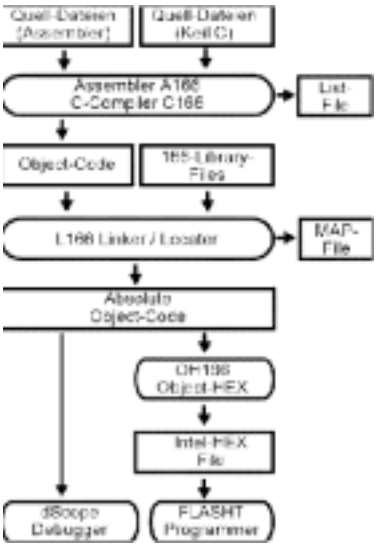
Die KEIL-Tool-Kette (Professional Developers Kit PK166 Version 3.xx) stellt ein vollständiges Entwicklungssystem für alle Mikrocontroller der C166-Familie dar. Sie enthält unter anderem einen C166-Assembler, einen für den C167-Mikrocontroller zugeschnittenen und optimierten C-Compiler, einen Linker/Locater und einen Simulator/Debugger.

Der Anwender arbeitet dabei unter Windows 3.x / 9x /NT mit einer integrierten Entwicklungsumgebung (μVision 1.xx), die das Projektmanagement übernimmt und die erforderlichen Schritte vom Source-Code bis zur lauffähigen Version weitgehend automatisiert, nachdem die Oberfläche entsprechend konfiguriert wurde.

Die Source-Files können in Assembler- oder C-Code erstellt werden. Die Übersetzung in den C-167-Objectcode erfolgt durch den Assembler A166 bzw. durch den ANSI-C-Compiler C166. Der Übersetzungsvorgang wird in einer List-Datei (Extension .LST)

MTM "Erfolgreicher Start"

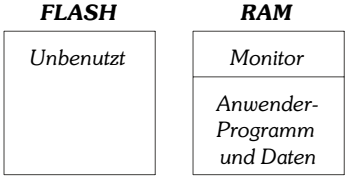
dokumentiert. Die so erzeugten Object-Files (relocateable = verschiebbar) können nicht direkt ausgeführt werden. Der L166-Linker/Locater verknüpft die Object-Dateien und Bibliotheksfunktionen, löst Symbole auf und erzeugt absolute Object-Dateien (mit absoluten Adressen innerhalb des 16 MB großen Adressraum des C167-Mikrocontrollers). Das Protokoll des Link/Locator-Laufes kann in der Map-Datei (Extension .M66) betrachtet werden. Die Object-Datei kann anschließend mit dem dScope-Debugger (und einem Monitorprogramm, das auf dem C167 läuft) in das SRAM geladen und ausgeführt werden. Alternativ läßt sich ein Programm aber auch in das Flash-Memory laden. Dazu verwendet man das Tool FLASHT, das als Eingabeformat allerdings eine Intel-Hex-Datei (ASCII-Format) erfordert. Diese Umwandlung erledigt das Programm OH166.



4. Wohin mit dem Programm ?

Eine ganz wichtige Frage, die den Programmentwicklungsprozess wesentlich beeinflusst, ist die Entscheidung, wohin das vom Anwender entwickelte Programm gespeichert werden soll. Das kitCON bietet ein SRAM und ein Flash-Memory an. Dementsprechend gibt es im wesentlichen zwei Möglichkeiten. Das Laden des Anwendungsprogramms auf die Ziel-Hardware erfolgt dabei stets über die serielle Leitung. Das kitCON-167 Board wird mit dem seriellen Kabel (liegt dem Starterkit bei) mit der COM-Schnittstelle des PCs verbunden. Am Evaluation-Board wird das Kabel an den Stecker neben dem Reset-Schalter angeschlossen. Ein Tipp: wird das Kabel zuerst mit der seriellen Schnittstelle des PCs verbunden, passt das andere Ende des Kabels ohnehin nur in die eine der beiden Anschlussbuchsen des kitCON.

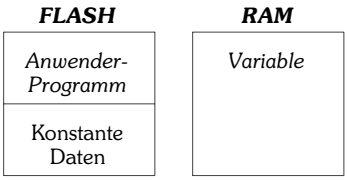
4.1 Möglichkeit 1 - RAM



Während der Entwicklungsphase einer Anwendung, insbesondere aber auch in der Ausbildung (Schulbetrieb), ist es sinnvoll, diese Variante zu wählen, da Programmänderungen sofort ins SRAM geschrieben werden können. Eine wesentliche Rolle spielt dabei der Monitor. Der Monitor ist Bestandteil der Toolkette und wird mit dem Keil-Simulator-Debugger dScope und dem in den C167-Mikrocontroller integrierten Bootstrap-Loader (über die serielle Verbindung) in das RAM des Phytec-Boards geladen. Diese Software erlaubt es, anschließend unser Anwendungsprogramm in das RAM zu laden und zu starten. Darüber hinaus kommuniziert das Monitor-Programm mit der dScope-Oberfläche und ermöglicht so ein sehr komfortables Debuggen unserer Software (Einzelschritt-Abarbeitung, Betrachten von Speicherinhalten, Anzeige von Special Function Registern und vieles mehr).

Ist unser Programm fertig entwickelt, kann es in das Flash-Memory programmiert werden. Das ist Möglichkeit 2 mit folgender Aufteilung.

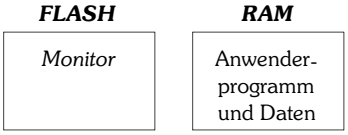
4.2 Möglichkeit 2 - FLASH Memory



Diese Variante hat natürlich den Vorteil, unser Programm nun permanent gespeichert ist und automatisch startet, sobald das kitCon-Board mit Spannung versorgt wird oder der RESET-Knopf gedrückt wird. Das Programmieren des Flash-EEPROMs wird nicht direkt durch das Keil-Entwicklungssystem unterstützt. Dazu muss das DOS-Programm `FLASHT.EXE` verwendet werden, das sich ebenfalls auf der Starterkit-CD-ROM befindet. Da der Flash-Speicher etwa 100.000 Schreibzyklen verträgt, kann dennoch bei Bedarf jederzeit eine Neuprogrammierung erfolgen.

Schließlich gibt es noch eine Variante zur Möglichkeit 1:

4.3 Möglichkeit 3 - FLASH / RAM



In dieser Variante wird das Monitor-Programm in das Flash-Memory geladen und steht somit ohne Neuladen nach jeder Spannungsunterbrechung zur Verfügung. Da der Monitor aber ein relativ kleines Programm ist (ca 5 kB), das schnell über die serielle Schnittstelle übertragen wird, bringt dies nur geringe Geschwindigkeitsvorteile. Will man allerdings das RAM auf Grund großer Datenmengen bis zum letzten Byte nutzen, ist das Laden des Monitors in den Flash-Speicher eine sinnvolle Methode.

Zu den verschiedenen Methoden des Arbeitens und Konfigurierens des Monitors habe ich ein eigenes Dokument erstellt (***siehe [1]***). Ich werde hier daher auf die verschiedenen Varianten nicht mehr näher eingehen, sondern beschreibe den direktesten und einfachsten Weg zum ersten lauffähigen C167-Programm.

5. Installation der KEIL-Toolkette

Zur Installation legen Sie die Starter-Kit-CD-ROM ein und starten die Datei `SETUP.EXE` im Verzeichnis `X:\CDROM\3RDTOOLS\KEIL\C166`

Während der Installation werden Sie nach einem Ziel-Verzeichnis gefragt. Für die folgenden Ausführungen wird angenommen, dass Sie die Standardvorgabe eingeben: `C:\C166EVAL`

Das gesamte System benötigt etwa 6.5 MB auf Ihrer Festplatte. Bei diesem Entwicklungssystem handelt es sich um eine Demoversion, die den vollen Leistungsumfang anbietet - nur die Codegröße der Programme ist mit 4 kB beschränkt, was für das

Kennenlernen des C167-Mikrocontrollers eine wenig relevante Einschränkung ist.

Nach der Installation des Demo-Pakets müssen nun noch die richtigen Dateien für den Monitor in das Verzeichnis BIN kopiert werden. Der Monitor ist jenes Programm, das in den Speicher des Evaluation-Boards geladen wird und das Laden von Anwendungsprogrammen, sowie das Debuggen dieser Programme ermöglicht, indem es über die serielle Schnittstelle mit dem PC kommuniziert (Variante 1, wie in ***4.1.*** beschrieben).

Auf der CD-ROM findet man an vielen Stellen Monitor-Programme, aber nur eines funktioniert mit dem Phytec kitCON-Board. Und diese Dateien befinden sich etwas versteckt (und ohne entsprechend Hinweise) im Verzeichnis

```
CDROM\STARTKIT\SK_167\MONITOR\KEIL
```

Kopieren Sie aus diesem Verzeichnis die beiden Dateien `BOOT` und `MONITOR` in das BIN-Verzeichnis des KEIL-Software-Pakets. Falls Sie bei der Installation die Default-Vorgabe übernommen haben, ist dies das Verzeichnis

```
C:\C166EVAL\BIN
```

Verbinden Sie nun eine serielle Schnittstelle des PCs (z.B. COM1) mit der Buchse P1 des kitCON-167-Boards. Ein Tipp: stecken Sie das Kabel zuerst an Ihrem PC ein, dann passt es am kitCON-Board ohnehin nur in die Buchse P1.

