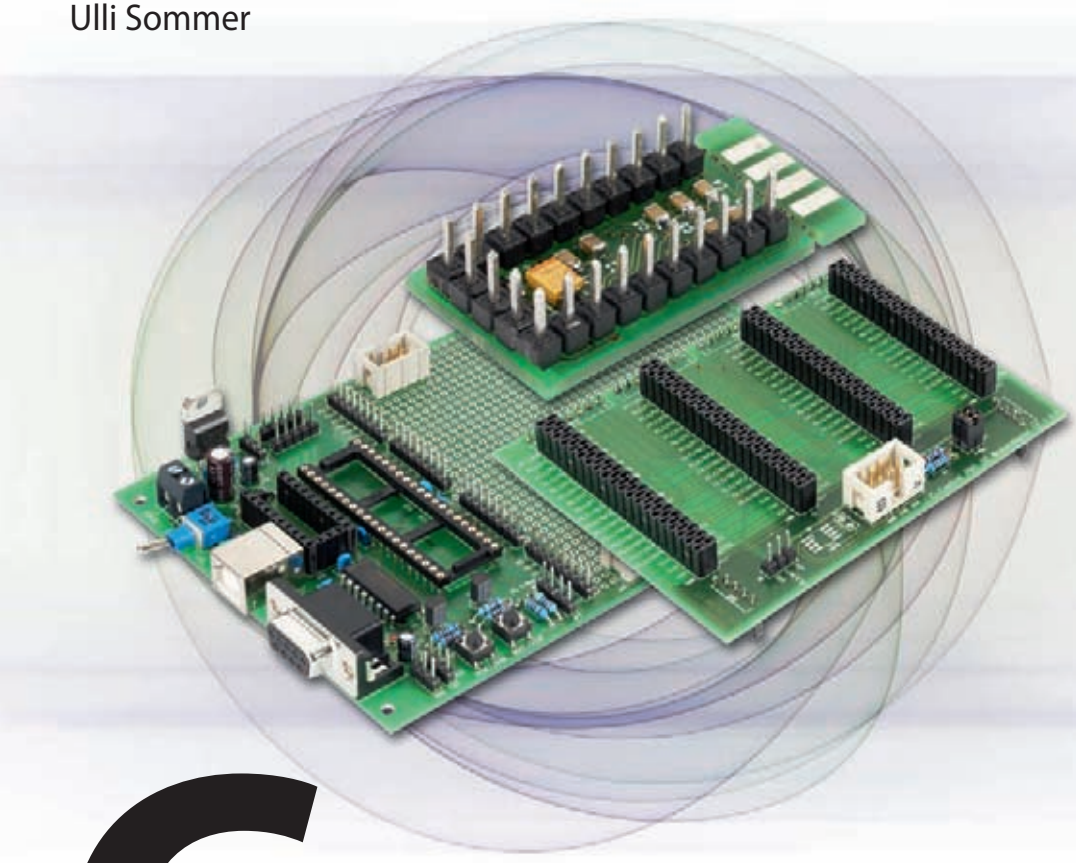
FRANZIS
PC+ELEKTRONIK

Ulli Sommer



C-Control-Pro

selbst programmieren und in der Praxis einsetzen

Das Praxisbuch für Einsteiger und Fortgeschrittene

Auf CD-ROM:

- Mikrocontroller-Grundlagen
- Aufbau und Funktionsweise
- Praktische Anwendungen

Inhalt

12 Das erste Programm

Starten Sie die C-Control-PRO-IDE, sehen Sie sofort den Programmeditor vor sich. Gehen Sie in der oberen Leiste auf *Projekt* und wählen Sie *Neu*. Legen Sie einen Ordner mit den Namen Ihrer Wahl an. Nun haben Sie ein neues Projekt erstellt. Um nun einen Quellcode schreiben zu können, müssen Sie unter *Projekt* noch *Datei neu hinzufügen* auswählen. Auch hier können Sie einen Namen Ihrer Wahl vergeben und zwischen Basic, C-Compact und ASM wählen. Nach dem Erstellen sehen Sie vor sich ein weißes Editorfenster, in das Sie Ihr Programm schreiben können.

