Monitor für Mikrocontroller

Generieren des Target-Monitors für das Phytec-kitCON-167-Board und die KEIL-Toolkette (Siemens C167CR-Starterkit)

© Walter Waldner, Dezember 1998

1. Einleitung

Die KEIL-Toolkette für die Siemens 16-bit Mikrocontroller-Familie erlaubt die Software-Entwicklung mit dem Assembler bzw. dem ANSI-C-Compiler und das komfortable Laden, Simulieren und Debuggen der Anwenderprogramme (**dScope**). Eine ausführliche Beschreibung der KEIL-Entwicklungsumgebung (**µVision**) für das Phytec-kitCON-167-Board, das dem Siemens C167CR-Starterkit beiliegt, finden Sie in [1].

1.1 Beachten Sie die Starterkit-Version

Vom Siemens C167CR-Starterkit gibt es inzwischen zwei Versionen, die sich geringfügig unterscheiden. Die Ausführungen dieses Artikels gelten mit kleinen Änderungen für beide Starterkit-Ausgaben.

So erkennen Sie Ihr Starterkit:

Starterkit 1997

Phytec-Hardware Manual
Version 1.0 6/1997

Starterkit-CD-ROM: Edition 2.1

Starterkit 1998

Phytec-Hardware Manual
Version 2.0 7/1998

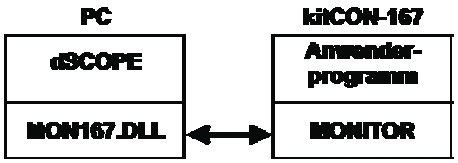
Starterkit-CD-ROM: Edition 3.2
Das Phytec-kitCON-Board enthält gegenüber der Vorversion 16 LEDs, die mit Port 2 verbunden sind.

In diesem Artikel wird vor allem das KEIL-Entwicklungstool dScope (ein MS-Windows-Programm für den PC) beschrieben. Die Ausführungen beziehen sich stets auf den Mikrocontroller C167CR bzw. das entsprechende Siemens C167CR-Starterkit. Falls es Unterschiede zwischen den beiden Starterkit-Versionen gibt, sind diese explizit angeführt.

dScope vereint zwei Funktionen unter einer einheitlichen Oberfläche:

- **C167-H*ardware-Simulator***
  dScope bildet die Funktionen des Mikrocontrollers durch Software nach und erlaubt so die Simulation von C167-Anwenderprogrammen
- ***Debugger* (*Target-Simulator*)**
  Das Anwenderprogramm läuft auf der Ziel-Hardware (in diesem Fall dem Phytec kitCON-167-Board). dScope kommuniziert mit der Ziel-Hardware und ermöglicht verschiedene Debug-Funktionen. KEIL nennt den Debugger auch **tScope**. In diesem Artikel werden wir für den Debugger aber stets die Bezeichnung dScope verwenden.
  Der Debugger (tScope) und der Hardware-Simulator (dScope) bieten dieselbe Benutzeroberfläche.

Für das Arbeiten mit dem dScope-Debugger muss in den Speicher (RAM bzw. FLASH Memory) des Phytec kitCON-167-Boards ein Monitor-Programm geladen werden, das über eine spezielle DLL und über die serielle Schnittstelle mit dem dSCOPE-Programm am PC kommuniziert.



Die dScope-Oberfläche und das Monitorprogramm ermöglichen unter anderem folgende Funktionen:

- Laden von Anwendungsprogrammen in das SRAM
- Starten / Unterbrechen von Programmen
- Schrittweise Programmausführung auf Source-Code-Ebene
- Setzen / Löschen von Breakpoints
- Betrachten der CPU-internen Register und SFRs
- Darstellung der Onchip-Peripherals des C167
- Anzeige von Unterprogramm- und Interrupt-Service-Routinen-Aufrufen
- Anzeige / Verändern von Speicherinhalten und Variablen
- Disassemblieren von Code-Sequenzen
- Performance-Analyse

Für das Zusammenspiel von dScope und dem Monitorprogramm gibt es eine Reihe von Konfigurationsmöglichkeiten, die in diesem Artikel beschrieben werden. Die Optionen sind im wesentlichen:

- MONITOR ins SRAM oder FLASH laden
- Kommunikation über die integrierte serielle Schnittstelle `ASC0` des C167 oder über eine software-simulierte serielle Schnittstelle

Außerdem können die Baudraten für die Kommunikation unterschiedlich eingestellt werden. Die Source-Dateien für das Erstellen individuell konfigurierter Monitor-Dateien sind Teil der Demo-Version der Keil-Toolkette, die sich auf der Starterkit-CD-ROM befindet.