Die serielle Verbindung zwischen PC und dem C167-Board erfüllt mehrere Aufgaben:

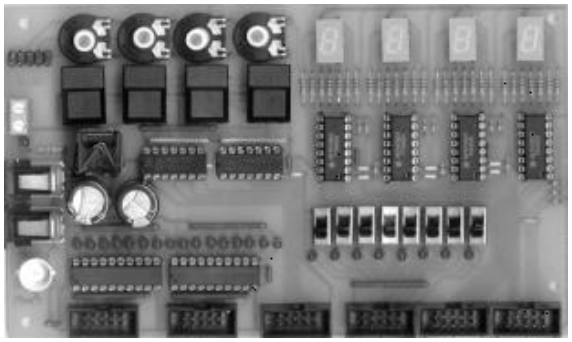
- Programme können damit wahlweise in das RAM oder das Flash-Memory des Boards geladen werden
- Bei der Arbeit mit dem Debugger werden Kommandos und Daten zwischen PC und Board transferiert
- Die C-Library des Keil-Systems enthält Routinen, mit denen User-Programme Daten über die serielle Schnittstelle senden und empfangen können

Wir wollen nun anhand eines einfachen Beispiels die Schritte vom Source-Code bis zum Ausführen des Anwendungsprogramms auf der Zielhardware zeigen. Wie unter ***4.1*** und ***4.2*** beschrieben, gibt es zwei grundsätzlich verschiedene Möglichkeiten, die beide hier dargestellt werden.

6. LED-Lauflicht - das erste C167-Projekt

6.1 Die Schaltung

Ein Mikrocontroller-Board ohne Peripherie (Sensorik, Ein- und Ausgabekomponenten) macht wenig Sinn. Insbesondere zum Experimentieren und Kennenlernen des Aufbaus und der Funktionsweise des Mikrocontrollers und seiner Bestandteile sind solche Zusatzkomponenten wünschenswert.



Ein **Experimentierboard** (EXBO), das genau auf die Phytec-kitCON-167-Platine zugeschnitten ist, ist inzwischen entwickelt worden und kann nachgebaut werden. Informationen über dieses Board und die zugehörigen Dateien zum Laden auf Ihren PC finden Sie auf meinen Webseiten im Internet (siehe ***[5]***). Das vorige Bild zeigt das Experimentierboard EXBO.

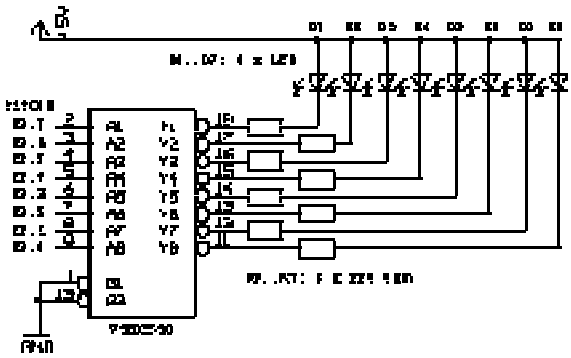
Wir werden ein Lauflicht-Programm schreiben und anhand dieses Beispiels die wesentlichen Schritte der Software-Entwicklung beschreiben.

Für dieses Projekt nehmen wir an, dass 8 Leuchtdioden an das Port 2 (Pins 0 bis 7) des kitCON-C167 angeschlossen sind.

Besitzer des kitCON-167-1998 haben solche, mit dem Port 2 verbundene Leuchtdioden, bereits auf dem Phytec-Board (siehe kitCON Hardware-Manual Version 2.0 7/1998). Die Kathoden der (low-current) LEDs sind am kitCON-167-1998 direkt mit den Pins des Ports 2 verbunden. Die Anoden sind über Vorwiderstände an 5 Volt geführt. Beachten Sie, dass die LEDs daher genau dann leuchten, wenn die entsprechenden Pins des Ports 2 LOW-Pegel aufweisen.

Wenn Sie mit EXBO arbeiten, ist lediglich eine Verbindung mit Flachbandkabel zwischen der LED-Steckerleiste des EXBO und dem kitCON-167erforderlich. Die LEDs am EXBO leuchten genau dann, wenn der Eingangspegel HIGH ist.

Sie können aber auch die einfache, nachfolgend abgebildete Schaltung aufbauen und mit den Pins 0 bis 7 des Ports 2 des kitCON-167-Boards verbinden.



Diese Schaltung verwendet den CMOS-Baustein 74HC540 (8fach invertierender Buffer/Line Driver mit Tristate-Ausgängen) zur Ansteuerung der Leuchtdioden. Die Pins des C167 erlauben nur einen Sink-/Source-Strom von etwa 1 mA, was für Standard-LEDs viel zu wenig ist. Die Ausgänge des 74HC540 hingegen können mit maximal 35 mA belastet werden. Der Strom durch die Leuchtdioden wird durch Widerstände von jeweils 220 Ohm auf ca. 15 mA beschränkt. Bei dieser Schaltung leuchten die LEDs genau dann, wenn der Eingangspegel HIGH ist.

6.2 Die µVision-Oberfläche (Version 1.xx)

Das Lauflicht-Programm wird mit der Sprache C unter der µVision-Oberfläche von KEIL entwickelt. Starten Sie nun das Programm µVision. Das Setup-Programm sollte eine Programmgruppe eingerichtet haben (unter Windows 95 im Start-Menü). Ansonsten finden Sie µVision unter dem Dateinamen `UVW166E.EXE` im Verzeichnis `C:\C166EVAL\BIN`.

Wie andere Programmierumgebung auch, verwaltet µVision Projekte. Zu einem **Projekt** gehören sämtliche Source-Dateien, Bibliotheken und Compiler/Linker-Einstellungen. Aus diesen Projekt-Elementen erstellt µVision ein lauffähiges C167-Programm, wobei die in 3 beschriebenen Stufen weitgehend automatisiert durchlaufen werden. Auch den dScope-Simulator/Debugger können wir über die µVision-Oberfläche aufrufen.

Unser Lauflicht-Projekt wird nur aus einer einzigen Quell-Datei (einem C-Programm) bestehen.

Wählen Sie im Menü: ***FILE - NEW***. Ein Editor-Fenster öffnet sich. Speichern Sie die noch leere Datei gleich einmal mit der Extension .C ab, um den Syntax-Check und die Farbkennung der C-Sprachelemente zu aktivieren. Für die weiteren Ausführungen nehmen wir an, dass Sie der Datei den Namen `LLICHT.C` gegeben haben und unter `C:\166EVAL` ein Verzeichnis `PROJECTS` angelegt haben, in das `LLICHT.C` gespeichert wird.

6.3 Der C-Source-Code

Hier ist nun der Source-Code für unser Lauflicht-Programm:

```
// Lauflicht ueber Port 2 // Walter Waldner, 1998/07
#include <reg167.h>

void warten(unsigned int w);
const unsigned int dauer = 0x2000;

void main(void)
{
  unsigned int x;

  // Pins 2.0 bis 2.7 als Ausgänge
  DP2 = 0x00FF;
  ODP2 = 0x0000;
  // Timer 3 konfigurieren
  T3CON = 0x0007;
  while (1)
  {
    for (x=1; x<=0x0080; x=x<<1)
    {
      P2 = x;
      warten(dauer);
    }
    for (x=0x0040; x>=0x0002; x=x>>1)
    {
      P2 = x;
      warten(dauer);
    }
  }
}

void warten(unsigned int w)
{
  T3 = 0;
  T3R = 1;
  while (T3 <= w);
  T3R = 0;
}
```

ACHTUNG: Wenn Sie für das Projekt die am kitCON-167-1998 vorhandenen LEDs verwenden, sind die beiden Anweisungen

```
P2 = x;
```

durch die Anweisung

```
P2 = x ^0x00FF;
```

zu ersetzen, da wir möchten, dass die LEDs leuchten, wenn die entsprechenden Bits in der Variablen `x` den Wert 1 haben (siehe Punkt ***6.1***).

Sehen wir uns nun den Source-Code und die Besonderheiten des Keil-Compilers an:

Alle SFRs (special function registers), aber auch Bitgruppen oder einzelne Bits der Register können über die Namen angesprochen werden, die im User-Manual des C167 definiert sind. Dazu ist lediglich die Header-Datei `reg167.h` mit der `#include`-Anweisung zu laden. Der Keil-C-Compiler entspricht dem ANSI-Sprachumfang. Jeder Programmierer mit C-Erfahrung wird sich schnell zurechtfinden.

