Erkenntnisse zum CC5X-Compiler

(beim Umgehen der 1k-Grenze)

Autor: Buchgeher Stefan

Letzte Bearbeitung: 4. Mai 2006

Inhaltsverzeichnis

1. Wozu dieses Dokument

Der CC5X-Compiler zur Programmierung der PIC16-Mikrocontroller ist anscheinend sehr beliebt. Nicht zuletzt deshalb weil es eine kostenlose gibt. Diese kostenlose Version hat jedoch einige gravierende Nachteile. Der größte Nachteil ist, dass er nur 1k Große Programme kompelieren kann. Diesen scheinbaren Nachteil kann man aber glücklicher Weise sehr elegant umgehen. Denn es besteht ja die Möglichkeit das Programm in mehrere C-Dateien aufzuteilen. Jede diese C-Dateien kann dann unabhängig zu den anderen kompeliert werden. Mit einem Linker können dann diese einzelnen kompelierten Dateien zu einem gesamten Projekt zusammengefügt werden.

Dieses Dokument soll zeigen, wie man dabei vorgeht.

Ich setze allerdings voraus, dass der Anwender die Programmiersprache C zumindest in den Grundzügen beherrscht.

Achtung: Dieses Dokument ist **kein** Kurs über die C-Programmierung eines PIC-Mikrocontrollers, sondern zeigt nur wie man die 1k-Grenze umgeht!

2. Aufteilung des Projekts in mehrere Dateien

Wird also eine C-Datei so groß, dass es beim Kompelieren die 1k-Grenze überschreitet, so muss es in mehrere C-Dateien aufgeteilt werden, wobei auch hier gilt: Jede C-Datei darf beim kompilieren die 1k-Grenze nicht überschreiten!

Im Prinzip sieht dann das gesamte Projekt zum Beispiel folgendermassen aus:

Erste C-Datei (z.B. Demo_Teil1.C)

```
/*****************************************************************************/
/* Projekt zur Demo (Umgehung der 1k-Grenze)                                 */
/*                                                                           */
/* Entwickler: Buchgeher Stefan                                              */
/* Entwicklungsbeginn der Software: 4. Mai 2006                              */
/* Funktionsfaehig seit: 4. Mai 2006                                         */
/* Letzte Bearbeitung: 4. Mai 2006                                           */
/*****************************************************************************/

/****************** Projekt-Header einbinden *********************************/
#include "PROJEKT.H"


/****************** Include-Dateien ******************************************/
#include <int16CXX.H>               // Ist fuer die Interrupts notwendig


/****************** Konfigurations-Bits **************************************/
#pragma config |= 0b.11110100111010


/****************** ISR - Timer0 *********************************************/

/*****************************************************************************/
/* Interrupt Service Routine (ISR):                                          */
/*****************************************************************************/
#pragma origin 4

interrupt ISR(void)                 // Interruptroutine
{
       int_save_registers           // W, STATUS (und PCLATH) retten
```

```
        // Beginn der eigentlichen ISR-Routine


        int_restore_registers         // W, STATUS (und PCLATH) Wiederherstellen
}


/*********************************************************************************************/
/* INIT:                                                                                     */
/*                                                                                           */
/* Aufgabe:                                                                                  */
/*   Initialisierung des Prozessor:                                                          */
/*********************************************************************************************/
void INIT(void)
{
        // Timer-0-Interrupt
        TMR0 = 0;                     // Timer 0 auf 0 voreinstellen
        OPTION = 0b.1000.0011;        // Pull-Up-Widerstaende am Port B deaktivieren
                                      // Timer-ISR (Vorteiler = 16)

        // Ports
        TRISA = 0xFF;                 // Port A  und Port E muessen bei verwendung als
        TRISE = 0xFF;                 //   ADC als Eingang konfiguriert werden
        TRISB = 0;                    // Port B als Ausgang definieren
        TRISC = 0b.1111.1001;         // Port C: Bits 1 und 2 als Ausgang definieren
                                      // Port C: Bits 0 und 3 bis 7 als Eingang definieren
        TRISD = 0;                    // Port D als Ausgang definieren

        // ADC
        ADON = 1;                     // ADC global einschalten
        ADCON1 = 0b.1000.0010;        // Ganzer Port A als Analog; Vref+ = Vdd, Vref- = Vss
                                      //   und linksbuendig
        ADCS0 = 0;                    // Geschwindigkeit des ADC mit ADCS0 und ADCS1
        ADCS1 = 1;

        Usw.
}


/****************** Unterprogramm zur Analog-Digital-Wandlung ***************************/

/*********************************************************************************************/
/* ADC:                                                                                      */
/*                                                                                           */
/* Aufgabe:                                                                                  */
/*   Analog-Digital-Wandlung mit dem PIC-interne ADC fuer den ausgewaehlten ADC-Kanal        */
/*   starten, auf das Wandlungsergebnis warten und dieses dem aufrufenden Programm           */
/*   zurueckgeben.                                                                           */
/*                                                                                           */
/* Uebergabeparameter:                                                                       */
/*   kanal:                                                                                  */
/*                                                                                           */
/* Rueckgabeparameter:                                                                       */
/*   6-Bit-Ergebnis der Analog-Digital-Wandlung im EXTERNEN UEBERGABEREGISTER                 */
/*   (tmp_uebergabe_uns16)                                                                   */
/*                                                                                           */
/* Vorgehensweise:                                                                           */
/*   + ADC-Kanal auswaehlen (Bits <3:5> im Register ADCON0). Dazu muss die Kanal-Nummer      */
/*     an diese Position geschoben werden und mit dem Register ADCON0 verknuepft werden      */
/*   + ca. 50 Mikrosekunden (us) warten. Diese Zeit benoetigt der PIC um den internen        */
/*     Kondensator mit der zu messenden Analogspannung zu laden.                             */
/*   + Analog-Digital-Wandlung starten. Dazu das Flag GO (im Register ADCON0) setzen.        */
/*   + Warten, bis der PIC mit der Wandlung fertig ist. Der PIC setzt das Flag GO auto-      */
/*     matisch zurueck, wenn er mit der Analog-Digital-Wandlung fertig ist.                  */
/*   + Aus den Registern ADRESL und ADRESH das Ergebnis zusammensetzen und an das auf-       */
/*     rufende Unterprogramm (oder Hauptprogramm) zurueckgeben                               */
/*                                                                                           */
/* Anmerkung:                                                                                */
/*   Das Ausgabeformat der ADC-Wandlung, also wie die 10 Ergebnisbits in den Registern       */
/*   ADRESL und ADRESH abgelegt werden wird an anderer Stelle (z.B. im Unterprogramm         */
/*   INIT) konfiguriert.                                                                     */
/*********************************************************************************************/
void ADC(char kanal)
{
        ADCON0 = ADCON0 & 0b.1100.0111;
        kanal = kanal << 3;
        ADCON0 = ADCON0 | kanal;

        delay_us(50);
```

```
        GO = 1;
        while(GO);

        tmp_uebergabe_uns16.low8 = ADRESL;   // externes Uebergaberegister
        tmp_uebergabe_uns16.high8 = ADRESH;
}
```

Zweite C-Datei (z.B. Demo_Teil2.C)

```
/*****************************************************************************************/
/* Unterprogramme zur Demo (Umgehung der 1k-Grenze)                                      */
/*                                                                                       */
/* Entwickler: Buchgeher Stefan                                                          */
/* Entwicklungsbeginn der Software: 4. Mai 2006                                          */
/* Funktionsfaehig seit: 4. Mai 2006                                                     */
/* Letzte Bearbeitung: 4. Mai 2006                                                       */
/*****************************************************************************************/

/****************** Projekt-Header einbinden ***************************************/
#include "PROJEKT.H"


/****************** externe Register ***********************************************/
uns16 tmp_uebergabe_uns16;

uns16 sollwert;
uns16 istwert;


/****************** weitere Unterprogramme *****************************************/

/*****************************************************************************************/
/* Unterprogramm 2                                                                       */
/*                                                                                       */
/*****************************************************************************************/
void UNTERPROGRAMM2(void)
{
        // irgendwelche Anweisungen

}


/*****************************************************************************************/
/* Unterprogramm 3                                                                       */
/*                                                                                       */
/*****************************************************************************************/
char UNTERPROGRAMM3(char Parameter)
{
        char wert;

        // irgendwelche Anweisungen

        return (wert);
}


/****************** Unterprogramm zur Ist- und Sollwerteingabe **************************/

/*****************************************************************************************/
/* ADC:                                                                                  */
/*                                                                                       */
/* Aufgabe:                                                                              */
/*   Sollwert und Istwert von den analogen Eingaengen AN0 und AN1 einlesen und in den    */
/*   externen Registern istwert und sollwert sichern                                     */
/*****************************************************************************************/
void SOLL_ISTWERT_EINLESEN(void)
{
        // Istwert von ADC-Eingang AN0 (Kanal 0) einlesen und in der externen Variable
        //   istwert sichern
        ADC(0);
        istwert = tmp_uebergabe_uns16;

        // Sollwert von ADC-Eingang AN2 (Kanal 2) einlesen und in der externen Variable
        //   sollwert sichern
        ADC(1);
```

```
        sollwert = tmp_uebergabe_uns16;
}
```