2. Generieren des Monitor-Programms

Wir nehmen an, dass die KEIL-Toolkette in das Verzeichnis

```
C:\C166EVAL
```

installiert wurde. Im Unterverzeichnis

```
C:\C166EVAL\MON166
```

befinden sich die Source-Codes zum Erzeugen der Dateien `BOOT` und `MONITOR`. `BOOT` wird vom internen Bootstrap-Loader des C167 geladen und gestartet und wird benötigt, um das Monitor-Programm in das SRAM des kitCON-Boards zu bringen. Zum besseren Verständnis der folgenden Ausführungen sollten Sie zunächst die Datei README.TXT im obigen Verzeichnis lesen.

Für das KitCON-Board sind die Source-Dateien

```
CONFPHY7.INC
BOOTPHY7.A66
INSTPHY7.A66
```

relevant. In der Datei `CONFPHY7.INC` werden Parameter für die Konfiguration der `SYSCON-` und der `BUSCON`-Register definiert. Diese Datei wird beim Assemblieren von `BOOTPHY7.A66` und `INSTPHY7.A66` gelesen.

Obwohl die `README.TXT`-Datei im MON166-Verzeichnis behauptet, dass die Dateien `CONFPHY7.INC, BOOTPHY7.A66` und `INSTPHY7.A66` für das kitCON-167-Board adaptiert wurden, sind dort einige Parameter falsch gesetzt.

2.1. Anpassen der Source-Dateien für das kitCON167-Board

In der Datei `CONFPHY7.INC` sind folgende Zeilen zu ändern:

```
_MCTC0      EQU    0
; Memory wait states is 0 (MCTC0 = 0FH).
_RWDC0      EQU  0
; 0 = Delay Time     0.5 States (Reset
Value)
_MTTC0      EQU  1
; 1 = No Delay Time  0   States
```

In der Datei `BOOTPHY7.A66` sind die folgenden Veränderungen durchzuführen:

Die Zeile

```
BFLDL SYSCON,#080H,#SYS_L
```

ist zu ersetzen durch

```
$IF (WRCFG_ENABLE == 1)
     BFLDL SYSCON,#084H,#SYS_L
$ELSE
     BFLDL  SYSCON,#004H,#SYS_L
$ENDIF
```

Starter-Kit 1998: Auf der diesem Starterkit beiliegenden CD-ROM (Edition 3.2) sind diese Änderungen in der Datei `CONFPHY7.INC` bereits vorgenommen worden.

Außerdem können die Zeilen

```
MOV         ADDRSEL2,#1008H
; 1MB RAM BANK2 (10:0000H - 1F:FFFFH)
MOV         BUSCON2,BUSCON0
```

gelöscht werden (bzw. durch ein Semikolon in der ersten Spalte als Kommentar markiert werden), da das kitCON-167-Board nur das Flash-Memory (wird durch das chip-select-Signal `CS0 [BUSCON0]` aktiviert) und den RAM-Speicher (wird durch `CS1 [BUSCON1, ADDRSEL1]` aktiviert) als externe Komponenten enthält. Es spricht allerdings nichts gegen das Verbleiben der Zeilen in der Source-Datei, außer Sie benötigen `CS2` in Ihre Applikation zum Ansteuern externer Komponenten. `CS2` ist auf dem kitCON-Board selbst nicht in Verwendung.

In der Datei `INSTPHY7.A66` sind folgende Einträge zu verändern:

Die Zeile

```
BFLDL SYSCON,#080H,#SYS_L
```

ist durch

```
$IF (WRCFG_ENABLE == 1)
        BFLDL   SYSCON,#084H,#SYS_L
$ELSE
        BFLDL   SYSCON,#004H,#SYS_L
$ENDIF
```

zu ersetzen.

Für die Zeilen

```
MOV         ADDRSEL2,#1008H
MOV         BUSCON2,BUSCON0
```

gelten die oben gemachten Aussagen.

Schließlich sind die Definitionen zur simulierten seriellen Schnittstelle falsch. Die Zeilen

```
T_LINE          BIT     P2.0
; Transmit Data Line TxD
T_OUT           BIT     DP2.0
; Port direction register for TxD
R_LINE          BIT     P2.1
; Receive Data Line RxD
R_IN            BIT     DP2.1
; Port direction register for RxD
```

müssen ersetzt werden. Hier sind Unterschiede für die beiden kitCON-Varianten zu beachten.

