

# Codegenerierung für C167

Laborversuch Automatische Codegenerierung aus Statemate mit Hilfe von Rhapsody in MicroC.

Dozent: Jochen Linkohr, Hochschule Esslingen, linkohr@hs-esslingen.de

Stand: 7.1.2011

*Der Laborversuch wurde auf Basis der beiden Praxissemester von Ruwen Kammerer und Maryam Kharazmi entwickelt.*

## Inhaltsverzeichnis

Ziel.....	3
Anwendung.....	3
Grundlagen des Entwicklungsboards.....	4
Ausgaben auf LEDs.....	4
Eingaben von den Tastern.....	4
Vorgehen.....	4
Erstellung der Applikation mit Statemate.....	5
Allgemeines.....	5
Zeitbasis des modellierten Codes.....	5
Modellierung der Applikationslogik.....	6
Statechart erstellen.....	8
Programmieren der Boardspezifischen Ein- und Ausgabe.....	9
Generierung des Codes für C167.....	10
Anpassung der generierten Dateien.....	12
Übertragung auf den C167.....	13
Board programmierbereit schalten.....	13
Flashtool starten.....	13
Serielle Kommunikation einstellen.....	14
Sektoren im Flash löschen.....	14
Binärdatei downloaden.....	15
Abschließender Reset.....	15

## Ziel

Ziel dieses Laborversuchs ist es, mit Hilfe diverser Tools eine Anwendung mit vollständig automatisch erzeugtem Code auf ein Embedded System zur Ausführung zu bringen.

## Anwendung

Die Anwendung soll auf einem Entwicklungsboard mit einem C167 Prozessor ausgeführt werden. Dabei stehen zur Eingabe eine Reihe von Tastern zur Verfügung.

Die Ausgabe erfolgt mit Hilfe von LEDs.

Die zu entwickelnde Anwendung soll beim Drücken eines Tasters auf dem Entwicklungsboard ein Lauflicht anstoßen, so dass zunächst nur eine LED eingeschaltet ist, dann zwei, dann drei, dann vier. Daraufhin soll die Anwendung einen anderen Taster abfragen und die LEDs der Reihe nach wieder ausschalten.

Der Programmablauf soll folgendermaßen aussehen:

- Zunächst sind alle LEDs aus.
- Taster A wird gedrückt
- LED 1 geht an
- 0,5s Wartezeit
- LED 2 geht zusätzlich an
- 0,5s Wartezeit
- LED 3 geht zusätzlich an
- 0,5s Wartezeit
- LED 4 geht zusätzlich an.
- Taster B wird gedrückt
- LED 4 geht aus
- 0,5s Wartezeit
- LED 3 geht aus
- 0,5s Wartezeit
- LED 2 geht aus
- 0,5s Wartezeit
- LED 1 geht aus.

Danach kann der Ablauf durch Drücken von Taster A wieder gestartet werden.

## Grundlagen des Entwicklungsboards

Das Herzstück des Entwicklungsboards ist ein C167 Prozessor. Als Betriebssystem kommt OSEK zum Einsatz, welches beim Erstellen der Anwendung auf dem Hostsystem generiert wird.

### **Ausgaben auf LEDs**

Um auf den LEDs des Entwicklungsboards eine Ausgabe zu machen, muss das entsprechende Bit in einem vordefinierten Feld gelöscht werden. Hierbei gilt es zu beachten, dass der Zustand „LED ein“ einem nicht gesetzten Bit entspricht.

**Ist das Bit gesetzt, ist die LED aus, es handelt sich also um eine negative Logik.**

Das Feld der LEDs kann innerhalb einer Subroutine mit Hilfe der Variablen P8 angesprochen werden. Ist P8 auf 0xFF gesetzt, so sind alle 16 LEDs ausgeschaltet. Um eine LED einzuschalten, empfiehlt es sich, das Feld P8 mit  $(P8 \& (\sim(1 \ll (n))))$  zu beschreiben (n = Nummer der LED). Um die erste LED einzuschalten, ist der Ausdruck  $P8 = P8 \& (\sim(0x01))$  nötig.

In C gibt es keine Möglichkeit, eine Zahl binär anzugeben, daher muss bei der späteren Programmierung die entsprechende Zahl in Hexadezimal oder dezimal eingegeben werden.

Eine LED kann mit Hilfe einer OR-Verknüpfung des entsprechenden Bitfeldes wieder ausgeschaltet werden.

Um z.B. die dritte LED auszuschalten, kann folgender Ausdruck verwendet werden:

$P8 = P8 | 0x04.$

### **Eingaben von den Tastern**

Die Taster auf dem Entwicklungsboard können mit Hilfe des Feldes P2 eingelesen werden. Auch hierbei gilt stets die negative Logik, d.h. Ist ein Bit gesetzt, ist der Taster nicht gedrückt.

Um z.B. den vierten Taster abzufragen, kann die Bitmaske folgendermaßen ausgewertet werden:

$Taster4Gedruickt = ((P2 \& (0x08)) == 0)$

Der Ausdruck ist dann Null, wenn der Taster nicht gedrückt war, sonst ungleich Null

### **Vorgehen**

Die Entwicklung des Programms soll folgendermaßen aussehen:

- 1) Modellierung der Anwendung mit Statemate.
- 2) Hinzufügen der boardspezifischen Ansteuerungsfunktionen als C-Code.
- 3) Automatische Codegenerierung.
- 4) Automatische Anpassung des generierten Codes.
- 5) Compilieren der Anwendung für das Zielsystem.

- 6) Laden der ausführbaren Anwendung auf das Zielsystem.
- 7) Starten und Validierung auf dem Zielsystem.

## **Erstellung der Applikation mit Statemate**

### ***Allgemeines***

In Statemate werden hinter die Zustände, bei denen eine Interaktion mit der Außenwelt stattfinden soll, eine Subroutine hinterlegt. In dieser Subroutine können dann die entsprechenden C-Aufrufe für die Ansteuerung bzw. das Auslesen der Hardware erfolgen.

### **Zeitbasis des modellierten Codes**

Bei der Modellierung in Statemate liegt die Zeitbasis 0.1s zugrunde. Ein timeout von 100 bedeutet also eine Wartezeit von 10s aus dem Zielsystem.

## Modellierung der Applikationslogik

Nach dem Starten von Statemate wird ein bereits vordefiniertes Projekt mit dem Namen LAB\_xx\_V3 geöffnet (xx ist die Nummer des Laborbenutzers unter dem gearbeitet wird, also z.B. 10 bei labor10).