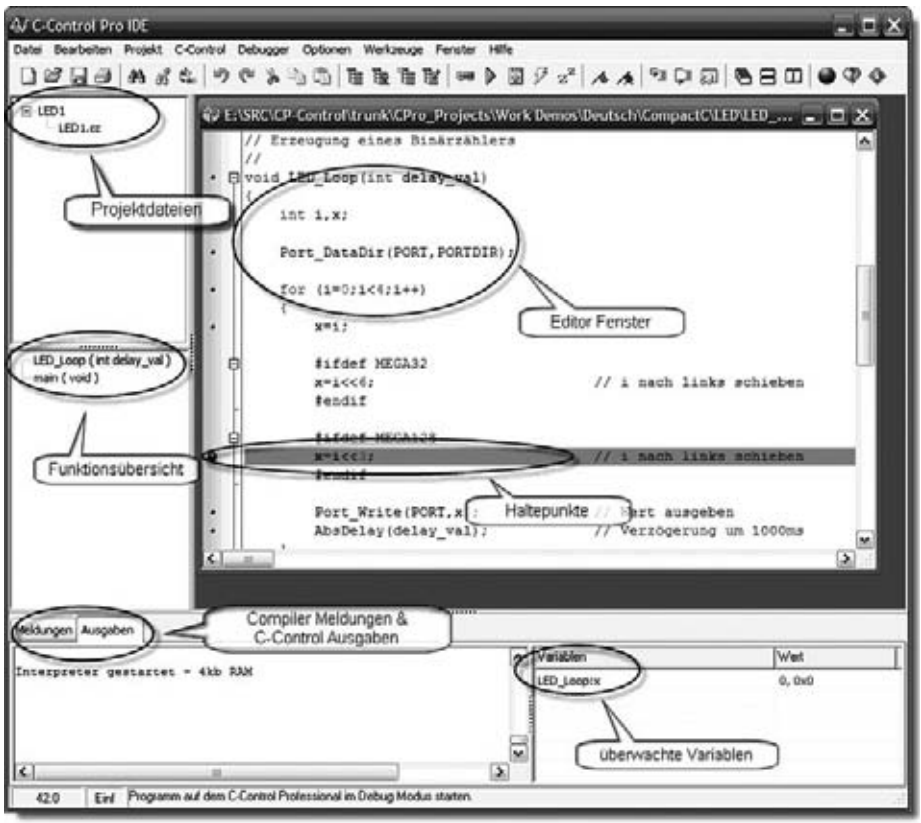


Abb. 12.1: Die IDE-Übersicht und ihre Funktionen

12.1 Die Programmstruktur

Der Aufbau des Programms sollte immer folgendermaßen aussehen:

1. Infotexte und Programmbeschreibung (Header)
2. Präprozessoranweisungen #define usw.
3. Variable anlegen (Dim x as Byte)
4. Sub main()
5. Eigene Funktionen und Unterprogramme

In der Routine Sub main(), die immer einmal vorhanden sein muss, werden, falls erforderlich, zuerst die Variablen auf einen gewünschten Startwert gesetzt. Danach werden die Ports konfiguriert und in den gewünschten Startzustand gesetzt. Das kann auch über eine eigens dafür geschriebene Initialisierungsroutine geschehen. Z. B. kann dieses System_Init() benannt werden. Es hat verschiedene Vorteile, die Routinen beim Namen zu nennen. Wird das Programm etwas größer, kann man mit undurchdachten Namen leicht ins Schleudern geraten. Danach beginnt das eigentliche Hauptprogramm, das meist in einer Endlosschleife läuft.

```
'*********************************************
'Das ist das erste Programm mit der C-Control PRO
'*********************************************

#define PI 3.14

Dim Ich_bin_eine_Variable als Byte

'Hauptprogramm
Sub main()

Port_DataDirBit(Portbit,Funktion)
Port_WriteBit(Portbit,Zustand)

Do
      'Hier steht das eigentliche Programm

Loop

End Sub

'Funktionen und Unterroutinen
Sub Testfunktion(a As Byte, b As Byte) As Word
      Dim x As Word
      x = a+b
      Return x
End Sub
```

12.2 Das Programm

Zuerst lassen wir eine LED blinken. Dazu schließen Sie eine Low-current-LED (If = 2 mA) über einen 1,5-KΩ-Widerstand an einen I/O-Port des Controllers an. Benutzen Sie einen freien Port, also keinen, der durch die SPI- oder LCD-Leitung bereits verwendet wird. Das andere Ende der Leuchtdiode, die *Katode,* schließen Sie an Masse (GND) an.

Achtung! Bevor Sie die LED anschließen, trennen Sie das Board vom PC und der Stromversorgung.

Um ein neues Programm zu schreiben, erstellen Sie einen beliebigen Ordner, der Ihre C-Control-PRO-Programme enthalten wird, sowie einen neuen Unterordner mit dem Namen *Blink*.

Nun müssen Sie in der IDE ein neues Projekt anlegen. Dazu gehen Sie mit der Maus auf den Menüpunkt *Projekt\Neu*. In das nun eingeblendete Dialogfeld geben Sie den Namen des Projekts ein. Jetzt können Sie den Speicherort des Programms festlegen und legen die Datei in den Ordner *Blink*.

Das Projekt ist nun angelegt. Jetzt fügen Sie noch ein neues Programmfenster hinzu, in dem Sie den Code eintippen. Dazu gehen Sie wieder auf *Projekt* und wählen *Datei neu hinzufügen*. Auch diesem Formular geben Sie einen Namen – er kann ebenfalls *Blink* lauten. Zur Auswahl steht noch, um welches File es sich handeln soll. Soll der Code in C (*.cc) oder Basic (*.cbas) geschrieben werden? In unserem Fall handelt es sich der Einfachheit halber um Basic-Code – also müssen Sie als Dateiendung *Blink.cbas* wählen. Jetzt erscheint im Editor ein neues, leeres Formular, in das Sie den Code eingeben können. Zum Schluss speichern Sie das Programm ab.

Bei unserem Beispiel ist die LED an Portbit 30 (PortD.6) angeschlossen. Dazu muss man diesen Port als Ausgang konfigurieren.

```
Port_DataDirBit(30,PORT_OUT)
```

Verwenden Sie hier PortD.6 = Portbit 30 und bestimmen Sie mit *PORT_OUT,* dass er ein Ausgang ist. Mit dem Präprozessor-#define-Befehl können Sie dem Portbit auch einen Namen geben, z. B. *LED1*. Das würde dann so aussehen:

```
#define LED1 30
```

30 gibt das Portbit an, an dem die LED angeschlossen ist.

