

## Messen, Steuern und Regeln mit C-Control M-Unit 2

Der Ein-Chip-Mikrocontroller von Conrad Electronic für C-Control Generation 2.0

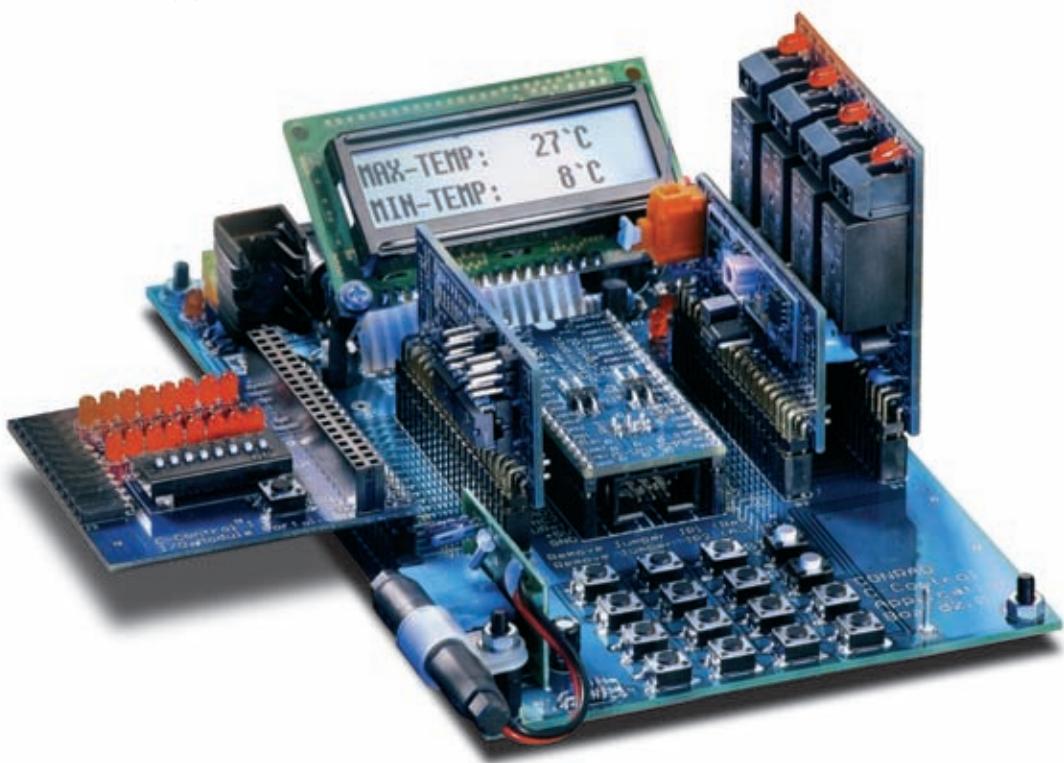
von  
Stefan Tappertzhofen

1. Auflage

Franzis Verlag, Poing 2007

Verlag C.H. Beck im Internet:  
[www.beck.de](http://www.beck.de)  
ISBN 978 3 7723 5488 5

Stefan Tappertzhofen



2. aktualisierte Auflage

# Messen, Steuern und Regeln mit **C-Control M-Unit 2**

Der Ein-Chip-Mikrocontroller von Conrad Electronic  
für C-Control Generation 2.0



## Auf CD-ROM:

- Anwendungsprogramme
- Beispiele
- Datenblätter

# Vorwort

In den letzten zwei bis drei Jahren wurden viele neue Entwicklungen rund um die Mikrocontroller-Familie „C-Control“ der Firma *Conrad Electronic* gemacht. Die Programmiersprache BASIC++ 1.0 wurde durch eine neue und leistungsfähigere Version verbessert: BASIC++ 2006. Auch die Mikrocontroller selbst wurden weiterentwickelt. Für professionelle Anwendungen sind jetzt sogar Controller mit implementierter 32-Bit-Fließkomma-Arithmetik verfügbar. Es war daher erforderlich, dieses Buch komplett zu überarbeiten.

Aus didaktischen Gründen habe ich mich dazu entschlossen, einige Kapitel zusammenzufassen, andere zu streichen und dafür neue Kapitel einzufügen. Anders als in der ersten Auflage wird der C-Control *Micro* nun kein separates Kapitel mehr gewidmet. Stattdessen werden alle Controller, von der M-Unit 2, über die Station 2.0, bis hin zu der neuen Advanced Version mit 32-Bit-FP-Arithmetik, im ersten Kapitel ausführlich beschrieben.

Die Veröffentlichung des Assemblers CCASM, mit dem nun auch Maschinen-Codeprogramme für die C-Control M-Unit 2, Micro und Station 2.0 geschrieben werden können, war Anlass genug, ihm in diesem Buch ein spezielles Kapitel zu widmen. Der fortgeschrittene Entwickler erhält im Kapitel „Assembler und Systemtreiber“ einen vertiefenden Einblick in die Hardware des Mikroprozessors.

Um der Komplexität der Programmiersprache BASIC++ in ihrer aktuellsten Version BASIC++ 2006 gerecht zu werden, wurde Kapitel 2 um ein weiteres Kapitel ergänzt, das sich vorwiegend an fortgeschrittene Programmierer richtet.

Die Strukturierung der Kapitel in der ersten Auflage war weitgehend beispielorientiert. Die Kapitel der vorliegenden Auflage orientieren sich hingegen mehr an Hardware und Peripherie des Controllers. Ein Kapitel ist speziell den digitalen Ein- und Ausgängen gewidmet. Ein anderes Kapitel beschäftigt sich mit den A/D- bzw. D/A-Wandlern. Der Leser erhält so einen besseren Überblick, ohne das komplette Buch lesen zu müssen.

Wie schon in der ersten Auflage, möchte ich auch an dieser Stelle denen danken, die zum Gelingen beider Auflagen beigetragen haben, allen voran Uwe Spies, Dierk Schmid und Henrik Reimers. Auf der Seite des Franzis-Verlags wären vor allem Herr Günther Wahl und Michael Büge zu nennen. Einen herzlichen Dank auch

denen, die mit ihren regen Beiträgen im Online-Forum einen großen Anteil an der Weiterentwicklung der C-Control und BASIC++ haben. Schließlich danke ich noch meinen Eltern und den vielen Helfern im Hintergrund, ohne deren Unterstützung die erste und zweite Auflage des Buchs nie hätte vervollständigt werden können.