Die 8 Leuchtdioden sind für unser Experiment mit den Pins 0 bis 7 (Low-Byte) des 16-Bit-Ports P2 verbunden. Diese Pins sind zunächst als Ausgänge zu definieren. Dazu wird in das Direction-Register DP2 der hexadezimale Wert 0x00FF geschrieben (ein 1-Wert auf der Bitposition b schaltet das Pin b des Ports als Ausgang). Das ODP2-Register wird auf 0 gesetzt. Damit werden die als Ausgang definierten Pins des Ports 2 in den Push-pull-Modus geschalten (1-Werte würden den Open-drain-Modus wählen). Die beiden for-Schleifen erzeugen für die Variable x der Reihe nach die Bit-Kombinationen 00000001, 00000010, 00000100, .., 10000000, 01000000 ... 00000010. Da ein 1-Wert einer leuchtenden Diode entspricht, erhalten wir so den gewünschten "Lauflicht"-Effekt. Durch die Anweisung `P2 = x` wird der Wert von x in das Port2-Register geschrieben und die entsprechenden Pegel erscheinen am Ausgang.

Nach jedem Schreibvorgang müssen wir eine Warteschleife einbauen, um den Laufeffekt überhaupt beobachten zu können. Wir könnten dies etwa durch eine for-Schleife der Art
`(for y=0; y<=30000; y++);`
erreichen. Viel attraktiver ist es allerdings, einen der zahlreichen Timer des C167 dafür zu verwenden. Für unser Beispiel setzen wir den Timer T3 ein. Durch Einschreiben eines bestimmten Wertes in das `T3CON`-Register (Timer 3 configuration register) können wir die Arbeitsweise von T3 festlegen. Lesen Sie dazu im User-Manual das ***Kapitel 9*** (General Purpose Timer Units). `T3CON = 0x0007` gibt an, dass das Zählregister T3 mit der durch 1024 geteilten internen Taktfrequenz der C167-CPU (20 MHz) angesteuert wird und aufwärts zählen soll (siehe ***Seite 9-5*** im C167 User Manual Version 2.0).

Das Unterprogramm `warten(unsigned int w)` setzt das Zählregister `T3` auf 0 und startet anschließend den Zählvorgang, indem das Run-Bit (`T3R`) auf 1 gesetzt wird. Die `while`-Schleife schafft eine Verzögerung, bis `T3` den Wert des Parameters w überschritten hat. Anschließend wird der Timer gestoppt (`T3R = 0`). `T3R` ist ein Beispiel für den Zugriff auf ein einzelnes Bit eines 16-Bit-Registers. Ein Blick in `reg167.h` zeigt, wie man solche symbolische Namen vereinbaren kann:

```
sbit T3R = T3CON^6;
```

Mit dem Datentyp `sbit` ist es möglich, einzelnen Bits eines `SFR` einen symbolischen Namen zu geben. User-Manual ***Seite 9-3*** zeigt die Bitzuordnungen für das `T3CON`-Register. `T3R` ist das Bit 6 dieses 16-Bit-Speichers.

Damit ist unser erstes Keil-Programm hinreichend beschrieben. Wir speichern das Programm mit ***FILE - SAVE*** ab (den Namen `LLICHT.C` haben wir ja bereits vergeben).

6.4 Definition eines Projekts

Bevor wir das Programm nun erfolgreich übersetzen und starten können, muss zunächst ein PROJEKT definiert werden. Wie in anderen Programmier-Umgebungen wird auch im Keil-System dabei eine `.PRJ`-Datei erstellt, die Informationen darüber enthält, welche Quell-Dateien (Assembler und/oder C) zum Projekt gehören und welche Einstellungen für den Compiler und Linker/Locater beim Erstellen des lauffähigen Programms verwendet werden sollen.

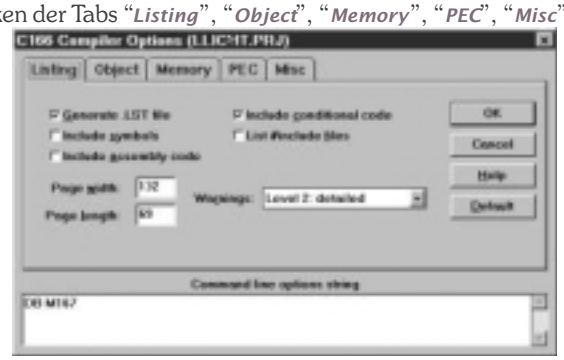
Wählen Sie im Menü den Punkt "***Project - New Project***". Geben Sie dem Projekt den Namen `LLICHT.PRJ` und speichern Sie diese (wie auch `LLICHT.C`) in das Verzeichnis `C:\C166EVAL\PROJECTS`.

In der nun erscheinenden klicken Sie auf "***Add***". In diese Liste tragen wir alle Dateien ein, die zu diesem Software-Projekt gehören. In unserem Fall ist dies nur das C-Source-Programm `LLICHT.C`. Achten Sie bitte darauf, dass das Feld "***Include in Link/Lib***" angekreuzt ist. Da unser einfaches Beispiel keine weiteren Quelldateien benötigt, drücken wir nun den Knopf "***Save***".

6.4.1 Compiler-Optionen

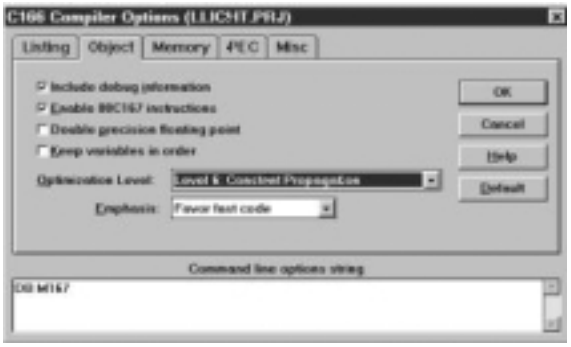
Nun müssen die Einstellungen für den C-Compiler vorgenommen werden. Wählen Sie im Menü "***Options - C166 Compiler***".

Es erscheint eine Dialogbox mit mehreren Seiten, die durch Anklicken der Tabs "***Listing***", "***Object***", "***Memory***", "***PEC***", "***Misc***" aufgerufen werden können.

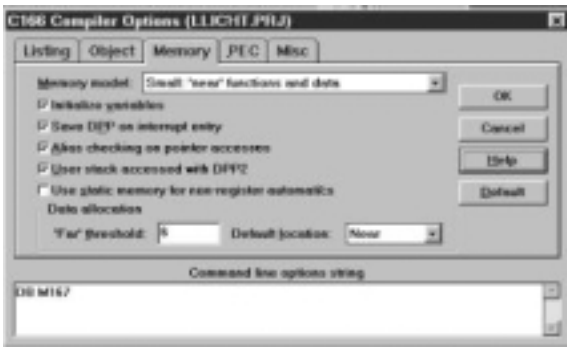


Unter "***Listing***" können Sie Parameter für die Listing-Datei angeben, die vom Compiler erzeugt wird. Für unser Projekt würde diese Datei `LLICHT.LST` heißen. Ein Blick in diese Datei offenbart, was der Compiler aus unserem Source-Programm macht. Als Einsteiger können Sie diese Datei und auch die Einstellungen getrost ignorieren.

Wichtiger ist schon die nächste Dialogbox "***Object***". Klicken Sie "***Include debug information***" und "***Enable 80C167 instructions***" an. Auf diesem Formular wird allgemein die Object-Code-Generierung gesteuert. Auch können Präferenzen für die Code-Optimierung gewählt werden.



Unter "***Memory***" wird das Speichermodell ausgewählt, das beim Compilieren und Linken verwendet werden soll.



Als Speichermodell wählen wir "***SMALL***", was bedeutet, dass der Code unseres Programms nicht größer als 64 KB sein darf und alle Programmverzweigungen (Unterprogramm-Aufrufe, bedingte und unbedingte Sprünge) innerhalb des 64 KB-Segmentes durchgeführt werden. Da die Demo-Version ohnehin ein 4 kB-Limit vorgibt, sind andere Speichermodelle nicht sinnvoll. Wählen Sie die Optionen, wie im obigen Screenshot dieser Dialogbox ersichtlich.

Die Dialogboxen "***PEC***" und "***Misc***" sind für unser erstes Beispielprogramm irrelevant. Sie können also jetzt den Knopf "***OK***" anklicken und die Einstellung der Compiler-Optionen beenden.

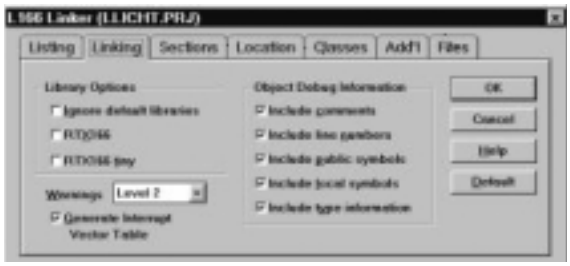
6.4.2 Linker-Optionen

Nun folgen die Einstellungen für den 166-Linker/Locater. Der Linker/Locater fügt die vom Compiler übersetzten Dateien und die Bibliotheksfunktionen zusammen und löst Symbole (Namen von Variablen, Konstanten, Unterprogrammen) in Adressen auf. Dazu muss er insbesondere Informationen darüber haben, in welche Speicherbereiche Code und Daten gelegt werden dürfen.

Wählen Sie im Menü den Punkt "***Options - L166 Linker***". Es erscheint eine Dialogbox mit den Tabs "***Listing***", "***Linking***", "***Sections***", "***Location***", "***Classes***", "***Add'l***" und "***Files***".

