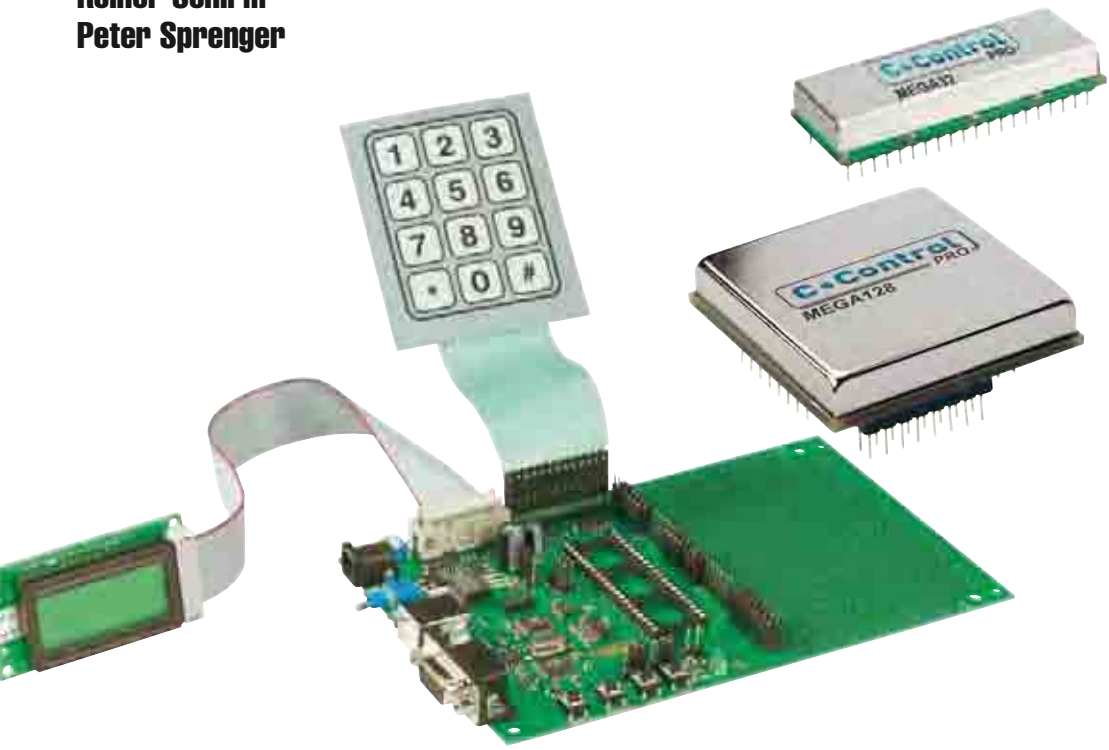
PC & Elektronik

Reiner Schirm
Peter Sprenger




# Messen, Steuern und Regeln mit C-Control Pro

## Praxisanwendungen, Schaltungstechnik und Programmierung

FRANZIS

# Vorwort

Das C-Control-Pro-System basiert auf dem Atmel Mega 32 RISC-Mikrocontroller. Dieser Mikrocontroller findet seinen Einsatz in Geräten von der Unterhaltungselektronik über Haushaltsmaschinen bis hin zu verschiedenen Industrieapplikationen. Dort übernimmt der Controller wichtige Steuerungsaufgaben. Mit C-Control Pro lassen sich beispielsweise analoge Messwerte und Schalterstellungen erfassen und abhängig von diesen Eingangsbedingungen entsprechende Schaltsignale ausgeben. In Verbindung mit einer DCF77-Funkantenne kann C-Control Pro die atomgenaue Uhrzeit empfangen und präzise Schaltuhrfunktionen übernehmen. Verschiedene Hardware-Schnittstellen und Bussysteme erlauben die Vernetzung von C-Control Pro mit Sensoren, Aktoren und anderen Steuerungssystemen.