Düsseldorf, im Juni 2007

Stefan Tappertzhofen

# Inhalt

<b>1 Erste Schritte .....</b>	<b>11</b>
1.1 Controllerversionen .....	11
1.1.1 M-Unit 2.0 .....	13
1.1.2 Station 2.0 .....	13
1.1.3 Micro .....	13
1.1.4 Advanced Versionen .....	19
1.2 Unterschiede zum Vorgänger .....	20
1.3 Überblick über die Programmiersoftware .....	21
1.4 Inbetriebnahme der C-Control .....	23
<b>2 Einführung in BASIC++ .....</b>	<b>26</b>
2.1 Umstieg von CCBasic auf BASIC++ .....	26
2.2 Die WorkBench++ .....	28
2.2.1 Installation und Inbetriebnahme .....	28
2.2.2 Der Code-Explorer .....	29
2.2.3 Die Speicher-Map .....	30
2.3 Globale Variablen und Konstanten .....	30
2.4 Selektive Programmierung .....	32
2.4.1 IF-Bedingung .....	32
2.4.2 SELECT-CASE-Anweisung .....	33
2.5 Schleifen .....	34
2.5.1 DO-Schleife .....	35
2.5.2 FOR-Schleife .....	36
2.5.3 WHILE-Schleife .....	36
2.6 Funktionen, Sprungmarken und lokale Variablen .....	37
2.6.3 Lokale Variablen .....	38
2.6.4 Referenzen .....	38
2.7 Binäre Operationen .....	39
<b>3 Fortgeschrittene Programmietechniken .....</b>	<b>41</b>
3.1 Split-Variablen .....	41
3.2 Speicher- und Zeigeroperationen .....	43
3.3 Strings .....	46

3.4	Fehlerbehandlungsroutine .....	48
3.5	Dynamische Lokale Variablen .....	50
3.6	Fließkomma-Arithmetik .....	51
3.6.1	Einfache Berechnungen. ....	52
3.6.2	Wissenschaftliche Notation .....	53
3.6.3	Typenkonvertierung .....	54
3.6.4	Weitere Funktionen und FP-Bibliothek .....	55
3.6.5	Fehlerbehandlung .....	56
3.7	Interrupts .....	57
3.8	Propertys .....	58
3.9	Vorkompilierte Bibliotheken .....	60
3.10	Präprozessor Anweisungen .....	60
3.11	Dynamischen Code zur Laufzeit erzeugen .....	61
3.12	Objekte .....	63
<b>4</b>	<b>Digitale I/O-Ports .....</b>	<b>64</b>
4.1	Technischer Überblick .....	64
4.2	Digitale Ausgänge .....	68
4.3	Digitale Eingänge .....	69
4.3.2	Taster entprellen .....	71
4.4	Einfache Beschaltung .....	72
4.5	Leistungsausgänge .....	74
4.6	Schrittmotoren .....	77
<b>5</b>	<b>Zeitmessung und -Steuerung .....</b>	<b>80</b>
5.1	Die interne Echtzeituhr .....	80
5.2	Das DCF-77 Funkuhrmodul .....	85
5.3	Zeit- und datumsgesteuerte Programme .....	89
5.4	Zeitmessung .....	91
5.4.1	Große Zeitmessung .....	91
5.4.2	Zeitmessung mit hoher Auflösung .....	92
5.5	Frequenzmessung .....	94
5.5.1	Einfache Frequenzmessung .....	94
5.5.2	Frequenzmessung von 32 bis 65 kHz .....	96
5.5.3	Messung hochfrequenter Rechtecksignale .....	97
<b>6</b>	<b>Serielle Schnittstelle .....</b>	<b>99</b>
6.1	Mehrere Controller miteinander verbinden .....	99
6.2	Fernsteuerung mit rs232-Server .....	104
6.3	Daten offline auslesen .....	105

<b>7 Analogie Ein- und Ausgänge .....</b>	109
7.1 Analog/Digital-Wandler .....	109
7.1.1 Zusätzliche I/O-Ports .....	112
7.2 Messverstärker .....	113
7.3 Temperaturmessung .....	115
7.4 Digital/Analog-Wandler .....	121
7.4.1 Leistungsausgänge für Motoren und Verbraucher .....	123
7.4.2 Servo-Mode .....	124
7.5 Regelungstechnik .....	125
7.5.1 Soll- und Istgröße .....	126
7.5.2 Reglertypen .....	127
7.5.3 Regelkreise .....	129
<b>8 Erweiterungsmodule und Extended Functions .....</b>	133
8.1 Interne Config-Register .....	133
8.1.1 Config1 .....	134
8.1.2 Config2 .....	136
8.2 433 MHz Funkübertragung .....	138
8.3 IR-Sender/Empfänger-Modul .....	144
8.4 Chipram .....	149
8.4.1 Externes EEPROM als Arbeitsspeicher .....	149
8.4.2 Programmcode von Chipkarte lesen .....	151
<b>9 I<sup>2</sup>C-Schnittstelle .....</b>	152
9.1 Technischer Überblick .....	152
9.2 PCF-8574-Porterweiterung .....	154
9.3 Serielle I <sup>2</sup> C-EEPROMs .....	156
9.4 I <sup>2</sup> C-Dallas-Temperatursensor .....	159
9.5 Die C-Control Micro als I <sup>2</sup> C-Slave .....	166
<b>10 Assembler und Systemtreiber .....</b>	169
10.1 Übersicht über CCASM .....	169
10.1.1 Technischer Überblick .....	169
10.2 Bereiche, Speicher und Konstanten .....	170
10.3 Hardware- und Software-Register .....	172
10.3.1 Hardware-Register .....	172
10.3.2 Software-Register .....	173
10.4 Sprungmarken, Sprünge und Schleifen .....	176
10.4.1 Sprungmarken .....	176
10.4.2 Unbedingte Sprünge .....	177
10.4.3 Bedingte Sprünge .....	178