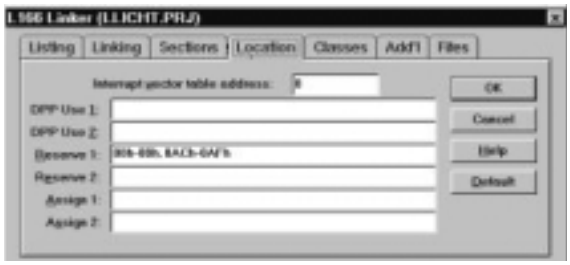
Das Formular "***Listing***" erlaubt die Angabe von Optionen für die vom Linker erzeugte Mapping-Datei. Die Voreinstellungen (siehe Screenshot) passen für unser Vorhaben.

Die folgende Seite ist Link-Optionen und Debug-Informationen gewidmet. Die Standardvorgaben laut Screenshot sollten unverändert übernommen werden.



Die Seite "***Sections***" können Sie für unser Beispiel leer lassen.

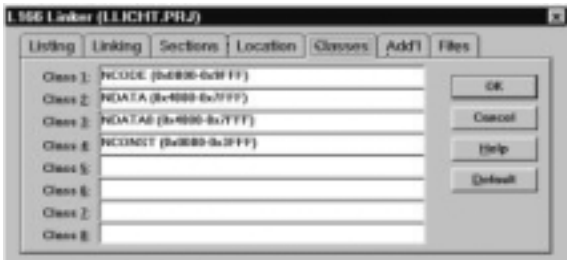
Nun folgt die Seite "***Location***". In der Zeile "***Reserve 1***" ist einzugeben:



08h-0Bh, 0ACh-0AFh

Diese Adressbereiche werden dadurch reserviert. Es handelt sich um Bereiche der Interrupt-Vektor-Tabelle. Das Monitor-Programm verwendet den NMI-Interrupt und den Receive-Interrupt der seriellen Schnittstelle. Aus diesem Grund dürfen diese Adressen nicht für den Link / Locate - Vorgang unseres Programmes verwendet werden.

Die nächste Seite "***Classes***" definiert die Adressbereiche für den Programm-Code (NCODE), die initialisierten und nicht-initialisierten Daten (NDATA0 und NDATA), sowie für die Konstanten (NCONST). Diese Angaben werden vom Linker verwendet, wenn dieser Symbole in absolute Adressen wandelt.



ACHTUNG: Die Eintragungen auf dieser Seite hängen davon ab, ob das Programm mit dScope/Monitor in das SRAM oder mit FLASHT in das Flash-Memory geladen werden soll.

Für die in ***4.1*** beschriebene Möglichkeit 1 (Programm wird vom Debugger in das SRAM geladen) können folgende Zeilen eingetragen werden:

NCODE (0x0000-0x9FFF)
NDATA (0x4000-0x7FFF)
NDATA0 (0x4000-0x7FFF)
NCONST (0x0000-0x3FFF)

Für nähere Informationen verweise ich auf meinen Artikel zum KEIL-Monitor ***[1]***.

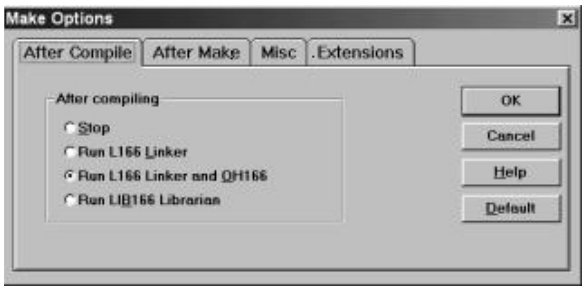
Die Einstellungen für das Laden eines Anwendungsprogramms in den Flash-Speicher beschreiben wir später (***siehe Kapitel 8***).

Damit sind die Linker-Optionen vollständig festgelegt.

6.4.3 Weitere Einstellungen

Wenn Sie Ihr Anwendungsprogramm später in den Flash-Speicher des Phytec-Boards laden möchten, ist dazu eine Datei im Intel-Hex-86-Format erforderlich. Dazu sind folgende Einstellungen erforderlich:

Wählen Sie den Menüpunkt ***Options – Make ....*** In der darauf erscheinenden Dialogbox klicken Sie auf der Seite ***After Compile*** die Option ***Run L166 Linker and OH166*** an.



Anschließend wird unter dem Menüpunkt ***Options - OH166 Object-Hex Converter*** eingestellt, welches Format die generierte HEX-Datei haben soll. Für unsere Zwecke ist als ***Output File Format*** die Option ***Intel Hex-86*** zu aktivieren.

6.4.4 Build Project

Nun kommen wir zum spannenden Augenblick. Das Keil-System ist bereit, unser Projekt in ein lauffähiges Programm zu übersetzen. Klicken Sie dazu den Menüpunkt ***Project - Make: Build Project*** an. Assembler, Compiler, Linker/Locater werden automatisch aufgerufen und die Ausgabe-Dateien werden generiert. In unserem Fall sind das die Dateien

LLICHT.OBJ	Object-Datei zum C-Sourceprogramm
LLICHT.LST	List-Datei (für den Compiler-Lauf)
LLICHT	die absolute Object-Datei
LLICHT.M66	Linker-Map-Datei

Eventuell wird auch LLICHT.H86 (die Intel-Hex-86-Datei) generiert (***gemäß Abschnitt 6.4.3***).

Beachten Sie bitte: die absolute Object-Datei, die wir nun anschließend mit Hilfe des Monitors in die Zielhardware laden werden, hat keine File-Extension.

7. Der Simulator/Debugger dScope/tScope

Ein sehr interessantes, umfangreiches (aber auch komplexes) Tool in der Keil-Software-Kette ist der Simulator/Debugger (dScope).

Starten Sie das Programm durch die Auswahl des Menüpunktes "***Run dScope Debugger***" (auf der Festplatte heißt das Programm DSW166.EXE und es befindet sich im Verzeichnis C:\C166EVAL\BIN). Mit dScope können Sie Ihr Anwendungsprogramm entweder am PC simulieren oder aber über ein Monitorprogramm in die Zielhardware laden und die üblichen Funktionen eines Debuggers verwenden. Dazu gehören:

- Schrittweiser Programmablauf
- Setzen / Löschen von Breakpoints
- Memory-Dumps
- Anzeigen von SFR-Inhalten
- De-Assemblieren von Code
- und vieles mehr

Wir werden uns hier auf das Arbeiten mit **dScope** als Debugger für das Programm in der Zielhardware beschränken (in dieser Form heißt der Debugger **tScope**).

7.1 Monitor und Anwendungsprogramm in die Zielhardware laden (SRAM)

Vergewissern Sie sich, dass Sie das Phytec kitCON-Board mit dem seriellen Kabel, das dem Starterkit beiliegt, verbunden ist. Für die folgenden Aktionen muss am kitCON-Board der **Bootstrap-Modus** aktiviert werden. Der Bootstrap-Modus erlaubt über eine fest in den C167 programmierte Codesequenz das Laden von 32 Byte Programmcode. Dieser 32 Byte Code kann dann weitere Programmteile laden. Hier ist nun zu unterscheiden, welche kitCON-Variante Sie in Ihrem Starterkit vorgefunden haben. Auf dem kitCON-167-1998 ist der **Schalter 1** von **SW3** auf **ON** zu stellen. Besitzer des kitCON-167-1997 müssen den Jumper **JP2** mit dem (roten) Stecker schliessen (Pins 1 und 2 dieses Jumpers verbinden). Drücken Sie in beiden Fällen danach die **RESET**-Taste am kitCON-Board.

Der tScope-Debugger benutzt den Bootstrap-Mechanismus, um das Monitorprogramm in das SRAM des Phytec-Boards zu laden. Zunächst wird über die C167-Bootsequenz das Programm BOOT geladen und gestartet. Diese Software lädt anschließend das wesentlich größere Programm MONITOR. Wie in ***Kapitel 5*** beschrieben wurde, müssen sich die beiden absoluten Object-Dateien `BOOT` und `MONITOR` im Verzeichnis `C166EVAL\BIN` befinden. Das Monitorprogramm kommuniziert über die serielle Schnittstelle mit der tScope-Oberfläche. Kommandos können so an das Monitorprogramm gesandt und ausgeführt werden (z.B. das Laden eines Anwendungsprogramms). Außerdem erlaubt der Monitor einen Blick in das "Innenleben" des C167-Mikrocontrollers. Wird die Anwendungssoftware im Einzelschritt-Betrieb abgearbeitet oder erreicht die Ausführung einen vorher definierten Breakpoint, werden die Inhalte der SFRs an tScope übertragen, wo sie angezeigt werden können. Ebenso können vom Monitor die Werte von Speicherbereichen abgerufen oder auch verändert werden. Sie können auf diese Art genau mitverfolgen, wie Ihr Programm die Speicherzellen und Registern verändert.

7.1.1 Monitor laden

Wir laden nun den Monitor mit Hilfe des Bootstrap-Loaders in das SRAM des kitCON.