Nun soll die LED blinken. Dazu benötigt man noch etwas Programmcode, denn bis jetzt wurde nur der I/O-Port konfiguriert.

```
Sub main()

      Port_DataDirBit(LED1,PORT_OUT)

      Port_WriteBit(LED1,PORT_ON)
      AbsDelay(1000)
      Port_WriteBit(LED1,PORT_OFF)
      AbsDelay(1000)
End Sub
```

Hier wird die LED einmal für 1 Sekunde ein- und dann wieder ausgeschaltet, nach einer weiteren Sekunde ist das Programm beendet.

Da die LED jedoch dauerhaft nach dem Programmstart blinken soll, muss man den Code noch in eine Endlosschleife einbauen. Mit dem folgenden Code blinkt die LED im Sekundentakt. Die Wartezeit für *Ein* oder *Aus* wird mit *AbsDelay* festgelegt. Hier wird der Interpreter komplett angehalten und wartet die angegebene Zeit ab. Die Zeitangabe erfolgt in ms (Millisekunden).

```
Sub main()

      Port_DataDirBit(LED1,PORT_OUT)

  Do While True
      Port_WriteBit(LED1,PORT_ON)
      AbsDelay(1000)
      Port_WriteBit(LED1,PORT_OFF)
      AbsDelay(1000)
  End While

End Sub
```

Das erste Programm ist erfolgreich geschrieben.

12.3 Programm auf die C-Control PRO übertragen und starten

Jetzt wollen Sie natürlich Resultate sehen und auch, ob das Programm auf den Controller wirklich läuft. Zu diesem Zweck müssen Sie es noch auf die C-Control PRO übertragen.

Dazu muss, falls es noch nicht geschehen ist, die Programmierschnittstelle eingestellt werden. Benutzen Sie das Applicationboard über USB und wählen Sie unter *Optionen\IDE\Schnittstellen USB0* aus. Verwenden Sie die RS232 (Projectboard), müssen Sie die verwendete COM-Schnittstelle auswählen.

Jetzt ist der Programmer eingestellt. Nun muss man das Programm noch kompilieren, damit der Controller es auch versteht. Dazu klicken Sie auf den kleinen blauen Pfeil unterhalb der Menüleiste. Alternativ können Sie auch F9 auf der PC-Tastatur drükken.

Wenn Sie sich nicht vertippt haben, werden keine Fehlermeldungen ausgegeben und das Programm wurde in einen C-Control-PRO-verständlichen Code übersetzt. Sollte die IDE doch einen Fehler melden, müssen Sie den Code auf Tippfehler überprüfen.

Nach dem Kompilieren können Sie das Programm auf den Controller übertragen. Dazu klicken Sie auf den kleinen grünen Pfeil nach oben unterhalb der Menüleiste (oder Shift+F9). Jetzt erscheint ein kleines Downloadfenster mit Fortschrittsbalken. Ist die Übertragung beendet, starten Sie das Programm mit dem kleinen gelben Blitz (oder F10). Die LED sollte nun wie oben beschrieben blinken.

13 Der Einstieg in die Programmierung

Für die Anwender, die noch keine große Erfahrung mit einer Programmiersprache gemacht haben, sondern sich immer wieder von den komplexen Strukturen von Assembler oder C abschrecken ließen, folgt hier eine kleine Einführung in das C-Control-PRO-Basic.

13.1 Die Variablen

Jedes Programm besteht aus verschiedenen Variablen, die entweder von der Außenwelt (wie einem ADC oder Port) stammen oder intern zur Verrechnung im Inneren des Programmes benötigen werden, um daraus wieder eine Ausgabe über ein LCD oder einen Port, RS232 etc. zu machen. Für die Programmierung stehen verschiedene Variablen-Typen wie Byte, Integer usw. (siehe Datentypen) zur Verfügung. Diese müssen vor der Verwendung immer definiert werden. Das kann als Konstante oder als Variable geschehen.

```
#define PI 3.14              'Konstante PI

Dim Var As Byte              'Byte-Variable, sie kann Zahlen von
                             'O bis 255 aufnehmen

Dim Var(10) As Char          'Char Array, hat eine ähnliche Bedeutung
                             'wie 11x ein Char,mit Dim Var As Char
                             'zu erstellen. Das jeweilige Char,wird
                             'über den Index angesprochen: Var( x)
```

Eine Variable in Basic kann eine Hex-, Binär oder Dezimalzahl oder, bei einem Char, ein ASCII-Zeichen sein.

13.2 Kontrollstrukturen und Bedingungen

Jedes Programm benötigt Bedingungen, um auf Ereignisse zu reagieren. Diese werden meist aus *If, Then, Else, End* und *Else If* angegeben.

```
If [Variable A] = [Variable B] Then
      'Hier steht normalerweise der Code,
      'der bei der Bedingung ausgeführt werden soll
End If
```

Der Code wird also nur ausgeführt, wenn A „gleich" B ist. Welche Operatoren erlaubt sind, können Sie in Kapitel 42 nachlesen.

```
If [Variable A] > [Variable B] Then
        'Code, der ausgeführt werden soll
  Else  'Oder wenn A nicht größer B
        'Code, der ausgeführt werden soll
End If
```

Man sieht: Mit der Else-Anweisung kann man eine Alternative anbieten.

```
If [Variable A] <> [Variable B] Then
      'Code, der ausgeführt werden soll
   ElseIf [Variable A] < [Variable B] Then
      'Code, der ausgeführt werden soll
   ElseIf [Variable A] > [Variable B] Then
      'Code, der ausgeführt werden soll
End If
```

Mit der ElseIf-Anwesiung kann man Verschachtelungen einrichten.