Danach kann mit der Erstellung der einzelnen Activity- und Statecharts begonnen werden. Es können jetzt auch weitere Charts aus anderen Projekten importiert werden. Statemate benötigt auf jeden Fall ein Activitychart, welches als *Task* definiert werden muss. Dazu wird zunächst das Activitychart erstellt. Bitte beachten Sie, dass das Chart als „Regular“ angelegt wird.

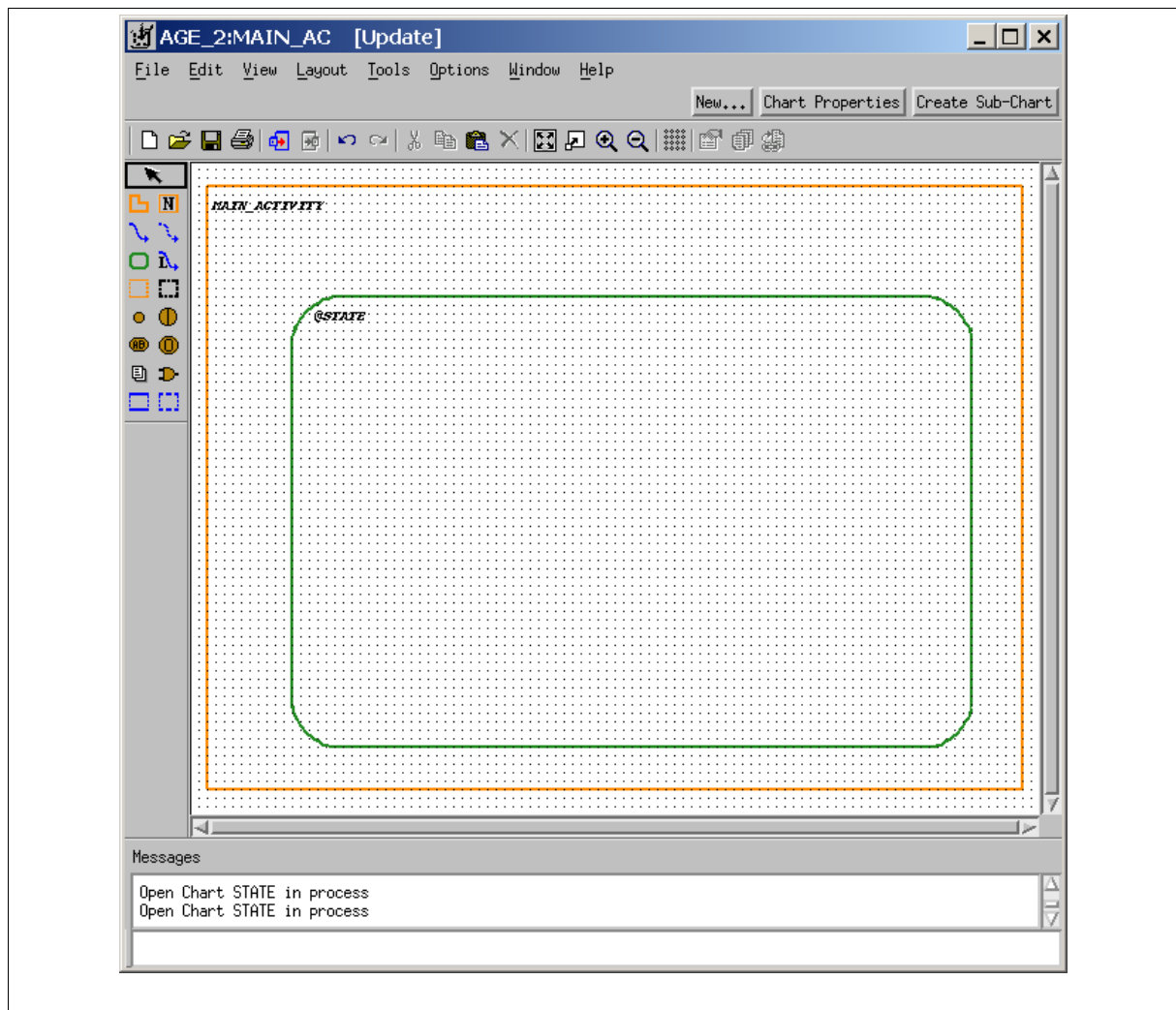


Abbildung 1: Activity mit Control Chart

Im Data Dictionary des Activitycharts unter den Design Attributes der Type *Task* ausgewählt. Die Task muss Automatisch gestartet werden (autostart)