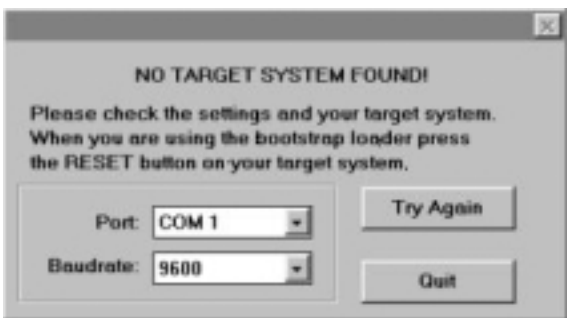
Erste Methode: Klicken Sie die Auswahlliste links oben an und wählen Sie `MON166.DLL`

Zweite Methode: Öffnen Sie das Command-Window (***View - Command Windows***) und geben Sie ein:

```
load mon166.dll
```

`MON166.DLL` ist die Schnittstelle zwischen dem Monitor-Programm (Target-Monitor), das auf dem Phytec-Board läuft und dem tScope-Debugger, der interaktiven PC-Oberfläche.

Es sollte ein Fenster mit einer Balkenanzeige erscheinen, die den Fortschritt des Ladevorgangs anzeigt.



Sollten Sie die Fehlermeldung "***NO TARGET SYSTEM FOUND***" erhalten, ist entweder die serielle Schnittstelle nicht richtig ausgewählt (COM1, COM2), oder aber die Baudrate falsch eingestellt. Sie sollten 9600 Baud verwenden, da der Monitor für diese Baudrate konfiguriert wurde. Im Artikel ***[1]*** wird beschrieben, wie höhere Geschwindigkeiten eingestellt werden können (dazu muss der MONITOR neu generiert werden).

ACHTUNG

Wir wollen an dieser Stelle nochmals darauf hinweisen, dass die Dateien `BOOT` und `MONITOR` aus dem Verzeichnis

```
CDROM\STARTKIT\SK_167\MONITOR\KEIL
```

der Starter-Kit-CD in das Verzeichnis

```
C166EVAL\BIN
```

der Keil-Installation kopiert werden müssen. Nach der Installation der Keil-Toolkette befinden sich dort Dateien `BOOT` und `MONITOR`, die nicht für das kitCON-Board gedacht sind. Auch in diesem Fall erhalten Sie die obige Fehlermeldung.

Im Command-Window sollte jetzt die Meldung erscheinen, wie sie im Bildschirmfoto am Beginn dieses Abschnitts zu sehen ist.

7.1.2 Anwendungsprogramm laden

Nun können wir unser Lauflicht-Programm in das SRAM laden. Auch hier gibt es zwei Methoden.

Erste Methode: Klicken Sie auf das linke Symbol ("geöffneter Ordner"). (siehe folgenden Screenshot)

Im darauf erscheinenden Dateiauswahlfenster selektieren Sie die absolute Object-Datei `LLICHT` (ohne Extension) aus unserem Projekt-Verzeichnis `C:\C166EVAL\PROJECTS\`

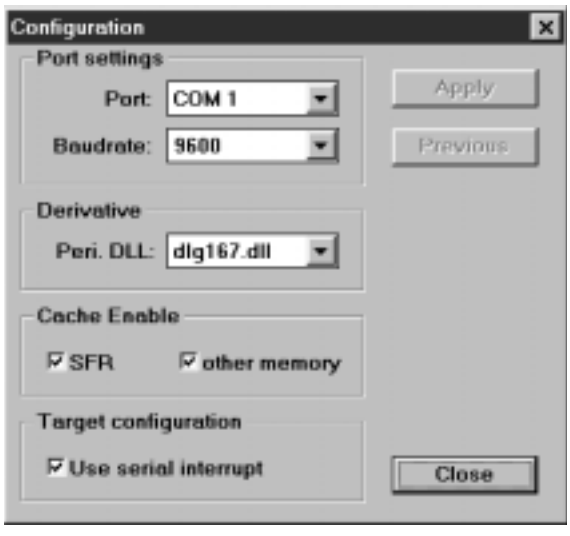
Zweite Methode: Geben Sie im Command-Window den Befehl

```
load LLICHT
```

ein.

7.1.3 Zielsystem wählen

Da die KEIL-Software sowohl für Systeme mit dem C166- als auch mit dem C167-Mikrocontroller entwickelt wurde, ist nun noch das Zielsystem auszuwählen. Wählen Sie den Menüpunkt ***Peripherals - Config*** und stellen Sie die Optionen so ein, wie dies das Bildschirmfoto zeigt.



WICHTIG: Unter "Derivative" ist als "Peri.DLL" die Datei DLG167.DLL zu wählen, da unser Zielsystem einen C167-Mikrocontroller enthält.

7.1.4 Anwendungsprogramm starten

Klicken Sie nun das Command-Window an und geben Sie dort den Befehl

```
g
```

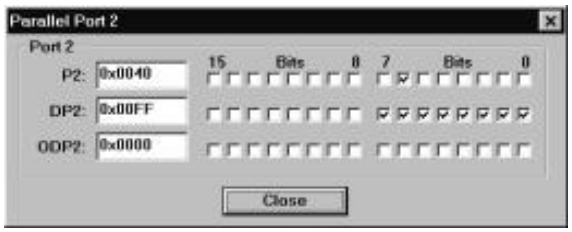
ein. Dieses Kommando (g = go) startet unser Anwendungsprogramm. Alternativ kann im Debug-Fenster der Menü-Punkt ***GO*** gewählt werden.

Die Leuchtdioden sollten im programmierten Muster aufleuchten.

Um das Programm anzuhalten, brauchen wir nur die ESC-Taste drücken (das Command-Fenster muss dazu allerdings das aktuelle Fenster sein - eventuell vorher anklicken). Auch der Menüpunkt "***STOP***" im Debug-Fenster hält ein Programm an.

Wenn das Programm angehalten wurde, kann der Inhalt von C167-Registern betrachtet werden. Unter dem Menü-Punkt ***View*** finden Sie zahlreiche Fenster, die zum Debuggen Ihres Programmes geöffnet werden können. Normalerweise ist zumindest das REGS-Fenster offen. Es zeigt die wichtigsten CPU-Register.

Die verschiedenen SFRs zur integrierten Peripherie des C167-Mikrocontrollers können Sie sich über den Menü-Punkt ***Peripherals*** anzeigen lassen. Unser Lauflicht-Programm arbeitet u.a. mit dem Port 2 und dem Timer 3. Sie können also beispielsweise die zugehörigen Fenster abrufen. Im ***Parallel Port 2*** - Window sollte genau das Bit angekreuzt sein, das HIGH-Pegel führt. Die damit verbundene Leuchtdiode sollte leuchten.



7.1.5 Automatisches Laden des Monitors und der Object-Datei

Die Keil-Oberfläche (µVision 1.xx) erlaubt es, den Monitor und das Object-File nach dem Aufruf von dScope automatisch in das RAM des kitCON-Boards zu laden. Dazu müssen wir lediglich eine kleine INI-Datei erstellen.

Beenden Sie dScope. Das Anwendungsprogramm muss vorher durch ESC unterbrochen werden. Sie befinden sich wieder in der µVision -Oberfläche. Wählen Sie ***File - New*** und geben Sie in das Editor-Fenster ein:

```
load mon166.dll
load llicht
```

Speichern Sie diese Datei unter dem Namen LLICHT.INI ab.

Nun geben Sie unter ***Options - dScope Debugger*** den Namen LLICHT.INI ein.

Wenn Sie jetzt ***Run - dScope Debugger*** anklicken, wird das Monitor- und das Object-Programm von dScope automatisch geladen. Befindet sich der Monitor noch im SRAM-Speicher, wird er nicht erneut geladen.

8. Programm-Entwicklung für das FLASH-Memory

Wie heute in der Praxis üblich, werden wir unsere Beispielprogramme in der höheren Programmiersprache C kodieren. Wenn C-Programme für PC-Systeme mit DOS / Windows erstellt werden, steht für deren Ausführung ein leistungsfähiges Betriebssystem zur Verfügung, das den Prozessor und die Peripherie bereits in einen vordefinierten Zustand gebracht hat.

Unser Mikrocontroller hingegen ist eine Stand-alone-Hardware. Für die geeignete Konfiguration der zahlreichen Komponenten unseres C167-Systems müssen wir daher selbst sorgen. Erst wenn gewisse SFRs (special function register) des C167 entsprechend gesetzt wurden, kann das Hauptprogramm main() unseres C-Sourcemoduls erfolgreich ablaufen. Unbedingte Voraussetzung dafür ist die korrekte Konfiguration des externen Buscontrollers, dessen Hauptaufgabe das Timing und die Adressierung (Chip-Auswahl) am externen Bus ist, an den die Flash- und SRAM-Speicherbausteine angeschlossen sind (START-UP-Konfiguration).

Die Startup-Konfiguration unseres Phytec-kitCon wird von einem Assembler-Modul vorgenommen, das wir mit dem leistungsfähigen Programm **DAvE** erstellen können. Es würde den Rahmen sprengen, hier auf alle Funktionen von DAvE einzugehen. Im wesentlichen ist DAvE eine Software, die es erlaubt, alle Komponenten des C167-Mikrocontrollers (oder anderer Infineon-µCs) zu konfigurieren und danach sowohl die erforderliche Assembler-Startup-Datei, aber auch C-Rahmenprogramme zu generieren. Wir werden uns hier zunächst auf die Erzeugung der Assembler-Startup-Datei beschränken.