Wenn Sie das ältere kitCON167-Board besitzen (Ausgabe 1997), ist einzugeben:

```
T_LINE          BIT     P3.0
; Transmit Data Line TxD
T_OUT           BIT     DP3.0
; Port direction register for TxD
R_LINE          BIT     P3.2
; Receive Data Line RxD
R_IN            BIT     DP3.2
; Port direction register for RxD
```

Für das neuere kitCON167-Board (Ausgabe 1998 mit den LEDs an Port 2) ist einzugeben:

```
T_LINE          BIT     P3.9
; Transmit Data Line TxD
T_OUT           BIT     DP3.9
; Port direction register for TxD
R_LINE          BIT     P3.8
; Receive Data Line RxD
R_IN            BIT     DP3.8
; Port direction register for RxD
```

2.2. Varianten für die Verwendung des Monitor-Programms

Für das Arbeiten mit dScope/Monitor kann der Anwender zwischen folgenden Varianten wählen:

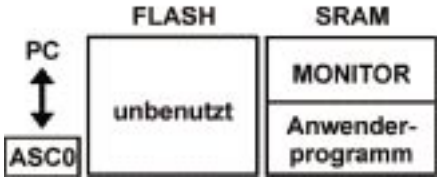
1 Monitor wird in das SRAM geladen
Die Kommunikation zwischen dScope und dem Monitor läuft über die serielle Schnittstelle des Phytec-Boards (DB9-Stecker P1)
Der Monitor wird über den C167-Bootstrap-Mechanismus und mit Hilfe des Programms `BOOT` geladen

2 Der Monitor wird mittels `FLASHT.EXE` in das Flash-Memory des Phytec-Boards programmiert. Die Kommunikation läuft über die serielle Schnittstelle des Boards (DB9-Stecker P1)

3 Der Monitor wird mittels `FLASHT.EXE` in das Flash-Memory des Phytec-Boards programmiert. Die Monitor-dScope-Kommunikation läuft über die simulierte serielle Schnittstelle (Debug-Interface DB9-Buchse P2)

Für die Variante 1 müssen die Absolute-Object-Dateien `BOOT` und `MONITOR` erzeugt werden, für die beiden anderen Varianten benötigt man die Intel-Hex-Datei `MONITOR.H86`, die dann mit dem Programmiertool `FLASHT.EXE` in das Flash-Memory programmiert wird. Im folgenden wird beschrieben, wie Sie diese Dateien erzeugen können. In jedem Fall sind zunächst die vorher beschriebenen Änderungen in den Dateien `CONFPHY7.INC`, `BOOTPHY7.A66` und `INSTPHY7.A66` erforderlich.

2.2.1. Variante 1



In dieser Variante wird nur das SRAM des kitCON-167-Boards und die Bootstrap-Sequenz des Mikrocontrollers verwendet. Das Bootstrap-Programm befindet sich im BOOT-ROM des C167 und wird aktiviert, wenn der BOOT-Mode gewählt wurde (der Jumper P2 1+2 muss mit dem [roten] Stecker geschlossen werden und die RESET-Taste am Phytec-Board muss gedrückt werden). Der Ablauf des Boot-Vorgangs ist wie folgt:

Schritt	Aktion
1	Die C167-Bootstrap-Sequenz wartet auf ein 0-Byte vom PC und sendet ID-Byte zum PC-Programm (in unserem Fall ist das dScope)
2	Die C167-Bootstrap-Sequenz lädt ein 32 Byte großes Preload-Programm ins RAM (das sind die ersten 32 Bytes der Datei BOOT)
3	Das Preload-Programm wird aktiviert und lädt den Rest des Programms BOOT ins RAM
4	BOOT wird aktiviert und lädt den MONITOR ins RAM
5	Der MONITOR wird aktiviert und kommuniziert mit dScope. dScope kann verschiedene Kommandos an den MONITOR senden(z.B. Laden des Anwenderprogramms ins RAM) bzw. Daten vom MONITOR empfangen und anzeigen

Die Kommunikation zwischen dScope und dem Monitor läuft in Variante 1 über die integrierte serielle Schnittstelle `ASC0` des C167-Controllers. Nachfolgend wird beschrieben, wie Sie die Dateien `BOOT` und `MONITOR` generieren können.

Öffnen Sie unter Windows das DOS-Fenster und wechseln Sie in das Verzeichnis

```
C:\C166EVAL\MON166
```

Nun geben Sie den Befehl

```
INSTALL PHY7 0 FEA FEC BOOTSTRAP
```

ein. Die Batch-Datei `INSTALL` erzeugt die Dateien `BOOT` und `MONITOR` (absolute Objekt-Dateien) und kopiert sie auch gleich in das richtige Verzeichnis `C:\C166EVAL\BIN`

Beachten Sie bitte, dass der C167 in den Bootstrap-Modus versetzt werden muß. Dazu wird auf dem Board - Ausgabe 1997 - der rote Jumper (JP2 1+2) gesetzt und die RESET-Taste gedrückt. Auf dem Phytec-Board – Ausgabe 1998 – ist der Schalter 1 auf SW3 zu schließen und RESET zu drücken.

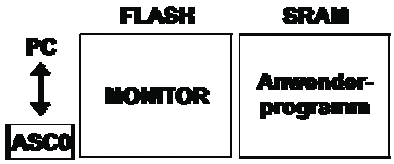
Außerdem muss in der µVision-Oberfläche unter "***Options - L166 Linker -Reserve 1***" eingetragen sein:

```
08h-0Bh, 0ACh-0AFh
```

Das Monitor-Programm verwendet den NMI (non maskable interrupt) als Breakpoint-Trap und den Empfangs-Interrupt der seriellen Schnittstelle `ASC0`. Daher müssen die oben angeführten Adressen in der Interrupt.Vektortabelle vor dem Überschreiben geschützt werden.