Die in diesem Buch vorgestellten Demos und Beispiele wurden sorgfältig und ausführlich getestet. Da es aber eine Vielzahl von Hardware- und Softwarevariationen gibt, kann für ein ordnungsgemäßes Funktionieren dieser Programme keine Garantie übernommen werden.

# Inhalt


**1 Einleitung** . . . . . 11

**2 Mega32** . . . . . 15

**3 Application-Board Mega32** . . . . . 16

**4 Mega128** . . . . . 18

**5 Application-Board Mega128** . . . . . 19

**6 Hardware-Einstellung** . . . . . 21
6.1 Application-Board MEGA32 . . . . . 21
6.1.1 Programmierung über USB . . . . . 21
6.1.2 Programmierung über RS232 . . . . . 22
6.2 Application-Board M128 . . . . . 23
6.2.1 Programmierung über USB . . . . . 24
6.2.2 Programmierung über RS232 . . . . . 25

**7 Software-Installation** . . . . . 28
7.1 Entwicklungsumgebung . . . . . 28
7.2 USB-Treiber . . . . . 32

**8 Software-Einstellungen** . . . . . 37
8.1 IDE-Update . . . . . 37
8.2 Compiler . . . . . 38
8.3 Editor . . . . . 39

**9 Das erste Programm** . . . . . 41
9.1 Programmierung . . . . . 41
9.2 Fehlersuche . . . . . 46
9.2.1 Software . . . . . 48
9.2.2 Hardware . . . . . 58

**10 C und Basic in einem Projekt** . . . . . 60

**11 Schutz der Programme (PIN)** . . . . . 63

**12 Anschluss externer Komponenten** . . . . . 67
12.1 DCF-Modul . . . . . 67

12.2 LCD Display 4 × 20 ... 70
12.3 Sensoren ... 74
12.3.1 Digitale Sensoren ... 74
12.3.2 Analoge Sensoren ... 76
12.4 CCI Relais-Modul ... 79
12.5 I²C-Bus-Thermometer-Modul ... 83
12.6 I²C-Bus-Tastatur ... 87

**13 Stringverarbeitung** ... 91
13.1 Strings in der C-Control-Pro-Umgebung ... 91
13.2 Strings sind Arrays ... 91
13.3 Stringfunktionen in der Bibliothek ... 92
13.4 Stringbearbeitung – selbst gemacht ... 93
13.5 Steuerzeichen ... 98
13.6 Formatierung numerischer Werte ... 99

**14 Optimierung von CompactC** ... 101
14.1 Optimierung ist Programmierersache ... 101
14.2 Optimierung Schritt für Schritt ... 102
14.3 Switch-Anweisungen sind effizient ... 104
14.4 Arithmetische Ausdrücke vereinfachen ... 105
14.5 Eingliedern von Funktionen ... 106
14.6 Einsparen von Programmcode ... 108
14.7 Projektoptionen prüfen ... 110

**15 Optimierung von BASIC** ... 113
15.1 Optimierung ist Programmierersache ... 113
15.2 Optimierung Schritt für Schritt ... 113
15.3 Select-Case-Anweisungen sind effizient ... 115
15.4 For-Schleifen benutzen ... 116
15.5 Arithmetische Ausdrücke vereinfachen ... 117
15.6 Eingliedern von Funktionen ... 118
15.7 Einsparen von Programmcode ... 120
15.8 Projektoptionen prüfen ... 122

**16 Der Preprozessor** ... 124
16.1 Definitionen ... 124
16.2 Bedingte Kompilierung ... 126
16.3 Einfügen von Dateien ... 128
16.4 Preprozessor-Makros ... 129
16.5 Vordefinierte Symbole ... 131
16.6 Compiler-Anweisungen ... 133
16.7 Mischen von BASIC und CompactC ... 133

**17 Interruptbehandlung** . . . . . 135
17.1 C-Control-Pro-Interrupts . . . . . 135
17.2 Externe Interrupts . . . . . 137
17.3 Interpreter-Interrupts im Detail . . . . . 139