10.4.4 Schleifen .....	179
10.4.5 Unterprogramme .....	180
10.5 Operationen und Adressierungen .....	180
10.5.1 Arithmetische und logische Operationen .....	180
10.5.2 Direkte, indirekte und relative Adressierung .....	181
<b>11 Sprachreferenz .....</b>	<b>183</b>
<b>12 Anhang .....</b>	<b>198</b>
12.1 Technische Daten .....	198
12.1.1 M-Unit 2.0 .....	198
12.1.2 Station 2.0 .....	199
12.1.3 Micro .....	200
12.1.4 Advanced Versionen .....	200
12.2 Anschluss- und Steckplatzbelegung .....	202
12.2.1 M-Unit 2.0 und Advanced Version .....	202
12.2.2 Station 2.0 .....	202
12.2.3 Micro .....	202
12.2.4 Application-Board .....	203
12.2.5 Anschlüsse Schnittstellen/Erweiterungsmodul .....	204
12.3 Schaltdiagramme .....	205
12.3.1 M-Unit 2.0 und Advanced Version .....	205
12.3.2 Station 2.0 .....	206
12.3.3 Application-Board .....	207
12.4 Bezugsquellen .....	208
12.5 Tokentabelle für die C-Control .....	208
12.6 Literaturhinweise .....	215
12.7 Abbildungsverzeichnis .....	215
<b>Sachverzeichnis .....</b>	<b>217</b>

# 5 Zeitmessung und -Steuerung

Die C-Control besitzt einen internen Timer, der auch vom schon bekannten PAU-SE-Befehl verwendet wird. Ähnlich wie die Timer normaler Computer-CPUs inkrementiert auch der C-Control Timer eine Systemvariable in einem gewissen Zeitintervall (hier 20 ms).

Neben dem System-Timer verfügt die C-Control auch über eine Echtzeituhr. Oft verwendet man auch den englischen Namen: Real Time Clock (RTC). Einmal eingestellt läuft die Echtzeituhr des Controllers autonom. Die zeitliche Abweichung von der tatsächlichen Zeit hält sich dabei laut Herstellerangaben ähnlich wie bei Quarz-Uhren in Grenzen. Wem dies jedoch noch nicht reicht, kann mittels einer DFC-Antenne die Langwellen-Zeitsignale der Atomuhr der Physikalisch-Technischen Bundesanstalt in Braunschweig auswerten.

In der Mess-, Steuer- und Regeltechnik spielen zeitkritische Anwendungen eine entscheidende Rolle. Als Beispiele hierfür können Steueraufgaben des Controllers gezählt werden, die nur zu bestimmten Tageszeiten gestartet werden. Ohne aufwendige Programmierung lässt sich so eine automatische Rollladen-Steuerung verwirklichen. Der System-Timer dagegen könnte für Zeitmessungen Verwendung finden. Letztendlich lässt sich mit dem Timer auch die Konzeption von Fehlerabfangroutinen durch Zeitüberläufe (sogenannte *Time-outs*) realisieren.

Im Vergleich zu den älteren Versionen besitzt die M-Unit 2.0 (seit OS Version 2.02) auch die Möglichkeit Hintergrundprogramme über einen Timer-Interrupt zu programmieren. Diese Option ermöglicht es, äußerst komplexe Unterprogramme im Hintergrund ablaufen zu lassen.

## 5.1 Die interne Echtzeituhr

Wenn Sie die C-Control als Zeitschaltuhr einsetzen, sollten Sie die Echtzeituhr nutzen. Bevor Sie aber mit der eigentlichen Konzeption des Programms beginnen, müssen Sie sicherstellen, dass die Echtzeituhr auch richtig eingestellt ist. Wenn die C-Control vom Strom getrennt wird, hört die Echtzeituhr nicht nur auf, weiterzuzählen, es wird auch automatisch die intern gespeicherte Uhrzeit gelöscht.

Beim ersten Einsatz der Echtzeituhr muss diese natürlich gestellt werden. Wenn Sie folglich nicht bei jedem Start des Controllers die interne Uhr neu stellen wollen, müssen Sie unbedingt auf eine gesicherte Versorgung des Controllers mit der nötigen Spannung achten.

```
, ****
,
' BASIC++ Beispiele
,
' Stellen der internen Systemuhr
,
PRINT „Stelle Systemuhr auf So. 1.1.2008, 15:00:00...“
DOW = 0 ' Sonntag
DAY = 1
MONTH = 1
YEAR = 8 ' 2008
HOUR = 15
MINUTE = 0
SECOND = 0
PRINT „Die Echtzeituhr wurde aktualisiert!“
PRINT „Datum: „ & DAY & „.“ & MONTH & „.200“ & YEAR
PRINT „Uhrzeit: „ & HOUR & „:“ & MINUTE & „:“ & SECOND
END
```

### **Beispiel 5.1:** Stellen und Lesen der Echtzeituhr

Durch Zuweisung der jeweiligen Werte können Datum und Uhrzeit eingestellt werden. Die Systemvariablen wie DOW und MINUTE können dabei zur Laufzeit sowohl beschrieben als auch ausgelesen werden. Man kann allerdings die Zuweisung des Datums und der Uhrzeit durch zwei Befehle verkürzen.

```
, ****
,
' BASIC++ Beispiele
,
' Stellen der internen Systemuhr
' Alternative zu Beispiel 5.1
,
PRINT „Stelle Systemuhr auf So. 1.1.2008, 15:00:00...“
SETDATE 8, 1, 1, BPPSunday
SETTIME 15,0,0
PRINT „Die Echtzeituhr wurde aktualisiert!“
PRINT „Datum: „ & DAY & „.“ & MONTH & „.200“ & YEAR
PRINT „Uhrzeit: „ & HOUR & „:“ & MINUTE & „:“ & SECOND
END
```

### **Beispiel 5.2:** Alternatives Stellen der Echtzeituhr

Der Code in Beispiel 5.2 stellt auf die gleiche Art und Weise Datum und Uhrzeit ein wie in Beispiel 5.1. Diese Methode ist jedoch etwas übersichtlicher. SETDATE erwartet bis zu vier Parameter. Sie können auch weniger angeben, jedoch müssen Sie dabei die Reihenfolge beachten. Zuerst erwartet SETDATE das Jahr, gefolgt von Monat und Tag. Danach können Sie die DOW-Variablen setzen. Es bietet sich dabei an, die BASIC++-Konstanten für die Tage zu verwenden. SETTIME erwartet zuerst die Stunde, gefolgt von der Minute und Sekunde.