8.1 Installation von DAvE (Version 1.x)

DAVE liegt dem Starterkit in Form einer eigenen CD-ROM bei. Legen Sie diese CD ein und starten Sie das Programm SETUP.EXE im Verzeichnis SETUP.

8.2. Generieren der STARTUP-Datei

Starten Sie nach der erfolgreichen Installation das Programm .EXE. Wählen Sie den Menüpunkt "***Project - New***" aus. Sie werden nun nach dem Mikrocontroller-Typ gefragt. Wählen Sie aus der Liste "***C167CR***" aus.

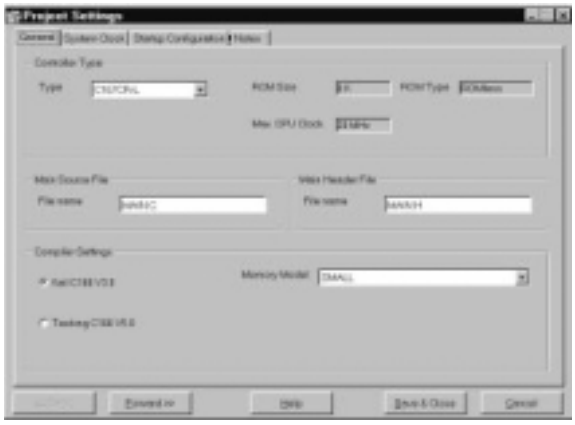
Anschließend müssen Sie Ihrem Projekt einen Namen geben. Wir wollen eine Startup-Datei erstellen, damit Anwendungsprogramme ins Flash programmiert und exekutiert werden können. Geben Sie als Name *PK167F* ein (PK für Phytec kitCON, F für Flash). Wir schlagen vor, dass Sie im Verzeichnis C:\C166EVAL\PROJECTS ein entsprechendes DAvE-Verzeichnis anlegen und dieDAvE-Projekt-Datei in dieses Verzeichnis speichern.

Auf der nun erscheinenden Formular-Seite wählen Sie unter "***Type***" den Chip "***C167CR-L***", der sich auf dem Phytec-kitCON-167-Board befindet.

Unter "***Compiler Settings***" klicken Sie die Option "***Keil C166 V3.0***" an. Im Eingabefeld "***Memory Model***" wird "***SMALL***" eingestellt. Danach drücken Sie auf den Knopf "***Forward >>***".

Wir kommen so auf die nächste Seite. Hier ist die Taktfrequenz einzustellen. Unser Board arbeitet mit einer externen Frequenz von 5 MHz, die durch die C167-interne PLL vervierfacht wird. Die CPU arbeitet somit intern mit 20 MHz. Ihre Einstellungen soll-

ten so vorgenommen werden, wie dies aus dem Screenshot ersichtlich ist.



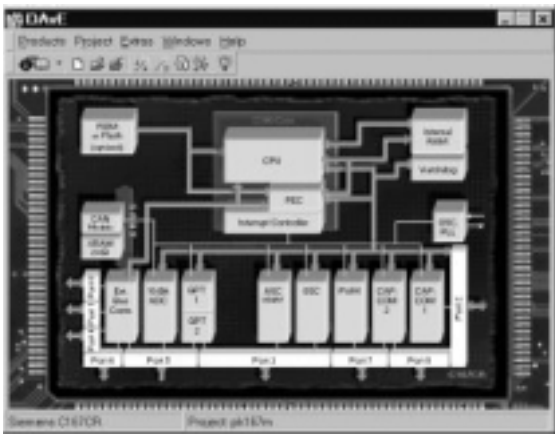
Weiter geht es wieder mit dem Button "***Forward >>***". Nun folgt die Einstellung der Startup-Konfiguration, die Sie gemäß dieses Bildschirmfotos eingeben sollten:

Aktiviert sein müssen folgende Optionen:

- Fetch code from external ROM
- Pin #WR and #BHE operate als #WRL and #WRH
- 4-Bit segment address
- 16 bit demultiplexed bus
- Five Chip Select Lines: #CS4 .. #CS0

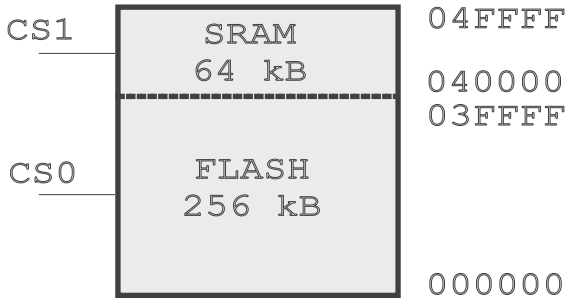
Damit ist die erste Serie von Einstellungen abgeschlossen. Drücken Sie den Knopf "***Save & Close***".

Nun kommen wir zur Konfiguration der C167-Komponenten. Wir sehen ein Bild über das "Innenleben" des C167-Mikrocontrollers. Bewegen Sie die Maus über die verschiedenen Symbole und drücken Sie die rechte Maustaste. Sie sehen, dass man durch Klick zur Konfiguration einer Komponente fortschreiten oder aber die entsprechene Seiten aus dem User-Manual anzeigen lassen kann (dazu muss der Acrobat Reader auf Ihrem System installiert sein, der sich ebenfalls auf der DAvE-CD-ROM befindet).



Für unser Lauflicht-Programm müssen wir nur eine Komponente konfigurieren - den "***External Bus Controller***".

Der "***External Bus Controller***" spielt in unserem System eine zentrale Rolle. Er ist dafür verantwortlich, dass der externe Speicher des kitCon korrekt aktiviert wird. Für ein tieferes Verständnis dieser Einheit lesen Sie bitte das kitCON-Handbuch und das ***Kapitel 8*** des User-Manuals.



Das Phytec kitCon-Board, das Sie mit dem Starterkit erhalten haben, verfügt über ein SRAM mit 64 KB (organisiert als 32 K x 16 bit) und ein FLASH-memory mit 256 KB (organisiert als 128 K x 16 bit). Das Flash-Memory wird durch CS0 (chip select 0), das RAM durch CS1 aktiviert. Der externe Bus-Controller ist äußerst flexibel und kann so konfiguriert werden, dass die externen Speicher mit dem korrekten Timing angesteuert werden. Insbesondere kann spezifiziert werden, in welchem Adressbereich des insgesamt 16 MB großen Adressraum welcher Speicher (oder auch Memory-mapped-Peripherie) liegen soll.

Im folgenden werden wir uns auf das SRAM- und das FLASH-memory des kitCONs beschränken. Wenn Sie weitere Speicher- oder andere Peripherie-Bausteine anschließen wollen, ist analog vorzugehen.

Im wesentlichen müssen wir nun mit Hilfe von DAvE die SFRs `SYSCON`, `BUSCON0`, `BUSCON1` und `ADDRSEL1` konfigurieren. Die `BUSCON`-Register legen das Timing und die Signalformen für die Ansteuerung der Speicherbausteine fest (`BUSCON0` für das FLASH memory - Speichertyp "ROM", `BUSCON1` für das SRAM - Speichertyp "RAM").

Das Register `ADDRSEL1` spezifiziert den Adressbereich für das SRAM. Der externe Buscontroller selektiert mit `CS1` das SRAM, wenn die CPU eine Adresse anspricht, die in dem durch den Wert von `ADDRSEL1` festgelegten Bereich liegt. Liegt die Adresse nicht in diesem Bereich, wird `CS0` und damit das Flash-Memory aktiviert.

Der externe Buscontroller wird natürlich nur dann tätig, wenn die angesprochene Adresse tatsächlich zu externem Speicher gehört und nicht etwa eine internes Register, internes RAM oder XRAM selektiert. Die Speicher-Organisation des C167 ist im ***Kapitel 3 des User-Manuals*** nachzulesen.

Wenn wir ein Programm in das Flash-Memory des Phytec-kitCON-167-Boards speichern möchten, haben wir für die Adresszuordnungen und damit insbesondere für die Konfiguration des Registers `ADDRSEL1` nicht viel Wahlmöglichkeiten. Der Code-Teil eines Programms muss beim C167 stets in einen Bereich gelegt werden, der bei der physikalischen Adresse 000000 beginnt, da dort die Interrupt-Vektortabelle gespeichert ist. Selbst wenn Sie keine Interrupt-Service-Routinen programmiert haben, muss zumindest der RESET-Vektor (auf die Adresse 0) geschrieben werden.

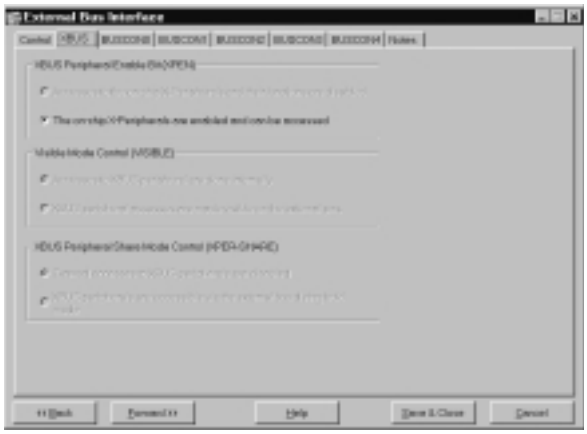
Aus diesen Überlegungen folgt, dass wir das Flash-Memory auf die Adresse 0 legen müssen. Das SRAM legen wir in den Bereich unmittelbar nach dem Flash.