**18 Multithreading** . . . . . 140
18.1 Starten von Threads . . . . . 140
18.2 Konfiguration des Multithreadings . . . . . 142
18.3 Warten in Threads . . . . . 145
18.4 Threads synchronisieren . . . . . 147
18.5 Multithreading im Detail . . . . . 149

**19 Anwendungen** . . . . . 151
19.1 Voltmeter . . . . . 151
19.2 Heizungssteuerung mit NTC-Sensoren . . . . . 154
19.3 Heizungssteuerung mit Raumtemperaturregler . . . . . 163
19.4 Temperaturschalter mit Sensorüberwachung . . . . . 168
19.5 Zwei-Kanal-Thermometer . . . . . 171
19.6 Temperatur-Differenzschalter . . . . . 174
19.7 Acht-Kanal-Lauflicht . . . . . 177
19.8 Digital-Timer . . . . . 181
19.9 Stoppuhr . . . . . 188
19.10 Gewächshausreglung . . . . . 193
19.11 3-Kanal-DCF-Zeitschaltuhr . . . . . 201
19.12 Ein-/Ausschaltverzögerung . . . . . 211

**20 Der Bytecode-Interpreter** . . . . . 215
20.1 Die Speicherbereiche im Interpreter . . . . . 215
20.2 Die Arbeitsweise des Arithmetik-Stacks . . . . . 216
20.3 Beispiel: Zuweisung . . . . . 217
20.4 Beispiel: Funktionsaufruf . . . . . 219
20.5 Beispiel: if-Anweisung . . . . . 221
20.6 Beispiel: for-Schleife und Array-Zugriff . . . . . 222
20.7 Beispiel: switch-Anweisung . . . . . 224

**21 Anhang – Bytecode-Übersicht** . . . . . 226
21.1 Bytecode-Übersicht . . . . . 226

**Sachverzeichnis** . . . . . 241

# 10 C und Basic in einem Projekt

Die IDE der C-Control Pro bietet die Möglichkeit, in einem Projekt Programme in C und Basic zu verwenden. Das hat den Vorteil, dass Module der verschiedenen Sprachen ohne zeitraubende Übersetzungsarbeit miteinander kombiniert werden können.

Als Beispiel ist nachfolgend das komplette Beispielprogramm aus „9 Das Erste Programm“ in C dargestellt. Geben Sie dieses bitte wie unter Punkt 9 beschrieben ein.

Der Unterschied bei der Eingabe liegt bei Abb. 9.5. Hier wählen Sie bitte als Dateityp „CompactC Dateien(*.cc)“. Als Projektnamen verwenden Sie bitte „ZweitesProgramm“.


Die Projektübersicht sollte am Schluss aussehen wie in Abb. 10.1.

Abb. 10.1: Zweites Programm

```
0  // Ein- und Ausgabe
1  // Verwendung der Tasten SW1
2  // LED1 leuchtet nach dem Drücken von SW1
3  // erforderliche Library: IntFunc_Lib.cc
4
5  // LD1 leuchtet nach dem Drücken von SW1 für 2 Sekunden
6  // Beide LEDs werden direkt über Einzelpinzugriff angesprochen
7
8  int delval;                              // globale Variablen deklarieen
9  byte schalter1;
10
11 // ---------------------------------------------------------------------
12 // Hauptprogramm
13 //
14 void main(void)
15 {                                        // Anfang des Hauptprogrammes
16     delval=2000;                         // Verzögerungszeit: 2s
17     Port_DataDirBit(PORT_LED1,PORT_OUT); // LED1 auf Ausgabe vorbereiten
18     Port_DataDirBit(PORT_SW1,PORT_IN);   // SW1  auf Eingabe vorbereiten
19     Port_WriteBit(PORT_LED1,PORT_OFF);   // LED1 ausschalten
20     while(1)                             // Endlosschleife wird gestartet
21     {
22         SW1();                           // Funktion SW1 wird abgefragt
23         if (schalter1==0)                // Auswertung der Tasterstellung
24         {
25             LED1_On(delval);             // Funktion LED_On wird mit
26                                          // Wertübergabe aufgerufen
27         }                                // Ende der Auswertung
28     }                                    // Ende der Endlosschleife
29 }                                        // Ende des Hauptprogrammes
```