Systemvariable	Werte	
HOUR	0...23	Stunde
MINUTE	0...59	Minute
SECOND	0...59	Sekunde
YEAR	0...99	Jahr
MONTH	1...12	Monat
DAY	1...31	Tag
DOW	0...6	Wochentag

Mögliche Werte für DOW sind, neben den Zahlen zwischen 0 und 6, auch BPP-Sunday, BPPMonday, BPPTuesday, BPPWednesday, BPPThursday, BPPFriday und BPPSaturday. Es handelt sich hierbei um die BASIC++-Konstanten für den jeweiligen Wochentag.

```

' ****
,
' BASIC++ Beispiele
,
' Formatierte Datumsausgabe
,
' ****
DEFINE Jahr AS WORD
SETDATE 8, 1, 1, BPPSunday
SETTIME 15,0,0
PRINT „Datum: „;
SELECT CASE DOW
    CASE BPPSunday
        PRINT „Sonntag“;
    CASE BPPMonday
        PRINT „Montag“;
    CASE BPPTuesday
        PRINT „Dienstag“;
    CASE BPPWednesday
        PRINT „Mittwoch“;
    CASE BPPThursday
        PRINT „Donnerstag“;
    CASE BPPSaturday

```

```

        PRINT „Samstag“;
END SELECT
Jahr = YEAR + 2000
PRINT „, „ & DAY & „.“ & MONTH & „.“ & Jahr
END

```

### Beispiel 5.3: Formatierte Datumsanzeige

Selbstverständlich können Sie die Datums- oder Uhrzeitausgabe auch nach Ihren Vorstellungen formatiert ausgeben. In Beispiel 5.3 wird so, anstatt der Konstante für den Wochentag, mit einer SELECT-CASE-Abfrage der Wert der DOW-Variablen entsprechend als Zeichenkette ausgegeben.

Eine sinnvolle Anwendung für Datumsvariable und Echtzeituhr könnte die Ausgabe auf dem LC-Display des Application-Boards sein. Eine digitale Zeitanzeige sollte jedoch dementsprechend konzipiert werden, dass nur bei der Aktualisierung der Uhrzeit eine neue Ausgabe auf das Display gelenkt wird. Würde man die Anzeige in einer DO-LOOP-Schleife ohne Unterbrechung durchführen, würde dies zu einem unleserlichen Flimmern der Anzeige führen. Auch sollte man nicht mit SLEEP 50 eine Sekunde warten, um dann das Datum neu anzuzeigen. Zwar aktualisiert sich dann wie gewünscht die Anzeige in einem Intervall von einer Sekunde, jedoch kann es sein, dass die tatsächliche Uhrzeit immer um einige Millisekunden abweicht. Am elegantesten ist eine DO-LOOP-Schleife, kombiniert mit einer IF-Abfrage, die stets prüft, ob die Uhrzeit vom Controller aktualisiert wurde.

```

' ****
,
' BASIC++ Beispiele
,
' Digitale Uhr
,
' ****
DEFINE AlteUhrzeit AS BYTE
DEFINE Jahr AS WORD
LCD.INIT
DO
    IF AlteUhrzeit <> SECOND THEN
        Jahr = YEAR + 2000
        LCD.CLEAR
        LCD.POS 1,1
        IF HOUR < 10 THEN
            LCD.PRINT „0“ & HOUR
        ELSE
            LCD.PRINT HOUR
    ENDIF
    AlteUhrzeit = SECOND
    SLEEP 50
END

```

```

END IF

LCD.PRINT „:“

IF MINUTE < 10 THEN
    LCD.PRINT „0“ & MINUTE
ELSE
    LCD.PRINT MINUTE
END IF

LCD.PRINT „:“

IF SECOND < 10 THEN
    LCD.PRINT „0“ & SECOND
ELSE
    LCD.PRINT SECOND
END IF

LCD.POS 2,1

SELECT CASE DOW

    CASE BPPSunday
        LCD.PRINT „So.“
    CASE BPPMonday
        LCD.PRINT „Mo.“
    CASE BPPTuesday
        LCD.PRINT „Di.“
    CASE BPPWednesday
        LCD.PRINT „Mi.“
    CASE BPPThursday
        LCD.PRINT „Do.“
    CASE BPPFriday
        LCD.PRINT „Fr.“
    CASE BPPSaturday
        LCD.PRINT „Sa.“

END SELECT

LCD.PRINT „ „ & DAY & „.“ & MONTH
LCD.PRINT „.“ & Jahr

AlteUhrzeit = SECOND

END IF
LOOP
END

```

**Beispiel 5.4:** Formatierte LCD-Uhrzeit und Datumsausgabe

Damit die Ausgabe auf dem LC-Display auch gut lesbar ist, sollten Sie zusätzlich die Ausgabe der Stunde, Minute und Sekunde formatieren. Achten Sie dabei darauf, dass der Betrag dieser Systemvariablen auch kleiner als 10 sein kann. Das würde dazu führen, dass nur eine einstellige Zahl ausgegeben wird.

Besser wäre hier aber eine vorangestellte 0. Neben der Methode in Beispiel 5.4 können Sie eine eigene Lösung entwickeln.

Vielleicht wundern Sie sich, wie man mit dem LCD-Objekt umgeht, das in dem Beispiel benutzt wird, um auf einfache Art und Weise Schreibeoptionen für das LC-Display auszuführen. Diese *Extended-Objekte* wurden eingeführt, um Anwenden den Umgang mit den Extended-Funktionen einfacher zu gestalten.

Zur Konzeption einer LCD-Uhr reicht in erster Linie, dass Sie mit der Funktion *LCD.INIT* das LCD-Objekt initialisieren. Mit *LCD.PRINT* kann das LC-Display beschrieben werden. Mit *LCD.CLEAR* wird der Text auf dem Display gelöscht.

Achten Sie darauf, dass Sie nach Gebrauch des Objektes mit *LCD.OFF* dieses auch wieder terminieren. Weitere Informationen über das LCD-Objekt und die übrigen Extended-Functions lernen Sie im Kapitel „Erweiterungsmodule und Extended Functions“ kennen.

## 5.2 Das DCF-77 Funkuhrmodul

Wie schon bei der Vorgängerversion kann man auch an die neue Generation der C-Control eine DCF77-Antenne anschließen, um die Zeitsignale des Funksenders in der Nähe von Frankfurt zu empfangen.

Die Signale bezieht der Sender von der Atomuhr der Physikalisch-Technischen Bundesanstalt in Braunschweig.