Eine Alternative zur If- und ElseIf-Anweisung wäre die *Select-Case-Anweisung*.

```
Select Case [Variable]

      Case > 10
      'Code, der ausgeführt werden soll
      Case > 100
      'Code, der ausgeführt werden soll
   Else
      'Alternativ-Code, wenn alle anderen Bedingungen nicht zutreffen
End Case
```

Eine noch ganz praktische, aber mittlerweile veraltete Anweisung ist *Goto*. Dieser Befehl stammt aus Zeiten, als Basic-Programme noch mit Zeilennummern und Sprüngen geschrieben wurden. Es gibt zwar in Basic noch die Möglichkeit, Zeilen anzuzeigen, aber sie dienen eher zur Orientierung. Man muss als Sprungmarke ein Label, kurz: *Lab,* angeben.

```
Lab Start           'Label anstatt Zeilennummer
      'Code, der ausgeführt werde soll
      Goto Start    'Sprung zum Label-Start
```

Labels kann man sehr gut zum Abbrechen oder zur Fehlerbehandlung benutzen.

Bei der Programmierung werden häufig Programmschleifen benötigt, z. B. für Dezimal- oder Binärzähler.

```
Dim X As Integer    'Variable anlegen
For X = 1 To 10     'Diese Schleife zählt von 1 bis 10
                    'mit einer Schrittweite von 1
                    'Code, der 10x durchlaufen werden soll
Next

For X = 1 To 10 Step 2     'Die Variable X wird jetzt immer um
                           '2 erhöht
      'Code
Next

For X = 10 To 1 Step -1    'Jetzt wird von 10 auf 1 heruntergе-
                           'zählt (Schrittweite 1)
      'Code
Next
```

Mit einer For-Next-Schleife kann man eine definierte Anzahl von Durchgängen einrichten.

Eine weitere Variante einer Schleife ist die *Do-While-Loop-Version*. Diese Version wird nur durchlaufen, wenn die Startbedingung erfüllt ist.

```
Do While [Variable] < 20          'Schleife beginnt, wenn
                                  'Variable <20 ist

      Variable = Variable + 1     'Variable wird immer um 1
                                  'erhöht, ist diese >20
                                  'wird die Schleife verlassen
End While

Do While True                     'Endlosschleife
      'Was auch immer wir hier tun
End While

Do While True                     'Endlosschleife
       Variable = Variable + 1    'Variable bei jedem Durchlauf
                                  'um 1 erhöhen
       If Variable > 10 Then      'Ist die Variable >10, wird mit
                                  'Exit die Schleife verlassen
         Exit                     'Exit, wie der Name schon sagt
      End If
End While
```

Endlosschleifen werden sehr oft bei Multithreading benötigt. Jeder Thread benötigt eine eigene Endlosschleife. Auch das Hauptprogramm läuft meist in solch einer Schleife, da sonst nach einem Durchlauf das Programm beendet werden würde.

Wenn wir eine Schleife mit Bedingung benötigen, die mindestens ein Mal durchlaufen werden muss, ist *Do Loop While* ideal.

```
Do
   Variable = Variable + 1
Loop While Variable < 10
```

Auch diese Version kann mit einer *Exit-Anweisung* vorzeitig beendet werden.

13.3 Funktionen und Routinen

Funktionen werden Sie von Anfang an benötigen. Die IDE ist auf fertige Funktionen gestützt und auch Sie können eigene Funktionen, die Sie immer wieder benötigen, selbst schreiben. Eine Funktion ist nicht anderes als eine Unterroutine, die einen bestimmten Algorithmus ausführt.

```
Sub IchBinEineFunktion(Variable As Byte)  As Byte
      Return (Variable * 10)
End Sub
```

Innerhalb der Klammer stehen die Übergabeparameter, nach der Klammer wird definiert, als welche Type die Rückgabe geschehen soll. Es ist nicht zwingend erforderlich, eine Rückgabevariable mit anzugeben. Es kann ja sein, dass Sie nur Werte in die Funktion übergeben wollen, um z. B. eine neuen PWM-Wert einzustellen.

Achtung! Wenn Sie jedoch Char Arrays übergeben wollen, wie es in der LCD-Routine der Fall ist, müssen sie mit *ByRef* als Referenz übergeben werden.

```
Sub Text(ByRef Variable As Char)
      'Code
End Sub
```

Ein anderer Fall wäre eine reine Routine ohne Übergabe oder Rückgabeparameter. Um z. B. eine LED einzuschalten, muss man immer eine Unzahl von Parametern übergeben. Wäre es nicht schön, nur *LED1_ON()* oder *LED1_OFF()* zu schreiben und die LED geht in den gewünschten Zustand? Dies kann man ganz einfach mit einer Routine lösen.

```
Sub LED1_ON()
      Port_DataDirBit(8,PORT_OUT)
      Port_WriteBit(8,PORT_ON)
End Sub
```

So könnte die Routine für *LED einschalten* aussehen.

29 Temperaturmessung mit DS18S20

Abb. 29.1: Der DS18S20 im Gehäuse, Conrad-Best.-Nr. 198284 (DS18S20 mit Kabel)

Abb. 29.2: Der Fühler für Printmontage, Conrad-Best.-Nr. 176168 (DS18S20 Printmontage)

Nach der „1-Wire-Bus-Theorie“ kann man nun eine Temperaturanzeige mit den DS18S20-1-Wire-Sensor von Maxim programmieren. Dazu muss man nur den Daten-Pin (DQ) mit einem freien I/O-Pin der C-Control PRO verbinden und für den Test die Stromversorgung am Sensor anlegen.