ACHTUNG: Der Monitor verwendet Interrupt-Level 15, Group-Level 0 für den `ASC0`-Receive-Interrupt (siehe README.TXT im MON166-Verzeichnis). Diese Kombination darf im Anwenderprogramm NICHT verwendet werden.

2.2.2. Variante 2



In diesem Fall wird der Monitor `MONITOR.H86` in das Onboard-Flash-EEPROM programmiert und muss nach Spannungsunterbrechungen des Boards nicht neu geladen werden. Das Anwenderprogramm wird vom Monitor in das SRAM geladen.

Auch in dieser Variante kommunizieren dSCOPE und der Monitor über die integrierte `ASC0`-Schnittstelle des C167-Microcontrollers. Zunächst beschreiben wir das Erzeugen der Datei `MONITOR.H86`, die mit dem Programmiertool FLASHT.EXE in das Flash-Memory programmiert werden kann.

Öffnen Sie unter Windows das DOS-Fenster und wechseln Sie in das Verzeichnis

```
C:\C166EVAL\MON166
```

Nun geben Sie den Befehl

```
INSTALL PHY7 0 FFE 2000
```

ein. Anschließend müssen Sie noch

```
..\BIN\OH166 MONITOR H167 OFFSET
(-0x200000)
```

eingeben. `OH166.EXE` konvertiert die "absolute object"-Datei `MONITOR` in eine intel-Hex-86-Datei `MONITOR.H86`.

Nun können wir das dadurch erzeugte File `MONITOR.H86` in das Flash-Memory laden.

Dazu verwenden wir das Programm `FLASHT.EXE`. Auf der Starterkit-CD-ROM finden wir es im Verzeichnis

```
X:\CDROM\STARTKIT\SK_167\FLASH
```

Zum Flash-Programmieren muss das Phytec-Board und der PC wird mit dem seriellen Kabel verbunden und der C167 in den **Bootstrap-Modus** gebracht werden. Für das **kitCON-Board 1997** ist der rote Stecker auf Jumper JP2 1+2 zu setzen und die RESET-Taste zu drücken. Für das **neuere Board** 1998 ist der Schalter 1 von SW3 zu schließen und RESET zu drücken. Das Flash-Programmiertool `FLASHT.EXE` kann über die Batch-Dateien FLASHT_1.BAT (für COM1) bzw. FLASHT_2.BAT (für COM2) oder auch direkt durch

```
flasht 1 br(9600) bzw.
flasht 2 br(9600)
```

gestartet werden. Die Parameter 1 bzw. 2 geben die Schnittstelle am PC an (1=COM1, 2=COM2), die Zahl nach br die Baudrate der Kommunikation.

Nach dem Laden der Programmiersoftware wird die Option 7 (Löschen des Flash-Memory) gewählt. Anschließend muss F2 gedrückt und der Pfad der Intel-Hex-Datei angegeben werden. In unserem Fall geben wir ein:

```
C:\C166EVAL\MON166\MONITOR.H86
```

Ist der Ladevorgang abgeschlossen, wird `FLASHT` mit F1 beendet.

kitCON167-Board 1997

Entfernen Sie nun den roten Stecker von Jumper JP2 (1+2) und drücken Sie RESET.

kitCON167-Board 1998

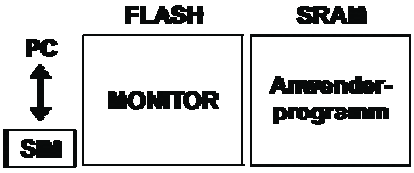
Öffnen Sie Schalter 1 auf SW3 und drücken Sie RESET.

Wie unter Variante 1 beschrieben, sind in der KEIL-Umgebung μVision unter den Linker-Optionen die Adressen

```
08h-0Bh, 0ACh-0AFh
```

zu reservieren und die Verwendung von Interrupt-Level 15, Group-Level 0 im Anwenderprogramm verboten.

2.2.3. Variante 3



Variante 3 ist eine interessante Möglichkeit mit dem Monitor zu arbeiten, die vor allem für fortgeschrittene C167-Anwender zu empfehlen ist. In diesem Fall wird das Monitor-Programm so generiert, dass es über eine software-simulierte serielle Schnittstelle mit dem PC kommuniziert.

Die vorher beschriebenen Varianten benutzen die C167-interne asynchrone serielle Schnittstelle `ASC0` zur Kommunikation zwischen Monitor und PC. Damit ist es aber nicht uneingeschränkt möglich, Anwenderprogramme zu testen, die selbst die asynchrone serielle Schnittstelle `ASC0` des C167 benötigen.