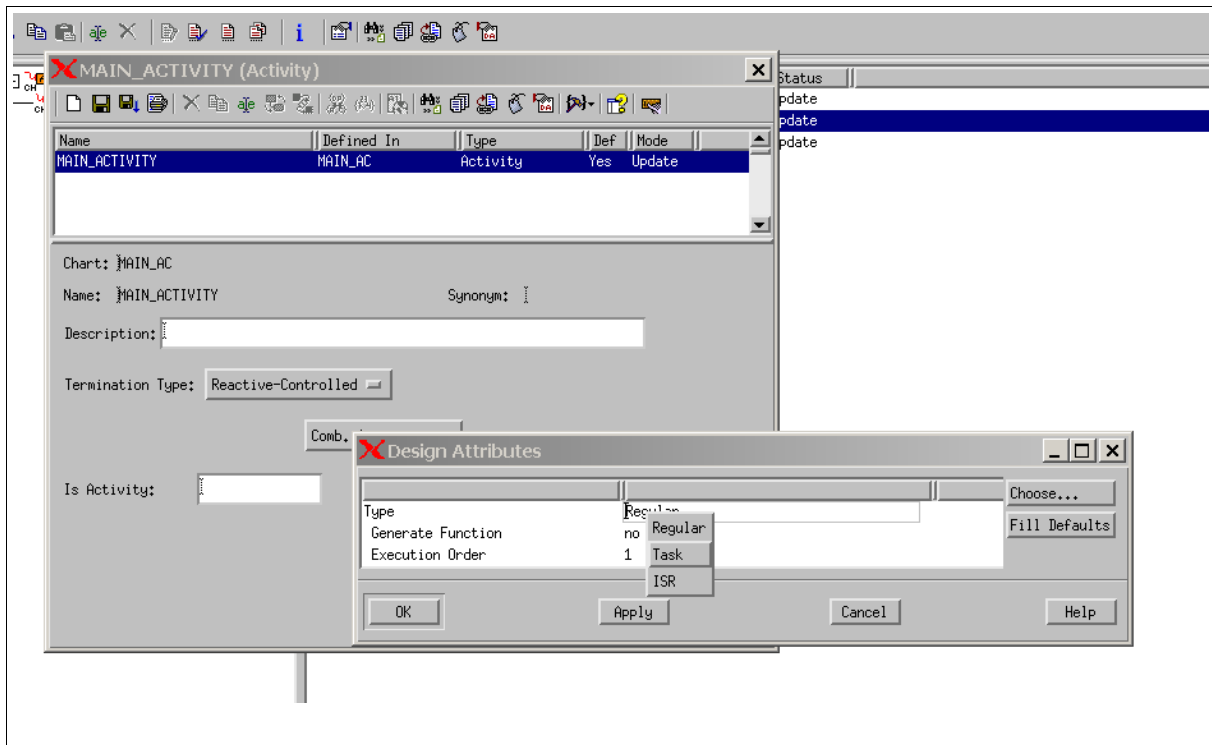


Abbildung 2: Eintragungen zur Activity im Data Dictionary

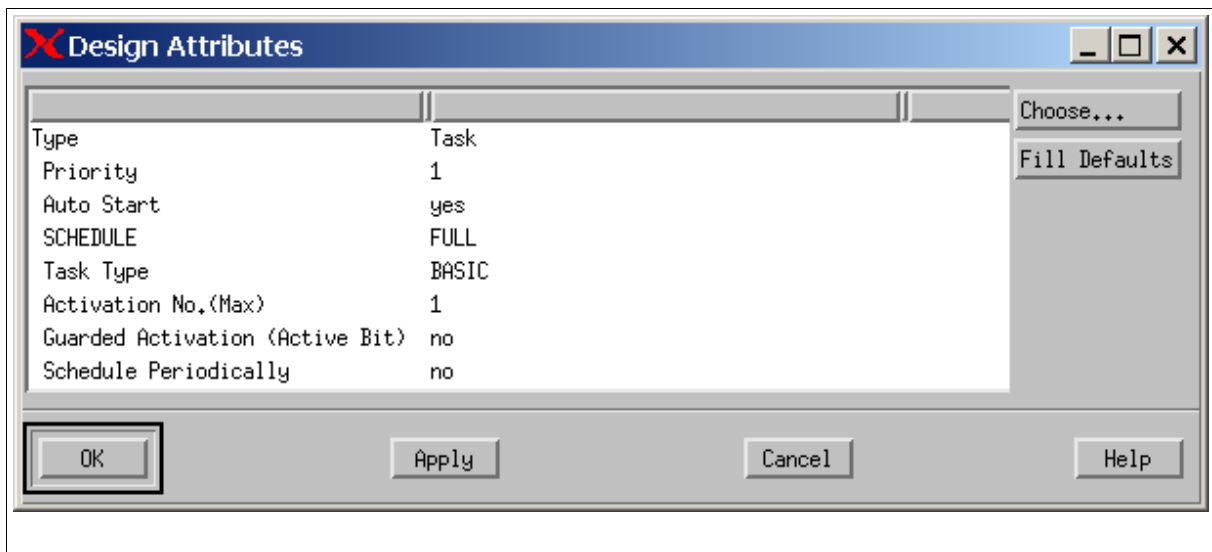


Abbildung 3: Eintragungen zu Design Attributes „Task“

## Statechart erstellen

Jetzt kann das zugehörige Statechart erstellt werden. Das im Folgenden abgebildete Chart ist nur ein Beispiel. Die hier verlangte Applikation kann deutlich einfacher modelliert werden!