29.1 Die Pinbelegung des Kabelsensors

Aderfarbe	Funktion
Grün	Vdd +5 Volt
Weiß	Data in/out
Braun	GND Masse

29.2 Die Pinbelegung des Sensors für die Printmontage

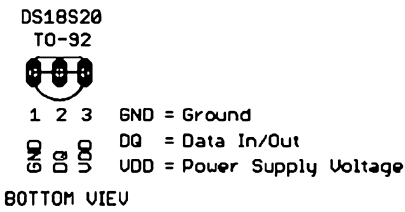


Abb. 29.3: Pinbelegung

Beispiel:

```
'DS18S20-1-Wire-Temperatur-Sensor lesen
Dim Text(40) As Char
Dim ret, i As Integer
Dim temp As Integer
Dim rom_code(8) As Byte
Dim scratch_pad(9) As Byte

Sub main()

    ret = OneWire_Reset(7)        'PortA.7

    If ret = 0 Then
       Text= "Keinen Sensor gefunden"
       Msg_WriteText(Text)
       GoTo Ende
    End If

    OneWire_Write(0xcc)           'ROM-überspringen-Kommando
    OneWire_Write(0x44)           'starte Temperaturmessung-Kommando
    AbsDelay(1000)
    OneWire_Reset(7)              'PortA.7
    OneWire_Write(0xcc)           'ROM-überspringen-Kommando
    OneWire_Write(0xbe)           'lese scratch_pad-Kommando

    For i = 0 To 9                'komplettes Scratchpad lesen
        scratch_pad(i)= OneWire_Read()
        Msg_WriteHex(scratch_pad(i))
    Next

    Msg_WriteChar(13)
    Text = „Temperatur: „
    Msg_WriteText(Text)
    temp = scratch_pad(1) * 256 + scratch_pad(0)
    Msg_WriteFloat(temp * 0.5)
    Msg_WriteChar(99)
    Msg_WriteChar(13)

    Lab Ende

End Sub
```

30 Der I²C-Bus und wie er funktioniert

(Philipps)

Der I²C-Bus (Inter Integrated Circuit) ist ein serieller synchroner Zweidraht-Bus, der in den 80er-Jahren von Philips für die interne Verbindung zwischen Baugruppen und ICs entwickelt worden ist. Trotz seines hohen Alters hat er bis heute nicht an Bedeutung verloren. Ganz im Gegenteil: Er wird sogar gern überall dort eingesetzt, wo an Leiterbahnen und Verdrahtung gespart werden muss – sei es aus Kostengründen oder aus Platzmangel. Auch die Sensoren und Port-Erweiterungen wie der PCF8574P und seine Artgenossen sind ideal, um mehrere I/O-Ports mit wenig Schaltungsaufwand zu gewinnen. Mithilfe neuer Expansions- und Steuerungsbauelemente kann der I²C-Bus inzwischen über die 400-pF-Grenze (ca. 20 bis 30 ICs pro Bus-Segment) hinaus erweitert werden. Dadurch können Entwickler mehrere Chips und sogar mehrere identische ICs mit derselben Adresse anschließen und flexibel auf die steigende Zahl von I²C-Bauelementen reagieren.

Ein großer Vorteil des I²C-Busses ist auch die einfache Ansteuerung. Da keine festen Taktzeiten eingehalten werden müssen, können sowohl langsame als auch sehr schnelle Busteilnehmer, Chips und Programmiersprachen eingesetzt werden.

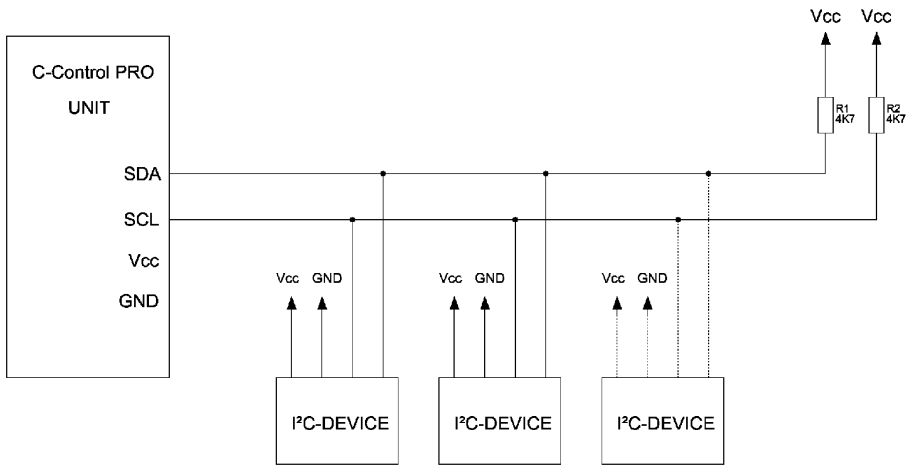


Abb. 30.1: Busstruktur des I²C-Busses

Man sieht, dass der Bus immer auf +5 Volt liegen muss. Dies wird durch R1 und R2 gewährleistet. Die Widerstände können zwischen 1K und 10K liegen. Jeder dieser Bausteine besitzt eine Adresse, um ihn zu identifizieren. Die Adresse ist meist bis auf 3 Bit vom Hersteller fest vorgegeben. Das letzte Bit der Adresse gibt an, ob der Baustein beschrieben oder ausgelesen wird.