Unter "simulierte serielle Schnittstelle" versteht man, dass das Monitorprogramm über zwei beliebige I/O-Pins des C167-Mikrocontrollers das asynchrone Protokoll der `ASC0`-Schnittstelle durch Software nachbildet. Ein Pin wird als Sendeleitung (TxD), der andere als Empfangsleitung (RxD) verwendet. Nun waren die Entwickler des Phytec-Boards so freundlich, die Pins 0 bzw. 9 (TxD) und 2 bzw. 8 (RxD) des Ports 3 mit dem MAX232-Baustein zu verbinden (Pin 0 und 2 gelten für das ältere kitCON167-Board-1997, Pin 9 und 8 gelten für das neuere kitCON167-Board-1998). Der MAX232 kann für zwei Kanäle CMOS-Pegel in die von der RS232C-Schnittstelle erforderlichen +10/-10 Volt-Pegel bzw. umgekehrt wandeln. Da nur ein Kanal des MAX232 für die C167-interne serielle Schnittstelle verwendet wird, ist der zweite Kanal für die software-simulierte serielle Kommunikation frei.

Die Buchse P2 des Phytec-Boards kann wahlweise als CAN-Interface oder als Debug-Schnittstelle für die simulierte serielle Kommunikation verwendet werden. Die Auswahl geschieht über insgesamt drei Jumper. Damit wir die simulierte serielle Schnittstelle verwenden können, müssen wir:

- JP8 schließen (Phytec 1998: auf 2+3 setzen)
- JP9 und JP10 auf 2+3 umsetzen
- Auf dem Phytec-1998-Board ist zusätzlich noch JP2 zu schließen (auf 2+3 setzen)

Dadurch verbinden wir die MAX232-Leitungen mit der P2-Buchse (siehe Schaltpläne im kitCON-167-Handbuch). (Die Pin-Belegungen der Jumper können Sie im Phytec-Handbuch auf der Seite 23 nachlesen.)

Nun ergibt sich nur noch das Problem, dass P2 im Gegensatz zu P1 ein Stecker (männlich) ist und somit das Kabel aus dem Starterkit nicht passt. Besorgen Sie sich entweder einen sogenannten "Gender-Changer" (Buchse-Buchse / weiblich-weiblich) oder löten Sie sich ein entsprechendes Kabel, was sehr einfach ist, da nur drei Leitungen erforderlich sind. Dazu besorgen Sie sich zwei DB9-Buchsen und verbinden

- Pin 2 mit Pin 2 (TxD)
- Pin 3 mit Pin 3 (RxD)
- Pin 5 mit Pin 5 (Ground)

Das Erzeugen der Datei `MONITOR.H86` für die Kommunikation über die simulierte serielle Schnittstelle geschieht durch

```
INSTALL PHY7 2 FFE 2000
```

Damit wird die absolute Objekt-Datei MONITOR generiert. Anschließend müssen Sie wie bei Variante 2

```
..\BIN\OH166 MONITOR H167 OFFSET
(-0x200000)
```

eingeben. `OH166` erzeugt die intel-HEX-86-Datei `MONITOR.H86`. Zum Laden der Datei `MONITOR.H86` in das Flash-Memory ist wie unter Variante 2 beschrieben vorzugehen. Beachten Sie, dass der C167 wiederum in den Bootstrap-Modus geschalten werden muss (kitCON167-1997: Jumper JP2 1+2 schließen und RESET drücken, kitCON167-1998: Schalter 1 auf SW3 schließen und RESET

drücken). Für die FLASH-Programmierung wird der PC über das Starterkit-Kabel mit der seriellen Schnittstelle ASC0 des C167 verbunden (Buchse P1 !), da das Flash-Tool FLASHT.EXE den Bootstrap-Mechanismus und damit die integrierte serielle Schnittstelle ASC0 des C167 erfordert.

Nach dem Programmieren des Flash-EEPROMs wird auf dem 1997er kitCON167-Board der rote Stecker (JP2 1+2) entfernt, auf der 1998er-Ausgabe des kitCON-Boards wird Schalter 1 auf SW3 geöffnet. In jedem Fall ist anschliessend die RESET-Taste zu drücken. Der PC wird nun mit dem Phytec-Board über die Debug-Buchse P2 verbunden (Gender-Changer oder selbstgelötetes Kabel verwenden). Die interne asynchrone serielle Schnittstelle ASC0 des C167 (über Stecker P1) steht dem Anwenderprogramm voll zur Verfügung und kann wahlweise mit einer freien Schnittstelle des PCs oder anderer Zielhardware verbunden werden.