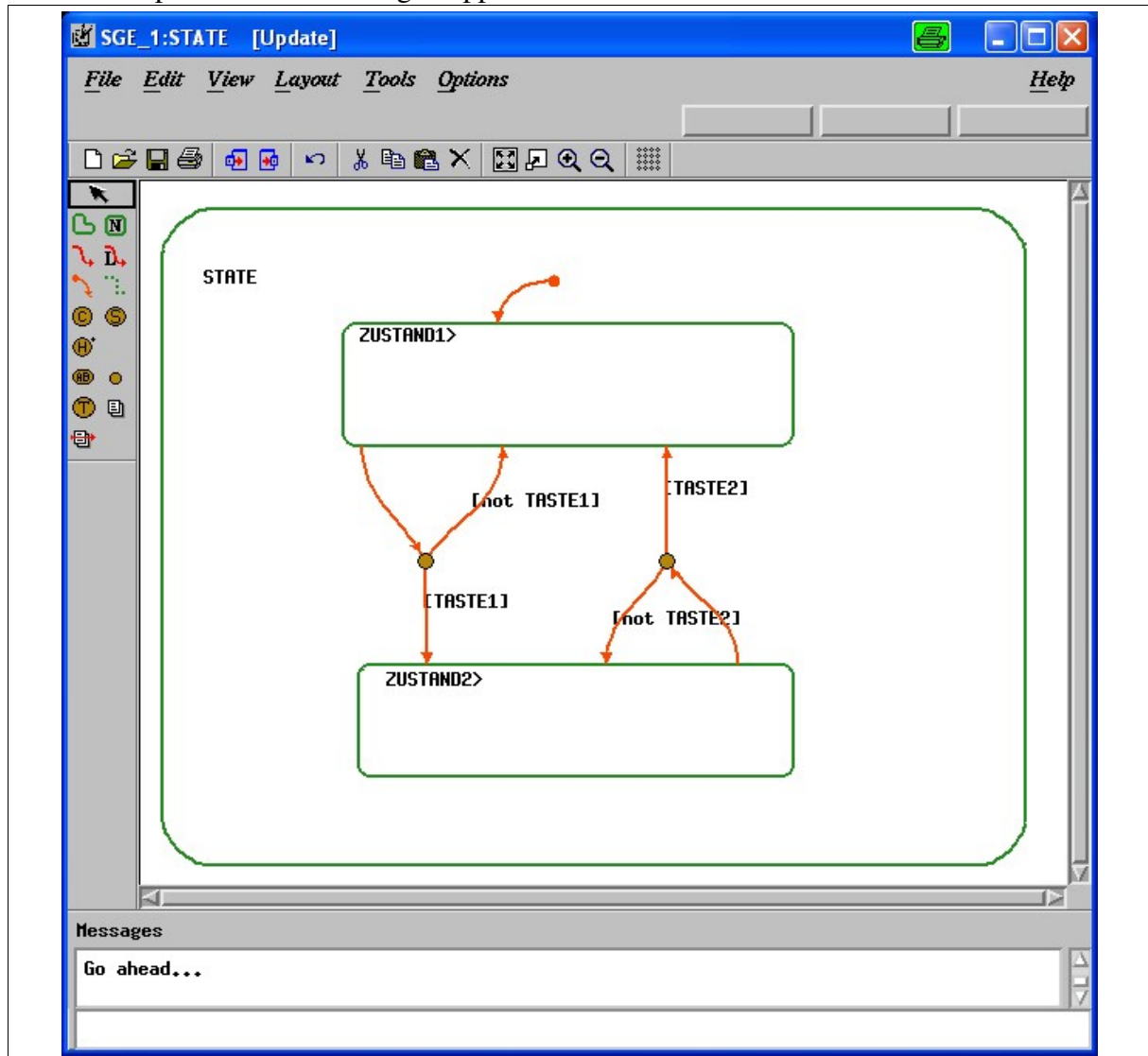


Abbildung 4: Beispiel eines Statecharts

## Programmieren der Boardspezifischen Ein- und Ausgabe

In den einzelnen Zuständen, wird nun eine Subroutine erstellt, in welcher der C-Code geschrieben wird.

Die Subroutine wird, ähnlich wie eine C-Funktion im Feld „reaction“ im Data Dictionary-Eintrag des jeweiligen Zustandes aufgerufen.

Beispiele:

```
entering/TASTER=taster_abfrage_sr();
```

oder

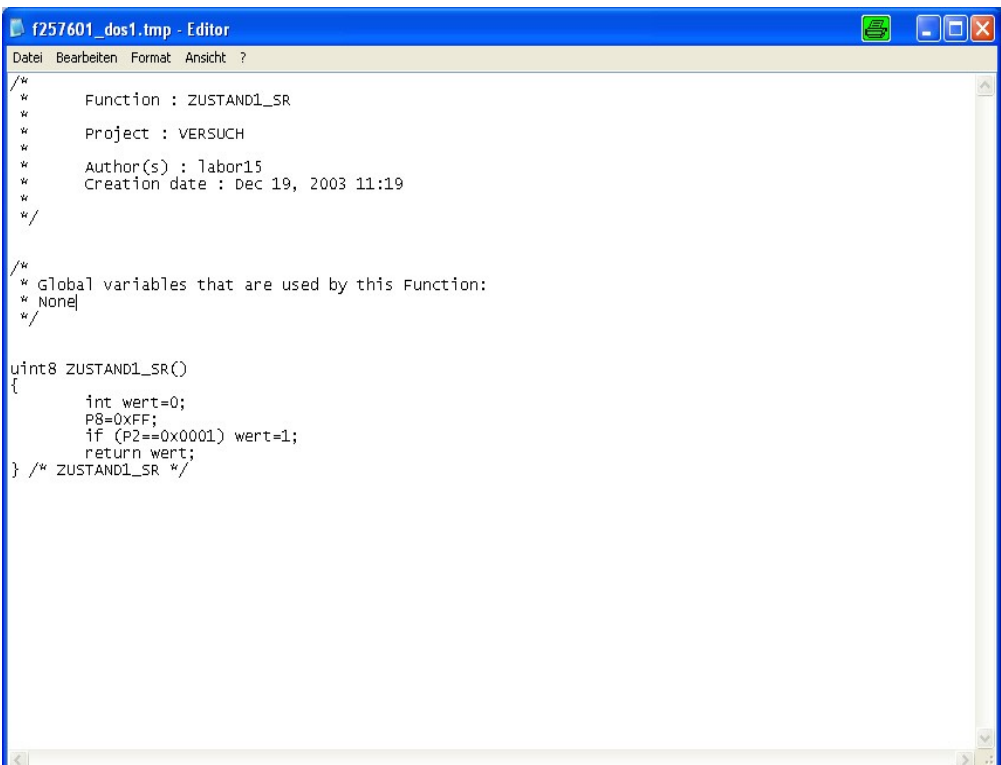
```
entering/led1_ein_sr();
```

Die Wahl der Namen der einzelnen Subroutinen ist frei.

Anschließend müssen die Subroutinen implementiert werden, indem sie im data Repository bearbeitet werden.

Dort soll zunächst der Typ (mit oder ohne Rückgabewert) definiert werden, danach folgt die Implementierung. Diese ist unter dem Menüpunkt „Implementation“ zu finden. Bei der Zielsprache „ANSI-C“ wird ein entsprechender Rumpf erzeugt, der dann mit dem entsprechenden Verhalten im Editor gefüllt werden kann.

**Der hier abgebildete C-Code ist nur ein Beispiel, im Labor verwenden Sie bitte den eigenen, richtigen Code!**



```
f257601_dos1.tmp - Editor
Datei Bearbeiten Format Ansicht ?

/*
 *   Function : ZUSTAND1_SR
 *   Project  : VERSUCH
 *   Author(s) : labor15
 *   Creation date : Dec 19, 2003 11:19
 */

/*
 * Global variables that are used by this Function:
 * None
 */

uint8 ZUSTAND1_SR()
{
    int wert=0;
    P8=0xFF;
    if (P2==0x0001) wert=1;
    return wert;
} /* ZUSTAND1_SR */
```

Abbildung 5: Beispielcode einer Subroutine, der eine Taste einliest und die LEDs ausschaltet

## Generierung des Codes für C167

Nachdem dies geschehen ist, wird nun alles gespeichert und es kann mit der Codegenerierung fortgefahren werden. Die Codegenerierung befindet sich unter Tools-> MicroC Code Generator

Dann muss ein neues Profil und ein Modul angelegt werden.