Gemeinsame Header-Datei (z.B. PROJEKT.H)

```
/*****************************************************************************/
/* Header zur Demo (Umgehung der 1k-Grenze)                                  */
/*                                                                           */
/* Entwickler: Buchgeher Stefan                                              */
/* Entwicklungsbeginn der Software: 4. Mai 2006                              */
/* Funktionsfaehig seit: 4. Mai 2006                                         */
/* Letzte Bearbeitung: 4. Mai 2006                                           */
/*****************************************************************************/

#ifndef  __PROJEKT
#define  __PROJEKT


/****************** Pragma-Anweisungen ************************************************/
#pragma chip PIC16F877                      // PICmicro Device


/****************** Externe Register  *************************************************/
extern bank0 uns16 tmp_uebergabe_uns16;

extern bank0 uns16 sollwert;
extern bank0 uns16 istwert;


/****************** Funktionsprototypen ***********************************************/
/* Initialisierung des Mikrocontroller */
extern page0 void INIT(void);

/* Unterprogramm zur Analog-Digital-Wandlung */
extern page0 void ADC(char kanal);

/* Unterprogramm zur Ist- und Sollwerteingabe */
extern page0 void Soll_Istwerte_einlesen(void);

/* Weitere Unterprogramme */
extern page0 void UNTERPROGRAMM2(void);
extern page0 char UNTERPROGRAMM3(char Parameter)

#endif
```

3. Externe Register

Wird eine Variable (= Register) in mehreren Unterprogrammen verwendet, so muss diese Variable als externes Register definiert werden.

Dabei gilt:

1) Dieses Register darf nur in **einer** C-Datei (wo dieses Register verwendet wird) definiert werden:

   z.B.:
   ```
   uns16       sollwert;
   char        zahl1;
   ```

2) Zusätzlich muss diese externe Variable auch noch in „PROJEKT.H“ als externe Variable definiert werden:

   z.B.:
   ```
   /* Externe Register */
   extern bank0 uns16 sollwert;
   ```

```
extern bank0 char zahl1;
```

Wichtig ist hier das Schlüsselwort `extern` und die Angabe `bank0` gibt an in welcher Registerbank (bank0 bis … je nach PIC unterschiedlich) diese Variable gesichert wird.

4. Unterprogramme

Befindet sich aufgerufene Unterprogramm **nicht** in derselben Datei wie das aufrufende Unterprogramm oder aufrufende Hauptprogramm so gelten folgende Einschränkungen:

1) Es ist **nur ein** Übergabeparameter möglich
2) Dieser eine Übergabeparameter kann nur **8bit breit** sein
3) Strukturen können **nicht** übergeben werden
4) Für einen eventuellen Rückgabeparameter gelten die selben Einschränkungen

Es sind daher nur möglich:

- `void Unterprogramm (void)`
- `void Unterprogramm (char Parameter)`
- `char Unterprogramm (void)`
- `char Unterprogramm (char Parameter)`

wobei, anstelle von `char` auch `unsigned char`, `int8` oder `uns8` verwendet werden kann.

Alle anderen sind so nicht möglich
z.B.:

- `void Unterprogramm (char Parameter1, char Parameter2)`
- `void Unterprogramm (int16 Parameter)`
- `uns16 Unterprogramm (uns16 Parameter)`
- `usw.`

Ausweg:
Externe Parameter (Varaiablen) verwenden!
Bei mir lauten diese z.B.

- `tmp_uebergabe_uns16, tmp_uebergabe_uns16_1` für 16bit vorzeichenlose Zahlen
- `tmp_uebergabe_int16, tmp_uebergabe_int16_1` für 16bit Zahlen mit vorzeichen

Wichtig:
Dabei müssen natürlich die schon gemachten Hinweise für externe Register (Abschnitt 3) beachtet werden!

Weiters:
Zu jedem Unterprogramm muss es auch einen Funktionsprototyp geben. Dieser befindet sich bei mir immer in der Datei „PROJEKT.H“
Für die Funktionsprototypen gilt:

```
extern page0 void Unterprogramm1(void);
extern page0 char Unterprogramm2(char parameter);
```

Wichtig ist hier das Schlüsselwort `extern` und die Angabe `page0 gibt den` Programmspeicherbereich (page0 bis … je nach PIC unterschiedlich) an.