Die Variante 3 erfordert unter "***Options - L166 Linker - Reserve 1***" in der µVision-Umgebung nur den Eintrag

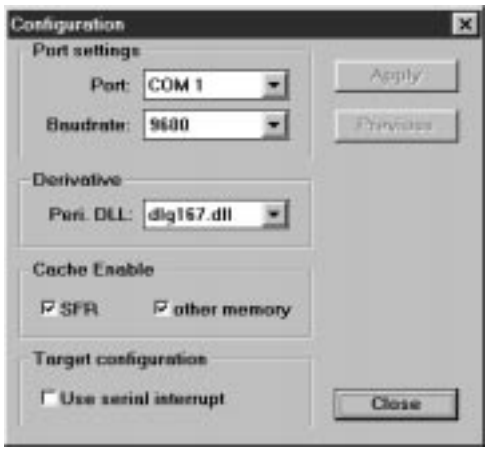
```
08h-0Bh
```

da in diesem Fall der ASC0-Receive-Interrupt vom Monitor nicht benutzt wird, sondern dem Anwenderprogramm zur Verfügung steht. Interrupt-Level 15 / Group-Level 0 ist frei zur Verwendung im Anwenderprogramm.

Im Programm dScope ist unter

"***Peripherials - Config. - Target Configuration***"

die Option "***use serial interrupt***" zu deaktivieren !



3. Einstellen der Baudrate

Wenn Sie mit der **Variante 1** arbeiten, erkennt der interne Bootstrap-Loader-Code des C167 die in dScope eingestellte Baudrate automatisch anhand eines Nullbytes, das die Kommunikation zwischen Bootstrap-Loader und dScope einleitet (siehe Kapitel "Bootstrap Loader" im C167-User Manual).

Für die **Variante 2** kann die Geschwindigkeit der serielle Kommunikation zwischen dem Target-Monitor (Mikrocontroller-Board) und der dScope-Oberfläche (am PC) in der Datei INSTPHY7.A66 eingestellt werden. Suchen Sie in der Datei INSTPHY7.A66 den Abschnitt "Initialization of Serial Interface 0". Dort ist eine der MOV-Anweisungen

```
;           MOV  S0BG ,#0040H
; SET BAUDRATE TO 9600 BAUD @ 40MHz
;           MOV  S0BG ,#0020H
; SET BAUDRATE TO 19200 BAUD @ 40MHz
;           MOV  S0BG ,#000FH
; SET BAUDRATE TO 38400 BAUD @ 40MHz
            MOV  S0BG ,#000AH
; SET BAUDRATE TO 57600 BAUD @ 40MHz
```

zu aktivieren. Das führende Semikolon deaktiviert eine Zeile als Kommentar, der beim Assemblieren überlesen wird. Um etwa 57600 Baud (die maximale Geschwindigkeit) einzustellen, löschen Sie das Seminkolon in der Zeile

```
MOV         S0BG ,#000AH
```

Alle anderen Optionen müssen durch ein Semikolon am Beginn der Zeile deaktiviert sein.

Für die **Variante 3** (simulierte serielle Schnittstelle) suchen Sie in der Datei INSTPHY7.A66 die Stelle, die mit "Initialization of simulated Serial Interface 2" überschrieben ist. Dort kann die Zeile

```
BAUDRATE        EQU     9600
```

auf

```
BAUDRATE        EQU     38400
```

abgeändert werden. 38400 Baud ist die maximale Geschwindigkeit für die simulierte serielle Schnittstelle. Sollten Kommunikationsprobleme auftreten, empfehle ich eine geringere Baudrate (z.B. 19200) zu wählen.

Noch eine Bemerkung: Ältere C167-Controller hatten keine PLL-Einheit zum Generieren des Systemtakts. Diese Bausteine erforderten einen externen 40MHz-Takt, der zur Erzeugung des Systemtakts mit exaktem duty-cycle von 50% intern durch zwei geteilt wurde. Aus diesem Grund ist in INSTPHY7.A66 stets 40 MHz die Berechnungsgrundlage. Die Zeile `XTAL EQU 40000000` darf **NICHT** geändert werden, da zur Berechnung der Zeitparameter für die serielle Kommunikation XTAL ohnehin durch zwei dividiert wird.

4. STARTUP-CODE

Wenn Sie mit dem Monitor (Varianten 1 bis 3) arbeiten, initialisiert dieses Programm die C167-Hardware bereits so, dass das Flash-Memory und das SRAM richtig angesteuert werden (Adressraum und Timing, Register SYSCON, BUSCON0, BUSCON1, ADDRSEL1). Ihr KEIL-Projekt erfordert keine spezielle STARTUP-Datei, da Ihr Programm in eine richtig konfigurierte C167-Umgebung geladen wird. Erst wenn Sie Ihr Anwendungsprogramm in das FLASH-Memory laden möchten, ist eine STARTUP-Datei erforderlich, die mit dem komfortablen Tool DAVE generiert werden kann.

5. EINSTELLUNGEN in der KEIL-Entwicklungsumgebung

Hier werden die wichtigsten Einstellungen für die Keil-Oberfläche beschrieben, die für das Arbeiten mit den verschiedenen Monitor-Varianten erforderlich sind.