Für die Übertragung wird ein codiertes Langwellen-Signal mit einer Frequenz von 77,5 kHz verwendet. Abbildung 5.1 verdeutlicht den Aufbau des DCF77-Signals. Die jeweiligen Informationen über die aktuelle Stunde oder den aktuellen Wochentag werden in Bitmustern bestimmter Längen abgegeben. Das gesamte Signal wird innerhalb einer Minute übergeben. Insgesamt werden so 60 Bits gesendet, wobei ein kurzer Impuls für eine Null und ein langer Impuls für eine Eins steht. Um das Ende des gesamten Signals zu markieren, wird das 60. Bit nicht gesendet.

Einige Bits haben noch eine spezielle Bedeutung. Die im Bild verwendeten P1-, P2- und P3-Markierungen sind jeweils Prüfbits. Das M-Bit ist eine Minutenmarke.

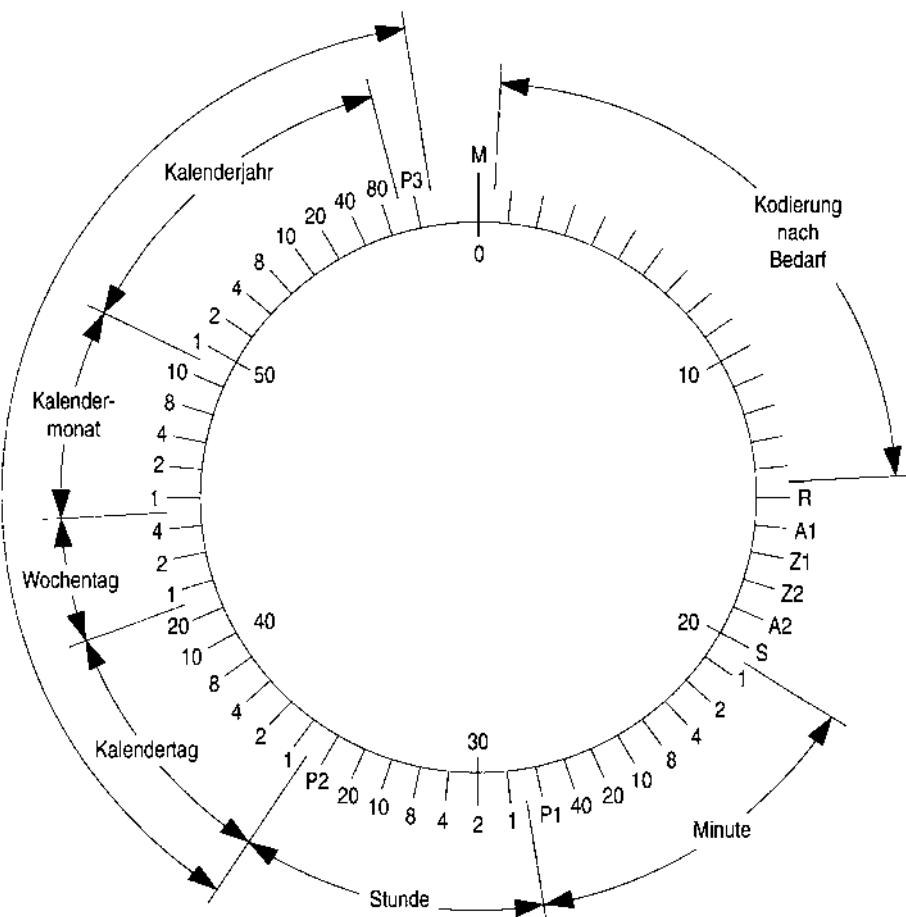


Abb. 5.1: Codierschema des DCF77-Signals

Der Vorteil des Langwellensenders ist die Erreichbarkeit in nahezu allen Teilen des Bundesgebietes. Selbst innerhalb von Gebäuden lässt sich das Signal noch gut herausfiltern und verarbeiten. Den Vorteil des DCF77-Signals nutzen daher selbst viele Schaltungen im professionellen und semiprofessionellen Bereich.

Zum Zubehör der C-Control gehört auch die DCF-Empfängerplatine. Auf dem Application Board 2.0 befindet sich links unten neben der Tastaturmatrix die Anschlussmöglichkeit der DCF-Platine. Nach Bestückung des Boards mit der Platine versucht der Controller das DCF77-Signal herauszufiltern und mit der Systemzeit zu synchronisieren. Sie müssen sich also keine Gedanken um die Umsetzung des Codierschemas machen. Dieser Vorgang dauert in der Regel nur einige Minuten bis wenige Sekunden. Trotz der allgemein guten Ausstrahlung des Lang-

wellen-Signals kann es zu Problemen beim Empfang des Codierschemas kommen. Besonders Spulen, Magnete und starke elektrische Felder (zum Beispiel Elektromotoren, Lautsprecher oder Netzteile) können zu Störungen des Empfangs führen. Ohne weiteres Zutun können Sie so mit dem Code aus Beispiel 5.4 eine einfache digitale LCD-Funkuhr programmieren. Eine kleine Erweiterung des LCD-Funkuhr-Codes könnte der Einbau einer Weckfunktion darstellen.

```

' ****
,
' BASIC++ Beispiele
,
' Digitale Uhr mit Weckfunktion
,
' ****
TYPE tUhrzeit
    Stunde AS BYTE
    _Minute AS BYTE
    Sekunde AS BYTE
END TYPE
DEFINE AlteUhrzeit AS BYTE
DEFINE Jahr AS WORD
DEFINE Weckzeit AS tUhrzeit
DEFINE i AS BYTE
Weckzeit.Stunde = 23
Weckzeit._Minute = 30
Weckzeit.Sekunde = 0
LCD.INIT
DO
    IF Weckzeit.Stunde = HOUR THEN
        IF Weckzeit._Minute = MINUTE THEN
            IF Weckzeit.Sekunde = SECOND THEN
                FOR i = 1 TO 5
                    BEEP 50, 50, 10
                NEXT i
            END IF
        END IF
    END IF
    IF AlteUhrzeit <> SECOND THEN
        AlteUhrzeit = SECOND
        Jahr = YEAR + 2000
        LCD.CLEAR
        LCD.POS 1,1
        IF HOUR < 10 THEN
            LCD.PRINT „0“ & HOUR
        ELSE