Anmerkung:
Das Hauptprogramm und die InterruptRoutine benötigen keinen Funktionsprototyp!

5. Interrupt

Für Interrupts gilt:

1) Datei „int16CXX.h“ einbinden (z.B. am Beginn der Datei, wo sich die ISR befindet)

```
/****************** Include-Dateien ****************************************************/
#include <int16CXX.H>                // Ist fuer die Interrupts notwendig
```

2) Die Interrupt-Routine (z.B.)

```
/****************** ISR - Timer0 *******************************************************/

/***************************************************************************************/
/* Interrupt Service Routine (ISR):                                                    */
/***************************************************************************************/
#pragma origin 4

interrupt ISR(void)                  // Interruptroutine
{
        int_save_registers           // W, STATUS (und PCLATH) retten

        // Beginn der eigentlichen ISR-Routine


        int_restore_registers        // W, STATUS (und PCLATH) Wiederherstellen
}
```

Anmerkung: ISR ist dabei ein beliebiger Name!

3) LKR-File des verwendeten PIC anpassen und im Projektordner speichern (z.B. für den PIC16F877, diese befindet sich z.B. unter C:\\Programme\Michrochip\MPASM Suite\LKR\):

Die rot markierte Stelle muss entfernt werden, und die blau markierten müssen hinzugefügt werden.

```
// Sample linker command file for 16F877
// $Id: 16f877.lkr,v 1.4.16.1 2005/11/30 15:15:29 curtiss Exp $

LIBPATH  .

CODEPAGE   NAME=vectors  START=0x0      END=0x3      PROTECTED

//CODEPAGE   NAME=page0    START=0x5      END=0x7FF
INCLUDE pid_demo.lkr
CODEPAGE   NAME=page1    START=0x800    END=0xFFF
CODEPAGE   NAME=page2    START=0x1000   END=0x17FF
CODEPAGE   NAME=page3    START=0x1800   END=0x1FFF
CODEPAGE   NAME=.idlocs  START=0x2000   END=0x2003   PROTECTED
CODEPAGE   NAME=.config  START=0x2007   END=0x2007   PROTECTED
CODEPAGE   NAME=eedata   START=0x2100   END=0x21FF   PROTECTED

DATABANK   NAME=sfr0     START=0x0      END=0x1F     PROTECTED
DATABANK   NAME=sfr1     START=0x80     END=0x9F     PROTECTED
DATABANK   NAME=sfr2     START=0x100    END=0x10F    PROTECTED
DATABANK   NAME=sfr3     START=0x180    END=0x18F    PROTECTED
```

Achtung: Dies muss bei jedem Projekt angepasst werden! (pid_demo ist hier der Projektname)

```
DATABANK   NAME=gpr0      START=0x20     END=0x6F
DATABANK   NAME=gpr1      START=0xA0     END=0xEF
DATABANK   NAME=gpr2      START=0x110    END=0x16F
DATABANK   NAME=gpr3      START=0x190    END=0x1EF

SHAREBANK  NAME=gprnobnk  START=0x70     END=0x7F
SHAREBANK  NAME=gprnobnk  START=0xF0     END=0xFF
SHAREBANK  NAME=gprnobnk  START=0x170    END=0x17F
SHAREBANK  NAME=gprnobnk  START=0x1F0    END=0x1FF

SECTION    NAME=STARTUP   ROM=vectors    // Reset vector
SECTION    NAME=ISERVER   ROM=intserv    // Interrupt routine
SECTION    NAME=PROG1     ROM=page0      // ROM code space - page0
SECTION    NAME=PROG2     ROM=page1      // ROM code space - page1
SECTION    NAME=PROG3     ROM=page2      // ROM code space - page2
SECTION    NAME=PROG4     ROM=page3      // ROM code space - page3
SECTION    NAME=IDLOCS    ROM=.idlocs    // ID locations
SECTION    NAME=CONFIG    ROM=.config
SECTION    NAME=DEEPROM   ROM=eedata     // Data EEPROM

SECTION    NAME=SHRAM     RAM=gprnobnk
SECTION    NAME=BANK0     RAM=gpr0
SECTION    NAME=BANK1     RAM=gpr1
SECTION    NAME=BANK2     RAM=gpr2
SECTION    NAME=BANK3     RAM=gpr3
```

6. Quellen

- CC5X Version 3.2 User's Manual
- www.cc5x.de