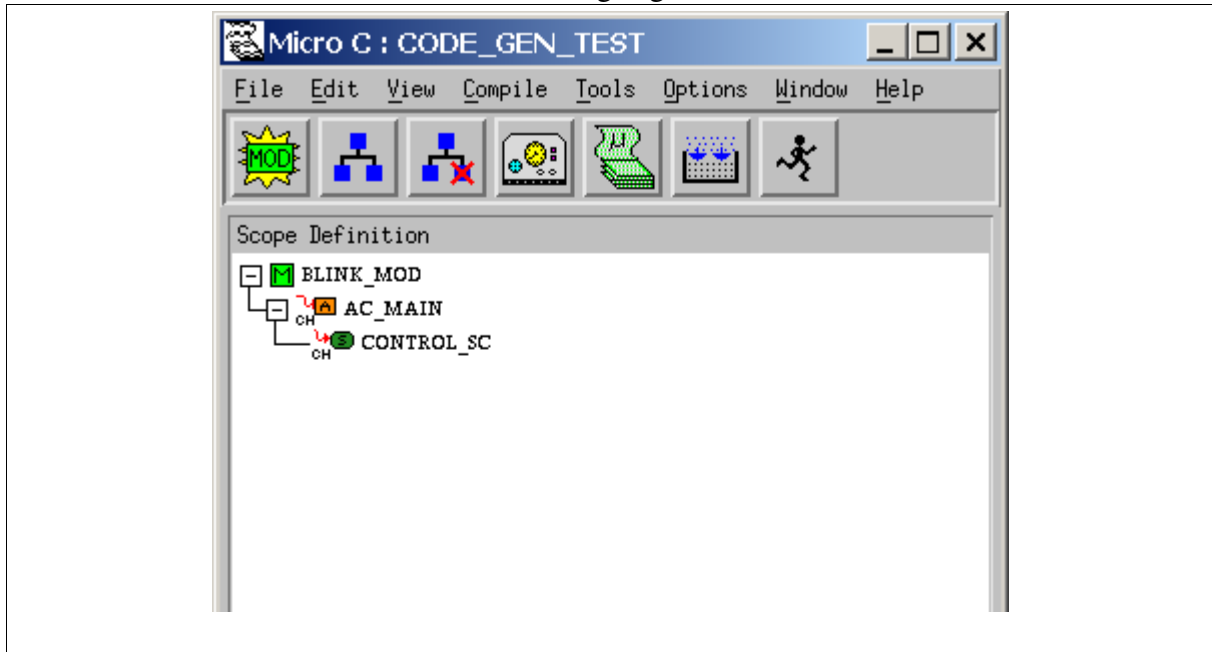


Abbildung 6: Profile mit Modul zur Codegenerierung

Unter *Options* muss nun die Option *Set Target Configuration* ausgewählt und das Ziel *C167\_OSEK* eingestellt werden.

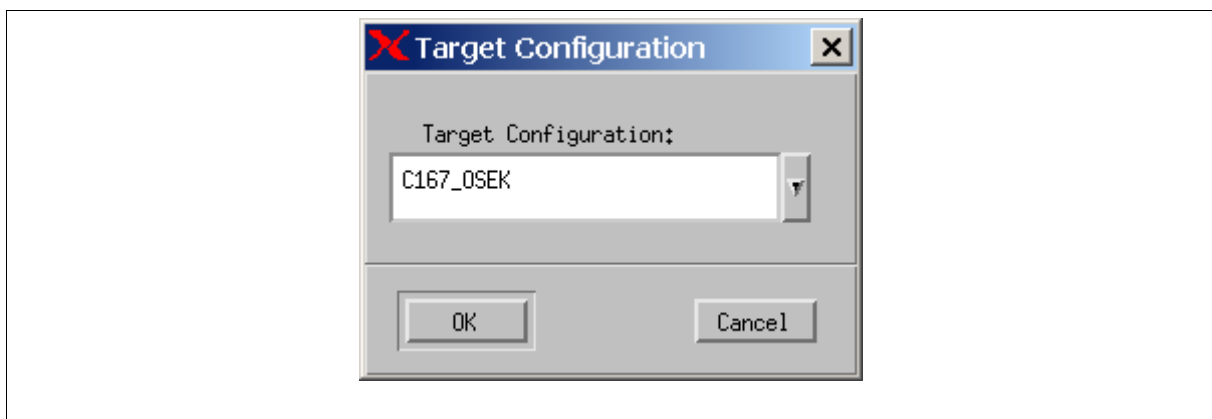


Abbildung 7: Auswahl des Zielsystems

Als nächstes muss der Name des erzeugten Moduls als Additional Object eingetragen werden. Dies geschieht unter dem Menüpunkt „Options->Settings“ im Tab „Application Configuration“ und dort mit dem Button *Application Files* (siehe Abbildung 8).