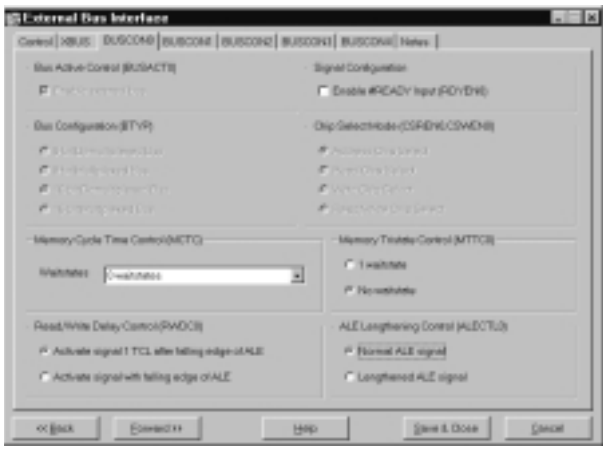
Speicherbereich	**Speichertyp**
000000-03FFFF	256 KB
040000-04FFFF	Flash-Memory 64 KB SRAM

Bemerkung: Wenn wir mit dem Monitor (und dScope) arbeiten und unser Programm in das SRAM speichern, ist eine andere Adresszuordnung erforderlich. In diesem Fall wird das Register `ADDRSEL1` von den Programmen `BOOT` und `MONITOR` so gesetzt, dass das SRAM auf der Adresse 0 liegt (***siehe Artikel [1]***).

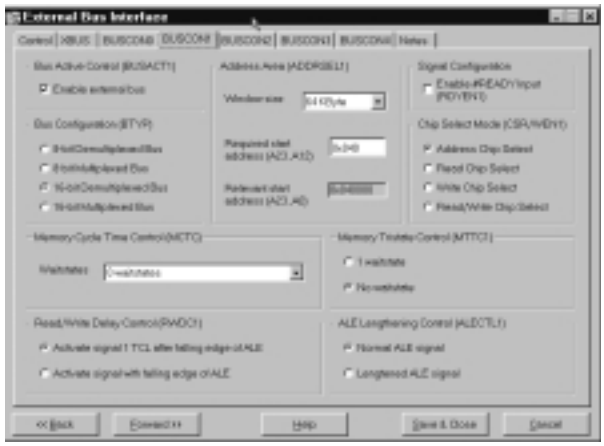
Nun aber zur Konfiguration der `BUSCON`- und `ADDRSEL`-Register. Bewegen Sie die Maus auf das Symbol "***Ext. Bus Contr.***" rechts unten, klicken Sie auf die rechte Maustaste und wählen Sie "***Configure***".

Stellen Sie der Reihe nach die Optionen auf den Formularseiten gemäß den folgenden Bildschirmfotos ein.





Das kitCON-167-Board hat nur zwei externe Speicherkomponenten (FLASH und RAM). Wir benötigen also keine weiteren Adressfenster. Aus diesem Grund werden für `BUSCON2`, `BUSCON3` und `BUSCON4` keine Einstellungen vorgenommen. Die dazugehörigen Seiten füllen Sie nicht aus, da die Standardvorgaben passen.



Einige Erläuterungen zu diesen Parametern:

Die wesentlichen Informationen für diese Einstellungen finden Sie im kitCON-167-Handbuch. Beide Speicherkomponenten (SRAM und FLASH) sind 16 Bit breit und werden über einen "demultiplexed bus" (getrennter Daten- und Adressbus) angesteuert. Die Parameter für das Bus-Timing sind im kitCON-Handbuch nachzulesen:

- 0 Waitstates (Memory Cycle Time Control)
- verzögertes R/W-Signal (Read/Write Delay Control)
- keine Tristate-Verzögerung (Memory Tristate Control)
- normales ALE-Signal (ALE Lengthening Control)
- kein READY-Signal

(Das ALE-Signal wird nur bei Multiplex-Bussen benötigt)

Drücken Sie "***Save & Close***", wenn Sie die vier Seiten "***Control***", "***XBUS***", "***BUSCON0***" und "***BUSCON1***" ausgefüllt haben.

Wir haben nun die Mindesteinstellungen für das Startup-File vorgenommen. DAvE kann daraus nun eine Assembler-Datei mit dem erforderlichen Startup-Code erzeugen. Dazu wählen wir im Menü den Punkt "***Project - Generate Code***". DAvE speichert eine Assembler-Datei mit dem Namen `START.ASM` in das Verzeichnis Ihrer Festplatte, das Sie vorher bei der Vergabe des Projekt-Namens gewählt haben. DAvE kann nun beendet werden. Es sei an dieser Stelle nochmals darauf hingewiesen, dass DAvE ein sehr umfangreiches Entwicklungstool ist, mit dem Sie sich eingehend beschäftigen sollten, wenn Sie später komplexere Projekte mit Ihrem C167-Mikrocontroller planen.

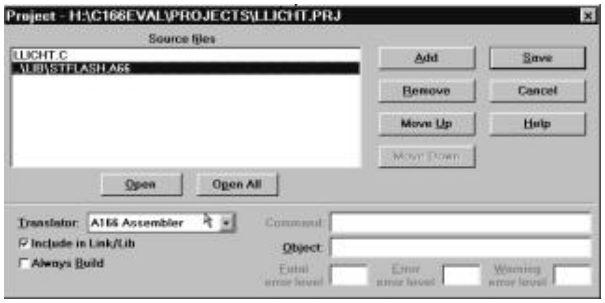
Kopieren Sie die von DAvE generierte Datei `START.ASM` in das LIB-Verzeichnis des KEIL-Systems. Wenn Sie bei der Installation das Default-Verzeichnis gewählt haben, ist dies `C:\C166EVAL\LIB`. Benennen Sie die Datei in `STFLASH.A66` um.

DAvE generiert nicht nur die Assembler-Datei `START.ASM`, sondern auch C-Dateien, die als "Rahmenprogramme" für die Software-Entwicklung verwendet werden können. Für unser einfaches Projekt bringen diese "Templates" keine großen Vorteile. Zum einen haben wir die C-Source-Datei `LLICHT.C` ja bereits erstellt und außerdem war es nur erforderlich, den externen Bus-Controller zu konfigurieren. Es sei aber noch einmal darauf hingewiesen, dass DAvE die Entwicklung umfangreicherer Projekte wesentlich erleichtert. Insbesondere wenn die verschiedenen internen C167-Komponenten Interrupts erzeugen sollen, sind die C-Rahmenprogramme, die von DAvE automatisch erstellt werden eine wesentliche Hilfe.

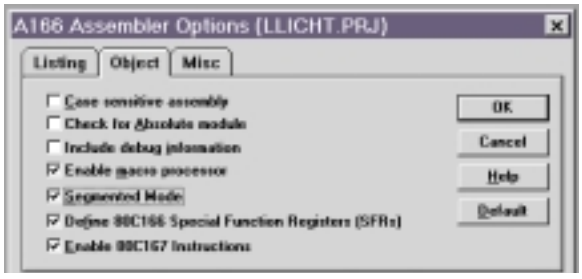
8.3 Einstellungen in der KEIL-Umgebung für das Speichern des Programms im FLASH

Damit unser Programm im Flash-Speicher laufen kann, müssen wir in der µVision-Umgebung einige wenige Änderungen gegenüber den im ***Abschnitt 6*** beschriebenen Einstellungen vornehmen.

Unser Projekt besteht jetzt nicht nur aus der C-Quelldatei .C, sondern auch aus der Assembler-Datei `STFLASH.A66`, die wir mit DAvE generiert haben. Fügen Sie die Datei `STFLASH.A66` Ihrem LLICHT-Projekt hinzu ("***Project - Edit Project - Add***"). Achten Sie darauf, dass die Option "***Include in Link/Lib***" aktiviert ist.



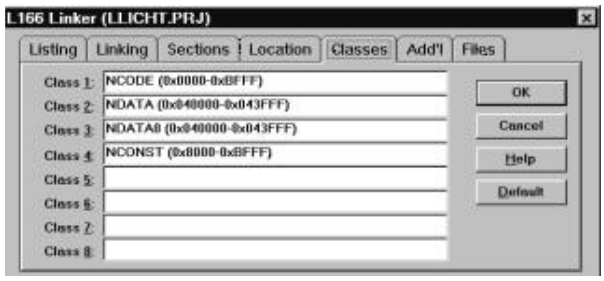
Da nun auch der Assembler am Erzeugen der Object-Datei beteiligt ist, sind auch für ihn Einstellungen unter dem Menüpunkt ***Options-A166-Assembler*** vorzunehmen. Klicken Sie, wie im folgenden Bild sichtbar, die Optionen ***Enable macro processor***, ***Segmented Mode***, ***Define 80C166 Special Function Registers*** und ***Enable 80C167 Instructions*** an.