Abb. 10.2: Zweites Programm (Main)

```
0  // ------------------------------------------------------------------------
1  // Einlesen der Stellung des Tasters 1
2  //
3  void SW1(void)                              // Anfang der Funktion SW1
4  {
5      schalter1=Port_ReadBit(PORT_SW1);       // Tasterstellung wird Ausgelesen
6  }                                           // Ende der Funktion SW1
```

Abb. 10.3: Zweites Programm (Eingabe)

```
0  // ------------------------------------------------------------------------
1  // Ein- und Ausschalten der LED 1
2  //
3  void LED1_On(int delay_val)                 // Anfang der Funktion LED1_ON
4  {                                           // mit Variablendeklaration.
5                                              // Die Variable delay_val erhält
6                                              // den Wert der Variablen delay.
7                                              // Dies erfolgt mit dem Funktions-
8                                              // aufruf LED1_On(delval)
9      Port_WriteBit(PORT_LED1,PORT_ON);       // Die LD1 wird eingeschaltet.
10     AbsDelay(delay_val);                    // Die mit delval festgelegte
11                                             // Verzögerungszeit wird
12                                             // abgearbeitet.
13     Port_WriteBit(PORT_LED1,PORT_OFF);      // Die LD1 wird ausgeschaltet.
14 }                                           // Ende der Funktion LED1_ON.
```

Abb. 10.4: Zweites Programm (Ausgabe)

Um nun Module aus den Programmen „ErstesProgramm“ und „ZweitesProgramm“ zu mischen, gehen Sie folgendermaßen vor. Schließen sie alle momentan geöffneten Projekte. Erstellen Sie ein neues Projekt. Verwenden Sie z. B. den Namen „Drittes-Programm“. Klicken Sie mit der rechten Maustaste auf den Projektnamen „Drittes-Programm“. Im erscheinenden Kontextmenü wählen Sie bitte den Punkt „Datei Hinzufügen“.


Abb. 10.5: Datei Hinzufügen

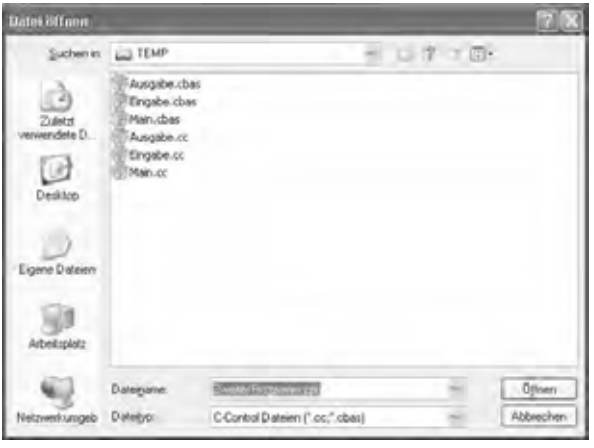



Abb. 10.6: Datei öffnen

Im nächsten Fenster können Sie nun die gewünschten Programmmodule Ihrem Projekt hinzufügen. Wählen Sie zum Beispiel „Main.cc“, „Ausgabe.cbas“ und „Eingabe.cbas“

Wenn Sie nun das Projekt kompilieren und auf die C-Control Pro übertragen, können Sie feststellen, dass die Programme ohne Probleme miteinander funktionieren.