```

```

LCD.PRINT HOUR
END IF

LCD.PRINT „:“

IF MINUTE < 10 THEN
    LCD.PRINT „0“ & MINUTE
ELSE
    LCD.PRINT MINUTE
END IF

LCD.PRINT „:“

IF SECOND < 10 THEN
    LCD.PRINT „0“ & SECOND
ELSE
    LCD.PRINT SECOND
END IF

LCD.POS 2,1

SELECT CASE DOW

    CASE BPPSunday
        LCD.PRINT „So.“
    CASE BPPMonday
        LCD.PRINT „Mo.“
    CASE BPPTuesday
        LCD.PRINT „Di.“
    CASE BPPWednesday
        LCD.PRINT „Mi.“
    CASE BPPThursday
        LCD.PRINT „Do.“
    CASE BPPFriday
        LCD.PRINT „Fr.“
    CASE BPPSaturday
        LCD.PRINT „Sa.“

END SELECT

LCD.PRINT „ „ & DAY & „.“ & MONTH & „.“
LCD.PRINT Jahr

END IF
LOOP
LCD.OFF
END

```

**Beispiel 5.5:** LCD-Uhr mit Weckfunktion

Neben den bekannten Programmteilen zur formatierten Ausgabe der Uhrzeit und des Datums im LC-Display wurde der Code noch über eine Weckfunktion erweitert, die bei jedem Schleifendurchlauf die Uhrzeit mit der programmierten Weckzeit vergleicht. Die hier voreingestellten Werte der Weckzeit liegen in einem Objektdatentyp vor. Bei der eingestellten Uhrzeit meldet sich die C-Control mit vier schrillen Tonsignalen. Diese werden mittels des BEEP-Befehls und des internen Rechteck-Tongenerators erzeugt. Die IF-Abfragen zum Vergleich der Uhr- und Weckzeit könnten natürlich auch durch eine einzelne IF-Abfrage durch Verknüpfungen mit dem AND-Operator ersetzt werden.

## 5.3 Zeit- und datumsgesteuerte Programme

Eine weitere Möglichkeit wäre es, den Code dahingehend zu erweitern, dass der Controller auf bestimmte Wochentage, Monate oder Jahreszeiten reagiert. Nützlich wäre dies beispielsweise für die Programmierung einer automatischen Schaltuhr der Lichtanlage für Heim oder Garten. Ähnlich wie bei der Uhrzeitabfrage müssten Sie Kontrollmechanismen einbauen, um das aktuelle Datum mit eingestellten Benutzerwerten zu vergleichen.

Bei datumsgesteuerten Ereignissen kombiniert mit Uhrzeiten könnte die Umstellung der Winter- auf Sommerzeit jedoch problematisch sein. Mit reiner Schaltungstechnik, ohne einen programmierbaren Mikrocontroller wie der C-Control, wäre die Lösung des Sommer- und Winterzeiten-Problems fast unmöglich.

Die Zeitumstellung ist innerhalb der EU einheitlich geregelt. Im Zeitgesetz vom 25. Juli 1978 und dessen geänderter Fassung vom September 1994 findet sich eine klare Regelung, wann auf Sommer- oder Winterzeit (die eigentliche Normalzeit) umgestellt werden soll. So findet die Zeitumstellung zur Sommerzeit am letzten Sonntag im März um 2:00 Uhr statt. Dabei wird die Uhr um eine Stunde vorge stellt. Die Winterzeit beginnt ab dem letzten Sonntag im Oktober um 3:00 Uhr. Hier wird die Uhrzeit um eine Stunde zurückgestellt.

```
' ****
',
' BASIC++ Beispiele
',
' Funktionen zur Bestimmung der
' Sommer- und Winterzeit
',
' ****
CONST cnstSommerzeit = 1
CONST cnstWinterzeit = 2
DEFINE AktuelleZeit AS BYTE
```

```

FUNCTION WinterSommerZeit()
  IF MONTH = 3 THEN
    IF DOW = BPPSunday THEN
      IF DAY > (31 - 7) THEN
        IF HOUR = 2 THEN HOUR = 3
        AktuelleZeit = cnstSommerzeit
      END IF
    END IF
  END IF
  IF MONTH = 10 THEN
    IF DOW = BPPSunday THEN
      IF DAY > (31 - 7) THEN
        IF AktuelleZeit = cnstSommerzeit THEN
          IF HOUR = 3 THEN
            HOUR = 2
            AktuelleZeit = cnstWinterzeit
          END IF
        END IF
      END IF
    END IF
  END IF
END FUNCTION

```

**Beispiel 5.6:** Unterprogramm zur Zeitumstellung

Mit der Funktion aus Beispiel 5.6 wird die Uhrzeit automatisch auf Winter- oder Sommerzeit umgestellt. In der Variablen *AktuelleZeit* wird dabei die momentane Zeiteinstellung zwischengespeichert. Zunächst wird in der Funktion geprüft, ob das Datum der Echtzeituhr auf den Monat März eingestellt wird. Da die Uhrumstellung nur an einem Sonntag erfolgt, werden alle anderen Wochentage mit einer weiteren IF-Abfrage ignoriert. Natürlich gibt es nicht nur einen Sonntag im März. Um den letzten Sonntag auszurechnen, wird geprüft, ob die Systemvariable DAY größer als 24 ist. Damit wäre gesichert, dass es sich um die letzte Woche im Monat (24+7=31) handelt. Die Zeitumstellung erfolgt dann um 2:00 Uhr morgens.

Wenn in der Variablen MONTH nun der Wert für den Oktober enthalten ist, wird wie bei der Sommerzeit geprüft, ob das aktuelle Datum dem letzten Sonntag im Monat entspricht. Um 3:00 Uhr soll dann eine Stunde zurückgestellt werden. Dies könnte zu einem Problem führen, da die Stunde zweimal für diesen Tag durchlaufen werden würde. Um dieses Problem zu umgehen, muss sich das Programm merken, dass die Umstellung auf Winterzeit bereits erfolgt ist. Hierfür wird die schon erwähnte Variable *AktuelleZeit* verwendet.

## 5.4 Zeitmessung

Ohne die Extended-Functions oder externe Bauelemente zu verwenden, können Sie mit dem System-Timer der C-Control eine einfache Zeitmessung im Rahmen einer Genauigkeit von 20 ms vornehmen. Für viele Anwendungsbereiche der Zeitmessung reicht dieser Wert schon aus. Um kürzere Intervalle messen zu können benötigen Sie jedoch Frequenzmesser oder Ereigniszähler.