30.1 Bit-Übertragung

Um ein Bit als gültig zu werten, muss SCL high sein. SDA darf sich währenddessen nicht ändern (es sei denn, es handelt sich um die Start- oder Stoppbedingung, doch dazu später mehr). Um beispielsweise eine 1 zu übertragen, müssen SDA sowie SCL high sein. Für eine 0 muss SDA low sein, SCL jedoch high.

30.2 Startbedingung

Um die angeschlossenen ICs zu informieren, dass eine Datenübertragung beginnt, muss eine Startbedingung erzeugt werden. Vorher kann keine Datenübertragung erfolgen. Eine Startbedingung wird erzeugt, indem, während SCL high ist, SDA von high auf low wechselt.

30.3 Stoppbedingung

Die Stoppbedingung funktioniert genau anders herum: SCL muss high sein und während dieser Phase wechselt SDA von low auf high. Die Stoppbedingung beendet, wie der Name schon vermuten lässt, eine Datenübertragung. So kann der Master signalisieren, dass er keine weiteren Daten empfangen oder senden möchte.

30.4 Byte-Übertragung

Wenn ein Byte verschickt werden soll, wird als Erstes das höchstwertige Bit verschickt. Dann folgen die anderen bis hin zum niedrigstwertigen.

30.5 Bestätigung (Acknowledgment)

Der Empfänger quittiert den Erhalt der Daten mit einer Bestätigung (auch *Acknowledgment* genannt). Nach acht Datenbits und folglich auch acht Taktimpulsen wird eine Bestätigung erzeugt.

30.6 Adressierung

Das erste Byte nach der Startbedingung, das der Master verschickt, ist die Adresse des Slaves, den er ansprechen möchte.

7-Bit-Adressierung
Die 7-Bit-Adressierung ist die erste Adressierungsform des I²C-Busses und ermöglicht bis zu 128 (27) Geräte an einem Bus. Dies ist auch der am weitesten verbreitete Standard.

Beispiel:

```
Sub main()

      I2C_Init(I2C_100kHz)      'I²C-Bit-Rate: 100kHz

      I2C_Start()
      I2C_Write(Slave_ID)       'Adresse des Chips z. B. und H40
      I2C_Write(Daten_Byte)     'zu schreibende Byte
      I2C_Stop()

End Sub
```

Man schreibt hier ein Byte an die gewünschte Adresse. Das folgende Beispiel zum LM75-Temperatur-Sensor zeigt, wie das Lesen eines Bytes aussieht.

Beispiel:

```
Sub main()

Dim MSB As Byte
Dim LSB As Byte
Dim Grad As Single

  I2C_Init(I2C_100 kHz)           'I²C-Bit-Rate: 100kHz

  I2C_Start()                     'I²C-Bus starten

  I2C_Write(&H91)                 'LM75 mit seiner Adresse
                                  'ansprechen
  MSB = I2C_Read_ACK()            'Highbyte lesen
  LSB = I2C_Read_NACK()           'Lowbyte lesen
  I2C_Stop()                      'I²C-Bus stoppen

  If MSB < 0x80 Then
        Grad = ((MSB*10) + (((LSB And 0x80) >> 7)*5.0))
     Else
```

```
        Grad = ((MSB*10) + (((LSB And 0x80) >> 7)*5.0))
        Grad = -(2555.0-Grad)
  End If

  Msg_WriteFloat(Grad)
  Msg_WriteChar(10)
  Msg_WriteChar(13)
End Sub
```

Das High- und das Lowbyte sind jetzt in der Variablen *MSB* und *LSB* abgelegt. Man könnte die Daten jetzt auswerten oder über die RS232-Schnittstelle zur Kontrolle an das Terminal ausgeben.

31 I²C-LCD-Modul

Abb. 31.1: I²C-BUS-LC-Display LCD

Es soll ja vorkommen, dass die Pins am Controller einfach nicht mehr ausreichen, um noch ein LCD-Display mit anzuschließen. Dafür gibt es jedoch eine Lösung: das I²C-Display. Es besteht aus dem eigentlichen LC-Display und einem I²C-Port-Expander, dem PCF8574. Er stellt zusätzlich 8 I/O-Aus- bzw. Eingänge über den I²C-Bus zu Verfügung.

Mithilfe einer Unterroutine für die C-Control PRO ist es möglich, solch ein I²C-Display anzusprechen. Der Nachteil ist, dass die Übertragung etwas länger dauert als bei einem direkt am Controller angeschlossenen Display über einzelne I/O-Ports, da die Daten zum Display seriell (nacheinander) übertragen werden müssen. Dennoch ist es oft die beste Lösung bei Port-Mangel.