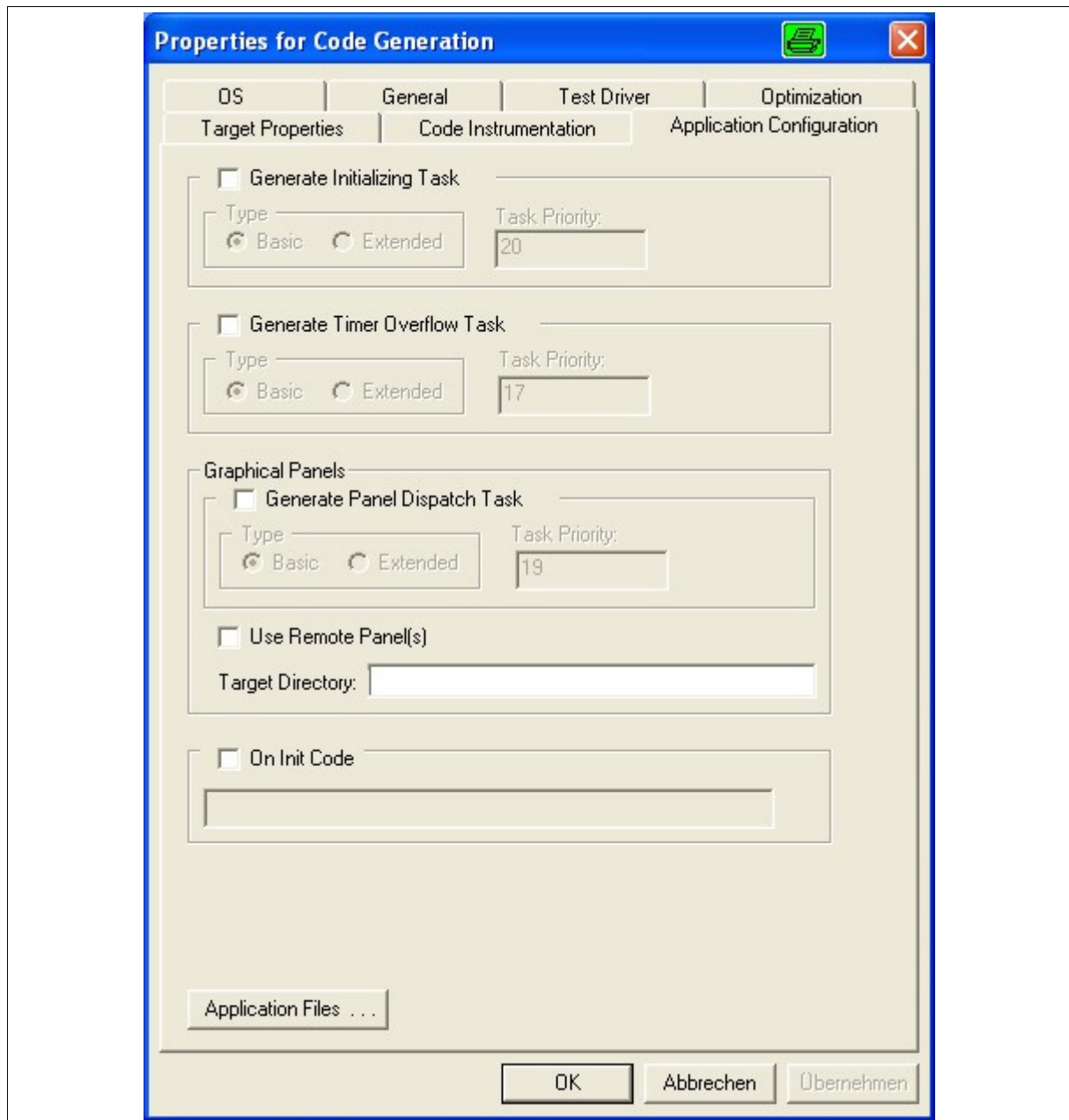


Abbildung 8: Einstellung der zusätzlichen Module (Teil 1)

Tragen Sie bei „Additional Objects“ den Namen des Moduls ein, den Sie im Codegenerierungsprofil gewählt haben (siehe Abbildung 9).

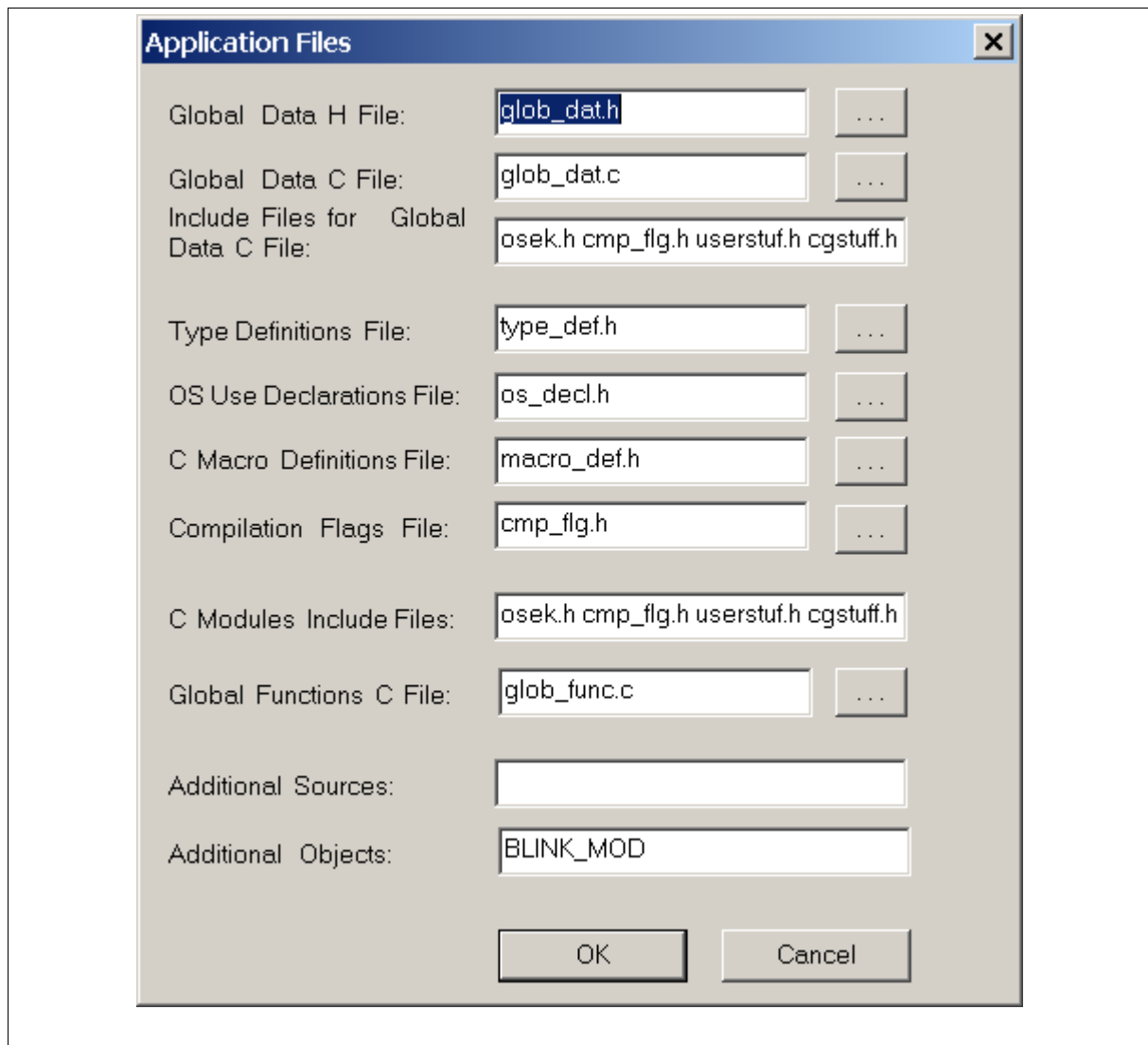


Abbildung 9: Einstellung der zusätzlichen Module (Teil 2: Modulname bei „Additional Objects“)

Jetzt kann der Code erzeugt werden (mit „**generate Code**“).

Dann kann Rhapsody in MicroC geschlossen und in den Windows-Explorer gewechselt werden.

### **Anpassung der generierten Dateien**

Mit Hilfe des Tools „Rhapsodypatcher“ kann der generierte Code an das Zielsystem angepasst werden (einige Einträge im C-Code und dem OIL-File werden geändert). Danach kann aus dem Tool mit „make“ die Übersetzung (Crosscompilierung) auf das Zielsystem durchgeführt werden.

## Übertragung auf den C167

Dieses Kapitel basiert auf dem 1. Praxissemesterbericht von Maryam Kharazmi.

### **Board programmierbereit schalten**

Der **DIP-Schalter 1** auf dem Controller-Brett muß auf **on** gestellt sein. Dann muß die **Reset-Taste** betätigt werden, damit die alten Dateien auf dem Chip gelöscht und das neue Programm geladen werden kann.

### **Flashtool starten**

Das mit Rhapsody in MicroC erstellte Binärfile wird mit Hilfe des **phytec->Flashtool** auf den Microcontroller übertragen. Beim Start öffnet sich ein Fenster, in dem bei **Supported Target Hardware KITKON->KITKON-167->Connect** ausgewählt wird.