In µVision (KEIL Software-Entwicklungsoberfläche) sind unter "***Options - L166 Linker - Location - Reserve 1***" die im Artikel beschriebenen Adressräume zu reservieren (für alle Varianten 08h-0Bh, bzw. 0ACh-0AFh für die Varianten 1 und 2)

Unter "***Options - L166 Linker - Classes***" sind die Adressbereiche einzustellen, die der Linker bei der Adresszuordnung für Anwenderdaten und -code verwendet. Alle drei hier beschriebenen Varianten laden das Anwenderprogramm in das SRAM. Der Compiler generiert, abhängig vom gewählten Speichermodell (in unserem Beispiel SMALL) und abhängig von den Steuerworten, die bei der Deklaration von Variablen, Konstanten und Unterprogramm verwendet werden, Klassen vom Typ Code oder Daten mit bestimmten Namen, denen der Locater gewisse Adressen zuweist.Wir können folgende Bereiche festlegen:

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

Die Classes bedeuten:

`NCODE`: Anwendercode
`NDATA`: nicht initialisierte Variable
`NDATA0`: initialisierte Variable
`NCONST`: Konstante

Nachdem ein Anwenderprogramm erstellt wurde ("***Project - Make: Build Project***), kann der dScope-Debugger aufgerufen werden. Wählen Sie im Hauptmenü den Punkt

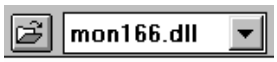
Run - dScope Debugger

dScope kann als Simulator und als Debugger arbeiten (in diesem Fall bezeichnet KEIL das Programm auch als tSCOPE). Durch die Wahl der entsprechenden DLL wird der entsprechende Arbeitsmodus

eingestellt. Wir beschränken uns hier auf die Verwendung als Debugger (tSCOPE).

Das kitCON-Board und der PC sind über das Starterkit-Kabel zu verbinden. Achten Sie darauf, dass nur für Variante 1 der Bootstrap-Modus gewählt sein muss (kitCON167-1997: den roten Jumper JP2 1+2 schließen, kitCON167-1998: Schalter 1 auf SW3 schließen). Für alle anderen Varianten muss der Jumper bzw. Schalter geöffnet sein. Der geschlossene Jumper bzw. Schalter aktiviert nach dem Drücken der RESET-Taste die C167-interne Bootstrap-Sequenz. Der MONITOR muss nur in Variante 1 mit dem Bootstrap-Mechanismus in das RAM geladen werden.

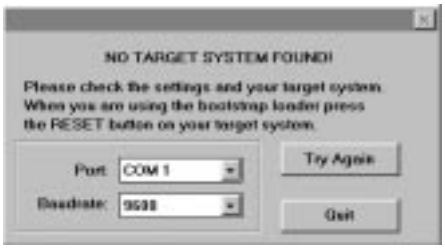
Wählen Sie nun im Auswahlfenster links oben die `MON166.DLL` aus. Durch die Auswahl der DLL



`MON166.DLL` setzen Sie dScope in den Debug-Modus (die DLL `80167.DLL` würde den Hardware-Simulator-Modus einstellen). Lassen Sie sich von der Bezeichnung `MON166.DLL` nicht verwirren. Die spezifische Auswahl des C167-Controllers für den Debugger erfolgt weiter unten.

Arbeiten Sie mit Variante 1, so sollte jetzt ein Fenster erscheinen, das das Laden des MONITOR-Programms in das SRAM des kitCON-Boards anzeigt. Die Varianten 2 und 3 erfordern kein Laden des Monitors, da er sich ja bereits im Flash-Memory befindet.

Sollten Sie die Meldung



erhalten, kommuniziert der Monitor nicht mit dem dSCOPE-Programm. Überprüfen Sie, ob das Kabel an der eingestellten PC-Schnittstelle angeschlossen ist und ob die in `INSTPHY7.A66` gewählte Baudrate mit der in dScope einzustellenden Geschwindigkeit übereinstimmt.

Die entsprechende Dialogbox erhalten Sie unter "***Peripherals - Conf.***" im dScope-Hauptmenü. Für die Varianten 1 und 2 kann "***Use serial interrupt***" aktiviert sein, für Variante 3 muss diese Option deaktiviert sein. Stellen Sie außerdem in jedem Fall unter "***Derivative - Peri. DLL***" die Option `dlg167.dll` ein, damit im Debug-Modus die Komponenten des C167-Microcontrollers korrekt angezeigt werden. Diese sind geringfügig anders als beim C166 (`dlg166.dll`).