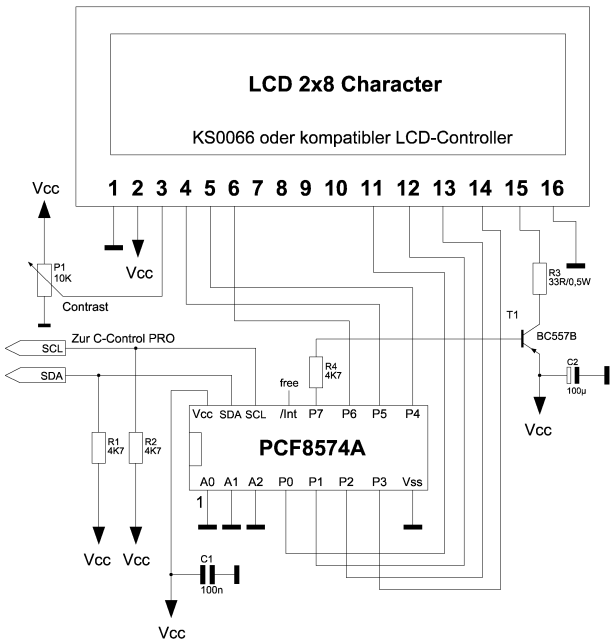


Abb. 31.2: Schaltplan für die Selbstbauer

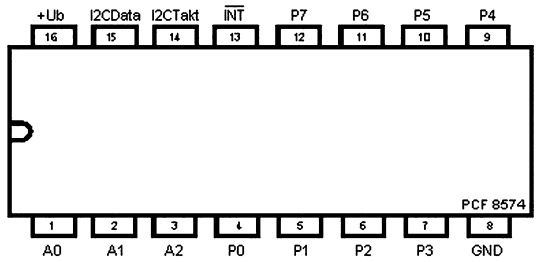


Abb. 31.3: Pinout des PCF8574

Es folgt ein abschließendes Beispiel, wie die Anweisung zur Text- und Zählervariablenausgabe aussehen soll. Der komplette Code ist auf der Buch-CD enthalten. Aus Platzgründen konnte er hier nicht komplett gedruckt werden.

Beispiel:

```
Sub main()

    'Init
    I2C_Init(I2C_100 kHz)          'I²C-Bit-Rate: 100 kHz
    LCD_Initialisieren()           'LCD initialisieren auf 4-Bit-Betrieb
    LCD_Clear()                    'Display löschen

    Do While (1)

       LCD_Ausgabe()               'Infotext und ADC-Wert ausgeben
       AbsDelay(500)

    End While

End Sub

'LCD Ausgabe (Ausgabetext = C-Control PRO)
 Sub LCD_Ausgabe()

    'Zeichenausgabe Test
    LCD_Locate_I2C(1,1)            'Zeile 1, Pos 1
    Text = „CTC C-Control“         'auszugebender Text
    LCD_Write_Text_I2C()           'Text schreiben

    LCD_Locate_I2C(2,1)            'Zähler
    LCD_Write_Word_I2C(X,5)
    X = X + 1

End Sub
```

32 I²C-Port-Expander mit PCF8574

Abb. 32.1: Ein fertiger Port-Expander der C-Control I M-Unit 2.0, der sich in Verbindung des Applicationboards und der Moduladapterplatine für die C-Control PRO einfach aufstecken lässt (siehe Moduladapter CC1 auf PRO).

Ein typisches und oft verwendetes I²C-Bus-IC ist der Port-Erweiterungsbaustein PCF8574 von Philips. Er besitzt acht *bidirektionale* Port-Leitungen. Sie sind im Ruhezustand *ON* und hochohmig, sodass sie als Eingänge einsetzbar sind. Alle Leitungen lassen sich als Ausgänge aktiv herunterziehen und sind dann niederohmig. Ein Datenrichtungsregister wie die C-Control PRO besitzt dieser Baustein jedoch nicht. Der IC besitzt die Basisadresse 01000000 (64) oder Hex40 und kann über seine drei Adresseingänge AO bis A2 Unteradressen bis 01001110 (71) Hex47 belegen. Es lassen sich bis zu acht gleiche ICs an einem Bus betreiben. Der PCF8574 eignet sich daher gut zur Port-Erweiterung auf bis zu 64 Port-Leitungen.

PCF8574 Beschaltung

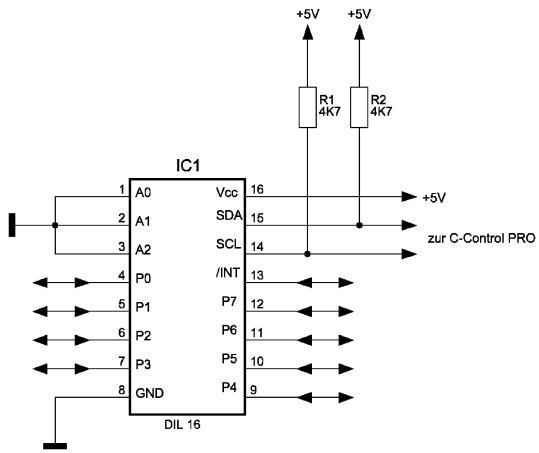


Abb. 32.2: Die Beschaltung des PCF8574

Das folgende Programm demonstriert die Ein- und Ausgaben über den I²C-Bus. Es stellt zugleich eine universell einsetzbare Sammlung von Basisroutinen zum I²C-Bus bereit.

Das Hauptprogramm demonstriert den Aufbau einer typischen Datenverbindung. Hier sollen aufsteigende Zahlenfolgen an den Erweiterungs-Port gesendet werden, wobei jedes Mal der Portzustand zurückgelesen wird. Bei unbeschalteten Portanschlüssen des PCF8574 werden alle gesendeten Daten unverändert wieder zurückgelesen. Schaltet man dagegen einen Anschluss an Masse, wird das entsprechende Datenbit als *OFF* gelesen.

Jeder Schreibzugriff wird zunächst von der Startbedingung und der Slave-Adresse eingeleitet. Dann folgt die Übertragung eines Datenbytes, das als Bitmuster direkt an den Portanschlüssen erscheint. Prinzipiell könnte man beliebig viele weitere Daten-Bytes folgen lassen, sodass sich auch die schnelle Ausgabe veränderlicher Bitmuster durchführen lässt. So wird es z. B. beim I²C-Bus-LCD gemacht. Hier wird jedoch immer nur ein Byte ausgegeben und der Bus dann durch die Stoppbedingung wieder in den Ruhezustand versetzt.