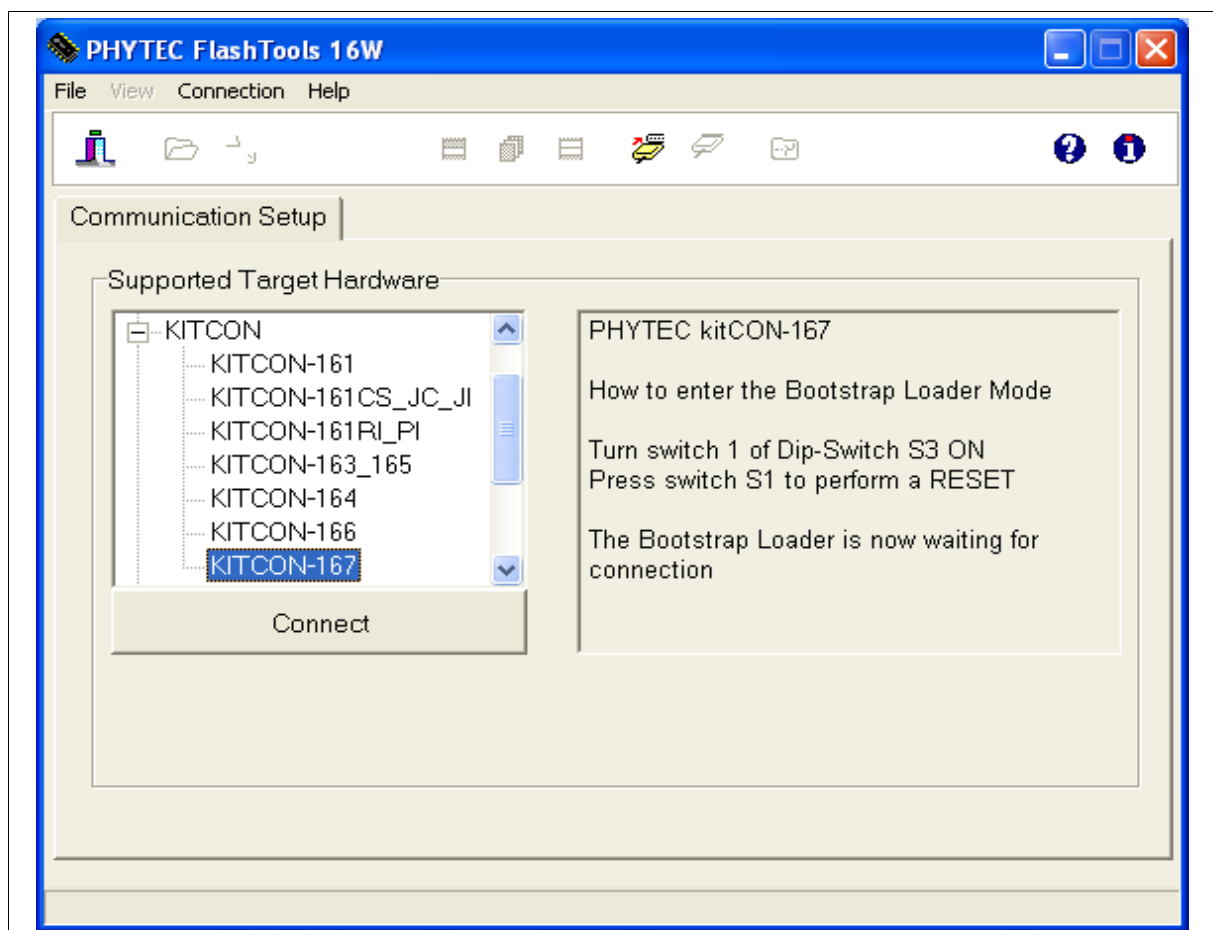


Abbildung 10: Auswahl des Zielboards im Flashtool

## Serielle Kommunikation einstellen

Anschließend muß im Fenster von **Properties for serial communication** die Schnittstelle (COM 1 oder 2) und die Baudrate (57600 Bit/s) festgelegt werden.

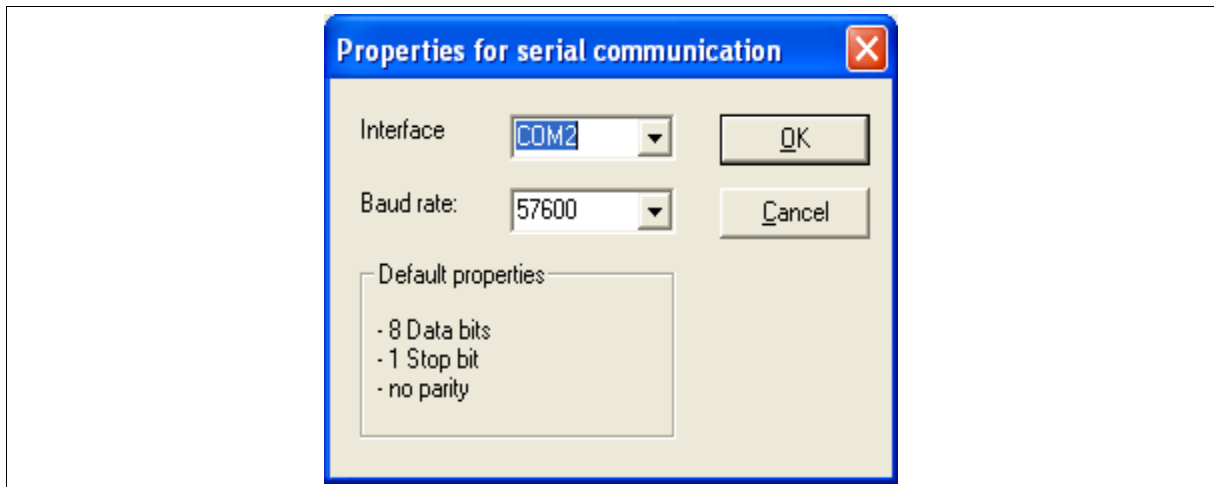


Abbildung 11: Schnittstellenparameter für die Verbindung zwischen PC und C167-Karte

## Sektoren im Flash löschen

Bei **Sector Status Information** müssen nun alle Speichersektoren gelöscht werden, d.h alle Sektoren auswählen und dann auf **Erase sector(s)** klicken.