Noch eine Bemerkung zur Option "***Use serial interrupt***": Wenn Sie für die Varianten 1 und 2 diese Option aktivieren, kann das Anwenderprogramm durch ESC (Command-Fenster) oder STOP (Debug-Fenster) angehalten werden. In diesem Fall kann das Anwenderprogramm jedoch die ASC0-Schnittstelle des C167 nicht für Text-Ein- und Ausgabe verwenden (z.B. printf, scanf). Einige Demoprogramme (im Verzeichnis `C:\C166EVAL\EXAMPLES`) verwenden diese C-Ein-/Ausgabe-Befehle.

Nun kann das Anwenderprogramm geladen werden. Klicken Sie dazu auf das Symbol links von der Auswahlliste, in der Sie `MON166.DLL` gewählt haben. Alternativ kann das Anwenderprogramm über "***File - Load Object File***" geladen werden. Anschließend sollten Sie zumindest das Debug-Fenster öffnen. Über das Debug-Fenster kann das Programm gestartet werden (Menüpunkt "***Go***") oder im Einzelschritt-Verfahren (Menüpunkte "***Step Into***", "***Step Over***") abgearbeitet werden. Alternativ können Sie das Command-Fenster öffnen und dort den Befehl g,main eingeben. Laufende Programme können im Debug-Fenster über "***Stop***" (oder alternativ dazu im Command-Fenster durch Drücken der ESC-Taste) angehalten werden. Außerdem können Breakpoints gesetzt werden. Nach jeder Unterbrechung des Anwenderprogramms bzw. Ausführung eines Befehls im Single-Step-Modus erfolgt ein Rücksprung in den Monitor. Der Monitor liest die aktuellen Werte von internen Registern des C167 aus und überträgt diese an die dScope-Oberfläche, wo sie dargestellt werden. Welche Register , Speicherbereiche, Variablen usw. angezeigt werden sollen, können Sie unter dem Menüpunkt "***Peripherials***" einstellen.

6. Insider-Informationen

Adressbereiche und Speicherbelegungen der Monitor-Programme

Zum Verständnis der nachfolgenden Informationen sollten Sie unbedingt vorher das Kapitel 8 des C167-User-Manuals (External Bus Controller) studieren.

Das kitCON-C167-Board des Starterkits enthält 256 KB Flash-Memory und 64 KB SRAM. Das Flash-Memory wird über `CS0` (chip select 0), das SRAM über `CS1` (chip select 1) aktiviert. Durch Konfiguration von `ADDRSEL1` kann festgelegt werden, für welchen Adressbereich der C167-Controller das CS1-Signal auf LOW (aktiv) setzt und damit das SRAM anspricht. Ein Blick in `CONFPHY7.INC` zeigt, dass `ADDRSEL1` mit der Hex-Konstanten `0008` geladen wird. Dadurch wird festgelegt, dass das erste Megabyte (`000000` bis `0FFFFF`) des Adressraums dem SRAM zugeordnet wird. Wird also eine Adresse im Bereich `000000` bis `0FFFFF` generiert, so ist damit stets eine Speicherzelle im RAM gemeint. Nun ist aber das SRAM physikalisch nur 64 KB groß. Das bedeutet, dass durch diese `ADDRSEL1`-Konfiguration das SRAM 16fach gespiegelt in das erste Megabyte eingeblendet wird, da das SRAM nur die untersten 16 Bit der Adresse interpretiert. Die Adressen `FEAC5, EEAC5, DEAC5, ..., 0EAC5` adressieren also alle dieselbe physikalische Speicherzelle.

Die Zuordnung von 1MB Adressraum für das SRAM wird vom Monitor deshalb gemacht, weil es das kitCON-167-Board von Phytec auch mit 1 MB SRAM gibt. Außerdem kann so der Adressbereich `F000` bis `FFFF` (Systembereich) auch für das SRAM genutzt werden (`0F000` ... adressiert das interne RAM bzw. die SFRs, `xF000` ... [x>>0] adressiert das SRAM).

Das Flash-Memory wird über `CS0` aktiviert. `CS0` wird immer dann LOW (aktiv) gesetzt, wenn eine Adresse nicht in einem Adressraum liegt, der durch `ADDRSEL1` bis `ADDRSEL4` festgelegt ist.

6.1.Memory-Mapping für die beschriebenen Variante 1:

Für die Variante 1 wurde durch den Befehl

```
INSTALL PHY7 0 FEA FEC BOOTSTRAP
```

der Monitor so generiert, dass er für Daten den Adressbereich `FEA00` bis `FEBFF` verwendet. Der Code selbst wird in den Adressbereich ab `FEC00` abgelegt und ist ca. 5 KB groß. Auf Grund der vorherigen Ausführungen beginnt der Code des Monitors am Starterkit-Board somit auf der physikalischen SRAM-Adresse `EC00` und der Datenbereich bei `EA00`. Ihr Anwenderprogramm darf somit in den SRAM-Bereich ab `EA00` weder Daten noch Code ablegen, da ansonsten Programmteile bzw. Daten des Monitor-Programms überschrieben werden.