Nach jedem Schreibzugriff soll jeweils ein Lesezugriff erfolgen. Dazu muss der IC erneut adressiert werden, wobei diesmal mit gesetztem Datenrichtungs-Bit die Adresse Slave_addr+1 gilt. Der eigentliche Lesezugriff für ein Byte erfolgt mit *I2C_READ()*. Prinzipiell können auch hier beliebig viele Lesezugriffe nacheinander ausgeführt werden, denen dann jeweils ein Aufruf des Unterprogramms ACK folgen muss. Hier folgt jedoch nach dem einzigen und zugleich letzten Lesezugriff *NACK*, um das Ende der Übertragung anzukündigen. Mit *I2C_STOP()* wird der Bus dann wieder freigegeben. Die Datenübertragung ist über I²C-Bus nicht sonderlich schnell. Der wesentliche Vorteil liegt darin, dass sehr viele ICs über nur zwei Port-Leitungen angesteuert werden können.

Beispiel:

```
'PCF8574 schreiben (Applicationboard Mega 32/ 128)
#define PCF8574_W &H40
#define PCF8574_R &H41
Dim Zeile1(9) As Char
Dim Zeile2(9) As Char
Dim X As Word
Dim Port As Byte

Sub main()

    'Init
    I²C_Init(I²C_100kHz)                'I²C Bit Rate: 100kHz
    LCD_start()                         'LCD Init

    'Info
```

```
    Zeile1 = "PCF8571T"
    LCD_Locate(1,1) : LCD_WriteText(Zeile1)
    Zeile2 = „  DEMO  "
    LCD_Locate(2,1) : LCD_WriteText(Zeile2)
    AbsDelay(2500)
    LCD_start()

    Zeile1 = "Binaer- "
    LCD_Locate(1,1) : LCD_WriteText(Zeile1)
    Zeile2 = "Zaehler "
    LCD_Locate(2,1) : LCD_WriteText(Zeile2)

    X = &H0

    'Alle Ports „low"
    I²C_Start()
    I²C_Write(PCF8574_W)
    I²C_Write(0)
    I²C_Stop()
    AbsDelay(500)
    Do While(1)
       X = X + 1
       I²C_Start()
       I²C_Write(PCF8574_W)
       I²C_Write(X)
       I²C_Stop()
       AbsDelay(50)
       If X = &HFF Then
          'Alle Ports „low"
          I²C_Start()
          I²C_Write(PCF8574_W)
          I²C_Write(0)
          I²C_Stop()
          AbsDelay(500)
          Exit
       End If
    End While

End Sub

Sub LCD_start()
    LCD_Init()
    LCD_CursorOff()
    LCD_ClearLCD()
End Sub
```

Sachverzeichnis

FRANZIS
PC+ELEKTRONIK

Ulli Sommer

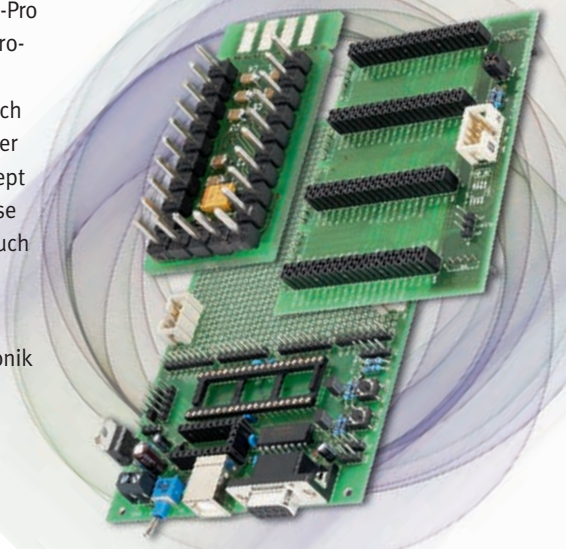
C-Control-Pro

selbst programmieren und in der Praxis einsetzen

Mit C-Control-Pro hat Conrad Electronic eine neu konzipierte Systemgeneration von Mikrocontrollern auf den Markt gebracht, die sich aufgrund ihrer Leistungsmerkmale und der Programmierung in C-Compact und Basic besonders für Anwendungen in der professionellen Mess-, Steuer- und Regeltechnik (MSR) eignet.

Aber auch im semiprofessionellen Bereich lässt sich C-Control-Pro erfolgreich einsetzen. Ein weiterer Vorteil der C-Control-Mikrocontroller ist ihr reichhaltiges Angebot von Beispiel-Codesammlungen und Literatur in deutscher Sprache. Dieses Buch vermittelt einen Überblick über die neuen Mikrocontroller der C-Control-Pro-Familie. Es werden die Grundlagen, das Konzept und die Programmierung auf kompakte und praktische Weise dargestellt. Zahlreiche Anwendungsbeispiele runden das Buch ab.

Die meisten Publikationen setzen fundiertes Grundwissen voraus – dieses Buch bringt auch dem Einsteiger die Elektronik und das Programmieren auf einfache Weise nahe.



Aus dem Inhalt:

- Mikrocontroller-Grundlagen
- Mikrocontroller-Anwendungen
- Anwendungsgebiete von C-Control-Pro
- Aufbau und Funktionsweise
- Einstieg in die Programmierung
- C-Control-Pro in der eigenen Schaltung
- Daten auswerten und am PC visualisieren
- Praktische Anwendungen

ISBN 978-3-7723-5089-4

Euro 19,95 [D]