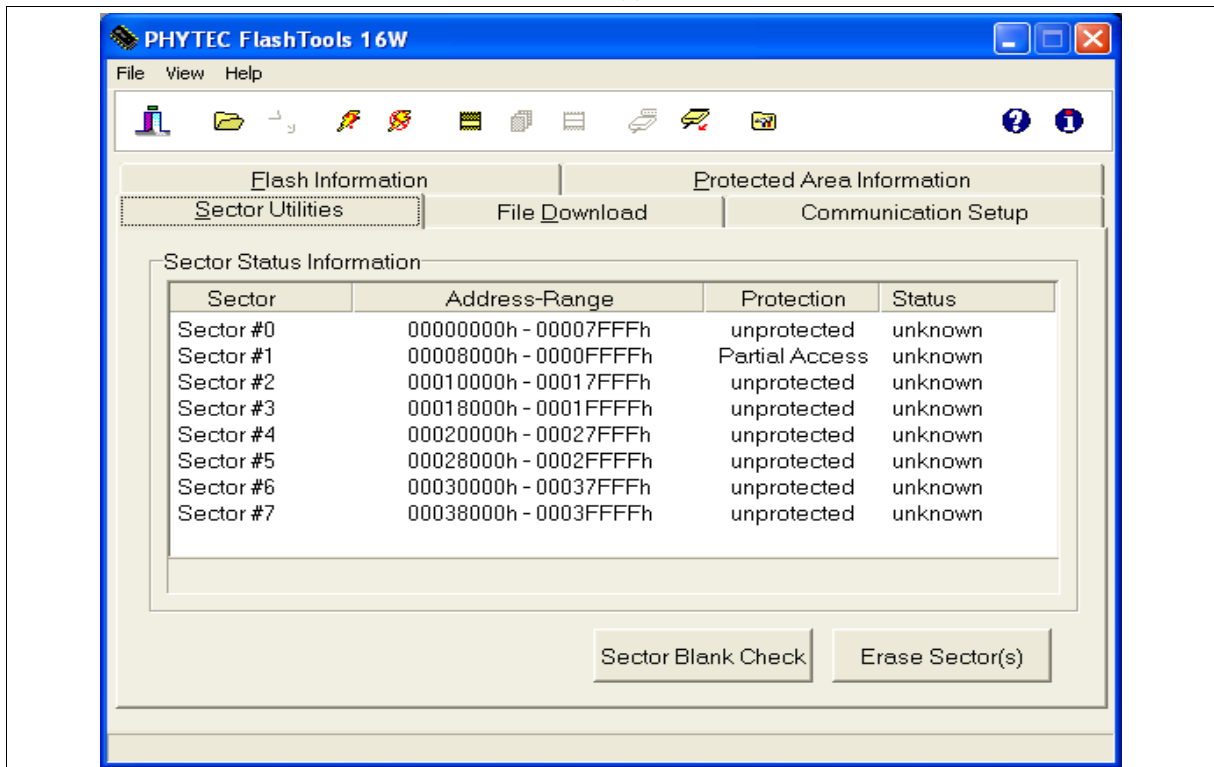


Abbildung 12: Übersicht über die Sektoren der Karte im Flashtool

## ***Binärdatei downloaden***

Dann muß der Tab „**Filedownload**“ ausgewählt werden und mit „**Fileopen**“ die Datei mit der Endung **.hex** ausgewählt werden. Der download wird durchgeführt durch **öffnen** und **download**. Die Datei t.hex befindet sich im Verzeichnis H:\rmc\wa\V3\prt\<<name des profils>.

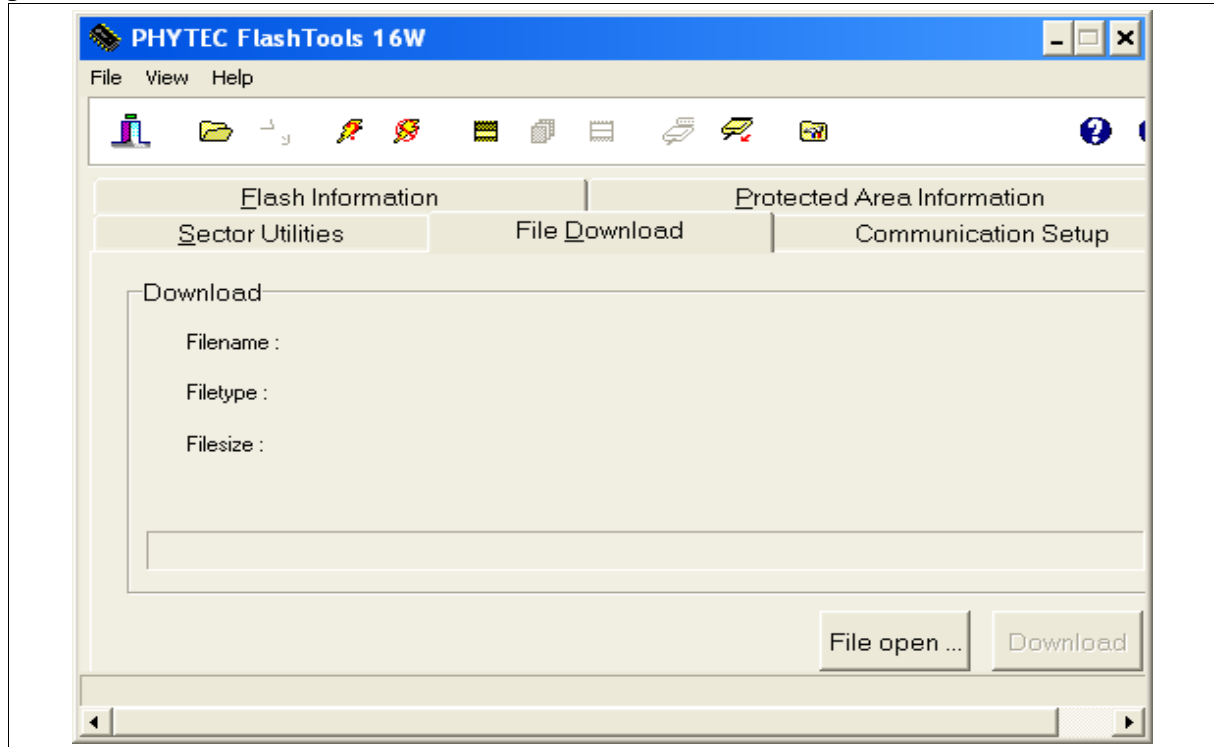


Abbildung 13: File download Dialog

## ***Abschließender Reset***

Auf dem Brett den DIP-Schalter 1 auf **aus** stellen und Reset-Taste betätigen damit das Programm ausgeführt werden kann. Das Programm startet automatisch!