### 5.4.1 Grobe Zeitmessung

Um Ihnen ein Beispiel der Zeitmessung mit dem System-Timer zu geben, soll die Code-Ausführgeschwindigkeit der neuen M-Unit 2.0 mit der Geschwindigkeit der alten verglichen werden. Ein praxisnaher Vergleich wäre eine simple For-Schleife (Beispiel 5.7), wobei dann die Systemtimer-Ticks der beiden Controller vor und nach der Schleife ausgegeben werden sollen.

```

' ****
,
' BASIC++ Beispiele
,
' Geschwindigkeitsvergleich
' Alte und neue M-Unit
,
' ****
DEFINE i AS WORD
DEFINE t1 AS WORD
DEFINE t2 AS WORD
CONST Limit = 100
DO
  t1 = Timer
  FOR i = 1 TO 100
  NEXT i
  t2 = Timer
  PRINT t1 & ":" & t2
LOOP

```

**Beispiel 5.7:** Geschwindigkeitsvergleich neue und alte M-Unit

Misst man die Geschwindigkeit mit den Einstellungen aus dem obigen Programm, erhält man für die M-Unit 2.0 eine nicht messbare Ausführgeschwindigkeit, da die Ausführung der Schleife in weniger als 20 ms erfolgt. Bei der alten M-Unit dagegen erhält man schon einen deutlich messbaren Zeitunterschied, der sich in folgender Tabelle widerspiegelt:

**Tabelle 5.1:** Gemessene Zeiten vor- und nach dem Schleifendurchlauf

Vor Schleifendurchlauf	Nach Schleifendurchlauf
294	308
309	322
328	341
343	356
358	371
373	386
388	401
403	416
418	431
433	446

Bildet man einen Durchschnitt der Zeitdifferenzen aus Tabelle 5.1, erhält man im Schnitt etwa 13 Systemticks pro Schleifendurchlauf. Das sind damit immerhin 260 ms für eine For-Schleife von 1 bis 100. Ändert man bei beiden Controllern die For-Schleife auf 1 bis 1.000 ab, erhält man eine Zeitdifferenz der alten Unit von 2,72 s und auf der neuen einen immerhin messbaren Unterschied von 5 Systemticks, also 100 ms. Beim maximalen Limit von 32.767 Schleifendurchläufen kommt man bei der alten Unit auf rund 4.450 Systemticks, also etwa 89 s. Auch beim neuen Controller kommt man nun auf ein deutlich größeres Ergebnis von 3,4 s.

Für einfache Messungen von Zeitdifferenzen reicht also der System-Timer aus. Werden jedoch Messungen mit Intervallen kleiner als 20 ms benötigt, kommt man mit dem einfachen Zähler nicht aus. Als mögliche Alternativen bieten sich hier die Frequenzeingänge Freq1 und Freq2 an, auf die im Verlauf des Buchs noch eingegangen wird.

#### 5.4.2 Zeitmessung mit hoher Auflösung

Zu den *Extended Functions* der neuen M-Unit gehört auch das CONFIG-Modul. Mit den acht Bits des Config-Registers kann man verschiedene Einstellungen des Controllers verändern. So kann man die beiden Frequenzeingänge, auf die das folgende Unterkapitel näher eingeht, in einen speziellen *Ereignismodus* schalten.

```
, ****
,
' BASIC++ Beispiele
,
' Ereigniszähler
,
' ****
DEFINE LetzterWert AS WORD
DEFINE NeuerWert AS WORD
DEFINE x AS BYTE
DA[1] = 200 ' Digitaler Ausgang
CONFIG.INIT
CONFIG.GET x
x = (x OR 00000110b)
CONFIG.PUT x
CONFIG.OFF

WITH LCD
    .INIT
    .POS 1,1
    .PRINT „DA[1] mit FREQ2“
    .POS 2,1
    .PRINT „verbinden!“

FREQ2 = 0

DO

    NeuerWert = FREQ2

    IF LetzterWert <> NeuerWert THEN
        LetzterWert = NeuerWert

        LCD.CLEAR
        LCD.PRINT „Impulse: „ & NeuerWert

    END IF

    LOOP

    .OFF
END WITH
```

**Beispiel 5.8:** Vom Frequenz- zum Ereigniszähler

Das Beispiel 5.8 wurde nun um die Zeilen zwischen `DA[1] = 200` und `WITH LCD` erweitert. Zunächst wird das CONFIG-Objekt, ähnlich wie das LCD-Objekt, mit dem Befehl INIT initialisiert. Als Nächstes wird die Konfiguration in einer Byte-Variablen zwischengespeichert, ansonsten würde die übrige Konfiguration ungewollt überschrieben werden. Durch die bitweise ODER-(OR-)Verknüpfung (siehe auch Kap. 2) setzt das Programm gezielt das zweite und dritte Bit der Variablen. Die anderen Bits bleiben im Ursprungszustand und werden demnach nicht verändert. Nun muss die Config-Variablen nur noch mit PUT aktualisiert werden. Jetzt befinden sich beide Frequenzeingänge im Ereignismodus.

## 5.5 Frequenzmessung

Die beiden Frequenzeingänge FREQ 1 und FREQ 2 können zur Messung von Impulsen und Ereignissen verwendet werden, wobei beide Frequenzeingänge bis zu 32 kHz (Frequenzen von 32 kHz bis 65 kHz erkennt man durch negative Frequenzwerte) darstellen können. Das Prinzip basiert auf dem System der Impulszählung. Neben einem Impulszähler benötigt ein solcher Frequenzeingang auch einen Zeitintervall, in dem die Impulse gezählt werden. Dieses Intervall bezeichnet man auch als Torzeit. Die im Betriebssystem voreingestellte Torzeit der C-Control beträgt eine Sekunde und kann nicht geändert werden.

Laut Bedienungsanleitung des Controllers liegt der Messfehler bei einer Frequenz bis zu 5 kHz bei rund 1 % Ungenauigkeit. Danach wird die Messgenauigkeit immer größer. Bis etwa 5 % Ungenauigkeit sind Messergebnisse in den meisten praktischen Anwendungsfällen brauchbar. Sofern Sie die DCF77-Empfängerplatine an das Application-Board angeschlossen haben, ist jedoch FREQ 1 schon belegt.