Da die Funktionen der Programme „Main.cc“ und „Main.cbas“, „Ausgabe.cc“ und „Ausgabe.cbas“, „Eingabe.cc“ und „Eingabe.cbas“ in den Projekten „ErstesProgramm“ und „ZweitesProgramm“ jeweils gleich sind, macht es natürlich nur Sinn, Unterprogramme mit verschiedenen Namen zu kombinieren („Main“, „Ausgabe“, „Eingabe“). In den Programmen werden Funktionen mit den gleichen Namen verwendet, dadurch würde die Kombination von z. B. „Main.cc“ und „Main.cbas“ zu Fehlermeldungen führen.

# Sachverzeichnis

**A**
AbsDelay 145
Add16Sig 217
aktiv 149
Anführungszeichen 98
Arithmetik-Stack 216
arithmetische Ausdrücke 105, 117
Array-Zugriff 222
ASCII 91
Ausführungszeit 145

**B**
Bedingte Kompilierung 126
Bibliotheksoptionen 122

**C**
Call 220
Carriage Return 98
C-Control-ProInterrupts 135
Character 216
Copy2 115

**D**
DATE 132
Datentypen 216
DEBUG 131
DecS 217
#define 124
Dekrement Operator 102
Dezimalzahlen 99
Drop 220
#elif 126
#else 126
#endif 126

**E**
ExterneInterrupts 137
Ext_IntDisable 138
Ext_IntEnable 137

**F**
FILE 132
Floating-Point 216
Formfeed 98
For-Schleifen 116
FUNCTION 132
Funktionsaufruf 219

**G**
GenFunc 219
gesichert 149
Glocke 98
Gnu Generic Preprocessor 124
Goto 221
Gr16BitSig 221

**H**
Hardware Interrupts 135
Hauptthread 140
Hexadezimale 99
hochspezialisierte Bytecodes 102

**I**
#if 126
#ifdef 126
#ifndef 126
inaktiv 149
#include 128
IncSP_8Bit 217

Inkrement-Operator 102
INT_0 135
INT_1 135
INT_2 135
INT_ADC 136
INT_ANA_COMP 135
Interpreter-Speicher 215
INT_TIM0COMP 135
INT_TIM1CAPT 135
INT_TIM1CMPA 135
INT_TIM1CMPB 135
INT_TIM1OVF 135
Inv16 217
Irq_GetCount 136
Irq_SetVect 136

**L**
LINE 132
LoadAdrTopRel 223
LoadImm8 217
LoadTopAdr8 217

**M**
Makros 129
Map-Datei 143
MAPFILE 131
MEGA128 131
MEGA32 131
Multithreading 140

**O**
$opc 219
Optimierung 101
#pragma Error 133
#pragma Language 133
#pragma Message 133
#pragma Warning 133
#pragma 133

**P**
Preprozessor Makros 129
Preprozessor 124
Prioritätenvon Threads 145
Programm-Stack 216
Projektoptionen 110
Projektoptionen 122

**R**
rekursive Funktionen 144
Rückstrich 98
Rücktaste 98

**S**
schlafend 149
Select-Case-Anweisungen 115
Signal 148
Speicherbereiche 215
Speicherverbrauch 143
Stackgröße 142
Stacklift 217
Stackpointer 216
Steuerzeichen 98
StoreTopAdr8 217
STR 92
Str 92
Str 94
Str 95
Strings 91
switch-Anweisungen 104

**T**
Tabulator 98
Thread_Delay 146
Thread_Kill 140
Threadkonfiguration 123, 142
Thread_Lock 149
Thread_MemFree 144
Thread_Resume 147
Thread_Signal 147
Threads synchronisieren 147
Thread_Start 140
Thread_Wait 147
TIME 132

**U**
#undef 126
UPN 216

**V**
Vertikaler Tabulator 98
Vordefinierte Symbole 131

**W**
wartend 149
Write Word 99
WriteSP2 219

**Z**
Zahlenbasis 99
Zeichenketten 91
Zeilenvorschub 98
Zykluszeit 145