Die Einstellungen für die CLASSES-Adressen sind wie folgt einzugeben ("***Options - L166 Linker***"):

```
NCODE       (0x0000-0xBFFF)
NDATA       (0x040000-0x043FFF)
NDATA0      (0x040000-0x043FFF)
NCONST      (0x8000-0xBFFF)
```

Der Programmcode (Class `NCODE`) und Konstante (Class `NCONST`) werden in das FLASH gespeichert, für Variable (Class `NDATA` und `NDATA0`) wird Speicher im SRAM reserviert. Konstante und Variable werden über die DPP-Register (siehe ***C167-Handbuch***) adressiert. Daher werden für sie jeweils 16 kB große Bereiche angegeben.



Für die Flash-Programmierung muss unser Programm im Intel-Hex-Format vorliegen. Diese Datei wird vom KEIL-Tool OH166 erzeugt. Damit dieser zusätzliche Schritt (***siehe Abschnitt 3***) durchgeführt wird, müssen wir unter "***Options - Make die Option Run L166 Linker and OH166***" anklicken. Die von OH166 erzeugte intel-Hex-Datei wird den Namen des Projekts mit der Dateiendung H86 tragen. In unserem Fall ist das `LLICHT.H86`.

Anschließend wählen wir im Menü "***Options - OH166 Object-Hex-Converter***" aus und stellen das Format Intel-HEX-86 ein.

Die Einstellungen sind damit abgeschlossen. Klicken Sie den Menüpunkt "***Project - Make: Build Project***" an und unsere Ausgabedatei `LLICHT.H86` wird gemäß unseren Angaben erstellt.

8.4 Das Flash-Programmiertool FLASHT

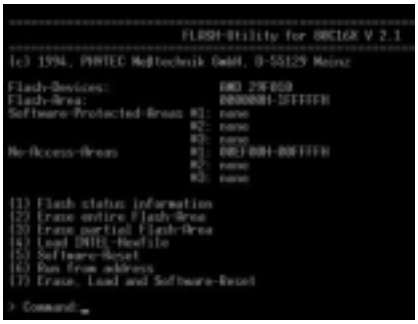
Zum Programmieren des Flash-Speichers benötigen wir ein entsprechendes Programm, das sich auf der Starterkit-CD-ROM im Verzeichnis

`CDROM\STARTKIT\SK_167\FLASH`

befindet. Kopieren Sie dieses Verzeichnis auf Ihre Festplatte. Das Flash-Tool heißt `FLASHT.EXE` und kann über die Batch-Dateien `FLASHT_1.BAT` (für COM1) bzw. `FLASHT_2.BAT` (für COM2) aufgerufen werden. Auf dem Phytec-kitCON-Board muss nun der Bootstrap-Modus aktiviert werden. Haben Sie das kitCON-167-1998, ist der Schalter 1 von SW3 auf ON zu stellen. Auf dem kitCON-167-1997 muss der rote Stecker so gesetzt werden, dass er die Pins 1 2 von Jumper JP2 schließt. Anschließend ist auf dem Board die RESET-Taste zu drücken.

FLASHT lädt über den Bootstrap-Mechanismus ein Monitor-Programm in den RAM-Speicher des Boards, das auch den Programmier-Algorithmus für das Flash enthält.

Das Flash-Tool von Phytec ist ein DOS-Programm, das aber auch in einem Fenster unter Windows 95/98 gestartet werden kann. Es bietet zahlreiche Optionen an, wie Sie auf dem folgenden Bildschirmfoto ersehen können.



Wählen Sie zunächst die Option 7 (***Erase, Load and Software Reset***). Das Flash-Memory wird gelöscht. Danach drücken wir die Taste F2 und geben den Namen der Intel-Hex-Datei ein, die in den Flash-Speicher geladen werden soll. In unserem Beispiel geben wir

`C:\166EVAL\PROJECTS\LLICHT.H86` ein.

Nach dem Laden des Programms führt das Flash-Monitor-Programm einen Software-Reset durch. Das Anwendungsprogramm sollte nun laufen. Da es sich nun resident im Flash befindet, kann es auch nach einer Unterbrechung der Spannungsversorgung jederzeit durch Drücken der Reset-Taste gestartet werden. Das Phytec-Board darf jedoch dazu NICHT in den Bootstrap-Modus geschalten werden. Auf dem kitCON-167-1998 ist der Schalter 1 von SW3 auf OFF zu stellen. Auf dem kitCON-167-1997 ist der (rote) Stecker von JP2 zu entfernen! Andernfalls würde durch ein Reset der Bootstrap-Loader und nicht unser Anwendungsprogramm gestartet werden.

9. STARTUP-Datei für RAM-Konfiguration

Es wird Ihnen wahrscheinlich aufgefallen sein, dass wir für das Arbeiten mit dem Debugger und das Laden unseres Anwendungsprogramms in das RAM keine DAvE-Startup-Datei generiert haben. Das ist auch nicht unbedingt erforderlich, da der KEIL-Linker beim Fehlen einer Startup-Datei im Projekt automatisch eine Default-Startroutine zum Programm linkt. Im Debug-Fenster von dScope sehen Sie nach dem Laden der Object-Datei auf der Adresse 0 (RESET-Vektor) den Aufruf

```
000000: JMPS C_STARTUP
```

Hier wird also gleich zu Beginn der Programmausführung diese Startup-Routine angesprungen. Die Source-Datei zum Default-Startup findet man in `C:\C166EVAL\LIB\START167.A66`

Dieser Source-Code wird allerdings nicht wirklich gelesen, wenn das lauffähige Programm erzeugt wird. Vielmehr ist der Objectcode bereits in der Library enthalten, die vom Linker Ihrem Programm hinzugefügt wird.

Wie bereits erwähnt, müssen Sie sich für das Arbeiten mit dem Monitor nicht unbedingt um eine Startup-Datei kümmern, was für den Anfänger eine zusätzliche Erleichterung darstellt. Fortgeschrittenen Benutzern wird allerdings (auch von KEIL) empfohlen, eine von DAvE generierte STARTUP-Datei hinzuzufügen, wie wir das auch in ***Abschnitt 8*** beschrieben haben. Ein Blick in die Default-Datei `START167.A66` zeigt nämlich, dass die C-Startup-Routine unter anderem für die Zugriffe über den externen Bus 2 Waitstates definiert, obwohl RAM und FLASH des Phytec-Boards auch mit 0 Waitstates funktionieren. Eine DAvE-generierte Startup-Datei nach ***Abschnitt 8*** bringt also geringe Geschwindigkeitsvorteile.

Eine DAvE-Startup-Datei erzeugen Sie, wie bereits in ***Kapitel 8*** beschrieben. Die einzige Änderung betrifft die Optionen für `ADDRSEL1` (auf der Seite `BUSCON1`). Geben Sie auf der entsprechenden Seite ein:

```
Window Size: 1 MB
```

der Monitor erfordert das, auch wenn Sie nur 64 kB physikalisches RAM haben

```
Required Start Address: 0x000
```

das RAM muss in den Adressbereich ab Adresse 0 gemappt werden

Die generierte STARTUP-Datei fügen Sie wie beschrieben Ihrem Projekt hinzu. Auch die Optionen für den Assembler sind, wie in ***8.3*** beschrieben, einzustellen.

Hintergrundinformationen finden Sie wieder in ***[1]***.

10. Schlussbemerkungen

"Aller Anfang ist schwer", schrieb ich zu Beginn dieses Artikels. Ich hoffe aber, dass diese Beschreibung Sie zum ersten erfolgreichen C167-Programm führen konnte. Die Darstellungen wurden bewusst ausführlich und an vielen Stellen mit Hintergrundinformationen und Tipps zum weiteren Selbststudium versehen. "Übung macht den Meister" stelle ich an das Ende des Dokuments - denn es gibt vieles zu erforschen - von den Möglichkeiten der KEIL-Umgebung über DAvE bis hin zum Innenleben des C167.

Viel Erfolg dabei!

Ergänzende und weiterführende Literatur und Web-Sites zum Thema des Artikels

[1] [Generieren des Target-Monitors für das Phytec-kitCON-167-Board und die Keil-Toolkette (Infineon C167-Starterkit), Walter Waldner, 1999
verfügbar über die Homepage des Autors:
http://www.htblmo-klu.ac.at/lernen/

[2] [Umfassende Informationen über die Infineon-Mikrocontroller und Programm-Updates für DAvE finden Sie auf der Web-Seite
http://www.infineon.com/
(Die **Siemens Semiconductor Group** wurde kürzlich in **Infineon** umbenannt)

[3] [Das Internet-Angebot der Firma KEIL finden Sie unter den Adressen *http://www.keil.com/*

[4] [Interaktive Online-Tutorials, insbesondere zu DAvE und CAN finden Sie unter der Adresse
http://www.mfuniversity.com/siemens/homepage.htm

[5] [EXBO – das I/O-Board für Mikrocontroller-Experimente zum Nachbauen. Informationen und Dateien zum Laden auf Ihren PC sind über die Homepage des Autors verfügbar:
http://www.htblmo-klu.ac.at/lernen/