### 5.5.1 Einfache Frequenzmessung

Als erste Anwendung zum Thema Frequenzeingänge soll die C-Control sich selbst messen. Ziel des Beispiels ist es, die Frequenz an einem digitalen Ausgang oder mit FREQ1 oder FREQ2 zu messen. Das Ergebnis soll der Einfachheit halber auf dem LC-Display ausgegeben werden. Eine andere Ausgabe oder Verarbeitung der Werte ist natürlich möglich. Sie sollten in jedem Fall aber nicht immer auf den Frequenzeingang zurückgreifen, sondern dessen Wert in einer Variablen speichern. Andernfalls könnte es sein, dass sich dieser in der Zwischenzeit geändert hat.

```

', ****
',
' BASIC++ Beispiele
',
' Frequenzmessung
',
', ****
DEFINE LetzterWert AS WORD
DEFINE NeuerWert AS WORD
DA[1] = 200 ' Digitaler Ausgang
WITH LCD
    .INIT
    .POS 1,1
    .PRINT „DA[1] mit FREQ“
    .POS 2,1
    .PRINT „verbinden!“
DO
    NeuerWert = FREQ
    IF LetzterWert <> NeuerWert THEN
        LetzterWert = NeuerWert
        LCD.CLEAR
        LCD.PRINT „FREQ: „ & NeuerWert & „ Hz“
    END IF
LOOP
.OFF
END WITH
END

```

### Beispiel 5.9: Einfache Frequenzmessung mit FREQ

Die interne Variable FREQ wird nicht deklariert, da sie bereits vorgegeben ist. Sie können das Beispiel 5.9 auch so umändern, dass der Controller nicht die Werte von FREQ, sondern FREQ2 zurückgibt. Um das Beispiel zu testen, müssen Sie den digitalen Ausgang 1 mit dem jeweiligen Frequenzeingang über ein Kabel verbinden. Bei beiden Frequenzeingängen müsste auf dem Display 1930 erscheinen. Der Wert sollte sich innerhalb weniger Sekunden einstellen, kann allerdings auch im Bereich von etwa  $\pm 100$  schwanken. Dies würde einer Ungenauigkeit des Ergebnisses von etwa 5 % bedeuten.

Die auf dem Display stehende Zahl bedeutet, dass die Signale vom digitalen Ausgang mit einer Frequenz von 1.930 Hz ausgegeben werden. Allgemein verwendet

man den Begriff der Frequenz, um die Anzahl von Ereignissen in einer bestimmten Zeitperiode wiederzugeben. 1 Hz entspricht einem Impuls pro Periode mit einer Periodenlänge von 1 Sekunde. Der digitale Ausgang gibt also kein wirkliches analoges Signal ab, sondern nur sehr kurze Spannungsimpulse.

### 5.5.2 Frequenzmessung von 32 bis 65 kHz

Ohne besondere Vorschaltungen lassen sich mit der C-Control auch Frequenzen über 32 kHz messen. Eine Frequenz ist eine 16-Bit-breite Integerzahl. Aufgrund des Vorzeichens kann man eine Frequenz bis 32.767 Hz messen. Frequenzen von 32.768 bis 65.535 Hz stellen sich als negative Zahl da. Sinnvoll wäre es auch, diese Frequenzen als positive Frequenzen darstellen zu können. Die C-Control ist jedoch von sich aus nicht in der Lage, positive Zahlen über 32.767 darzustellen.

```

' ****
,
' BASIC++ Beispiele
,
' 0 ... 65536 Hz Frequenzmessung
,
' ****
DEFINE Frequenz AS WORD
WITH LCD
    .INIT
    .CLEAR
    .PRINT „Warte...“
DO

    IF Frequenz <> FREQ2 THEN
        .CLEAR
        Frequenz = FREQ2
        IF Frequenz >= 0 THEN
            .PRINT Frequenz & „ Hz“
        ELSE
            SchreibeZahl(Frequenz)
            .PRINT „ Hz“
        END IF
    END IF
LOOP
.OFF
END WITH
FUNCTION SchreibeZahl(x AS WORD)
    DEFINE ObereZiffern AS WORD
    DEFINE UntereZiffer AS WORD
    ' Obere vier Stellen:

```

```

    ' Zahl um ein Bit nach rech
    ' verschieben; entspricht Division durch 2
    ObereZiffern = x SHR 1
    ' Nur die unteren 15 Bit betrachten
    ObereZiffern = ObereZiffern AND 7FFFh
    ' Durch 10 teilen (2 * 5 = 10)
    ObereZiffern = ObereZiffern / 5

    ' Vorzeichen-Bit entfernen und untere Ziffer ermitteln:

    UntereZiffer = x AND 7FFFh
    UntereZiffer = UntereZiffer MOD 10
    UntereZiffer = (UntereZiffer + 12h) MOD 10

    ' Alternativ auch platzsparender:
    ' ObereZiffern = ((x SHR 1) AND 7FFFh) / 5
    ' UntereZiffer = ((x AND 7FFh) MOD 10 + 12h) MOD 10

    LCD.PRINT ObereZiffern & UntereZiffer

END FUNCTION

```

**Beispiel 5.10:** Frequenzen bis 65.535 darstellen

Das Beispiel 5.10 belegt 163 Byte Basic-Speicher, was bei fast 10 kByte Flash-Speicher noch im Rahmen des Verträglichen ist. Allerdings wurde zur besseren Übersicht auf Codeoptimierung verzichtet. Wenn Sie die Zeilen zwischen „Obere vier Stellen:“ bis „Alternativ auch platzsparender:“ auskommentieren und stattdessen die Kommentare vor der alternativen Berechnung entfernen, sparen Sie weitere 16 Byte.

### 5.5.3 Messung hochfrequenter Rechtecksignale

Höhere Frequenzen als 65 kHz kann man aber ohne Vorschaltung mit der C-Control nicht mehr messen. Abhilfe schaffen Frequenzteiler. Sie sind in der Lage, Frequenzen rechteckförmiger Signale (bei sinusförmigen Signalen sollte man besser keine Frequenzteiler verwenden) in einem bestimmten Verhältnis herunterzuteilen. Vom Prinzip her bestehen Frequenzteiler aus T-Flip-Flops. Durch jedes Flip-Flop wird die Frequenz im Verhältnis 2:1 geteilt. Bei 14 Flip-Flops teilt man also die Frequenz durch  $2^{14} = 16384$ . Somit wäre man theoretisch in der Lage eine Frequenz von 535 MHz mit einem 14-stufigen Frequenzteiler mit der C-Control zu messen. Da aber auch der Frequenzteiler eine physikalische Grenze hat, ist man meist nur in der Lage Frequenzen bis rund 30 MHz zu messen.

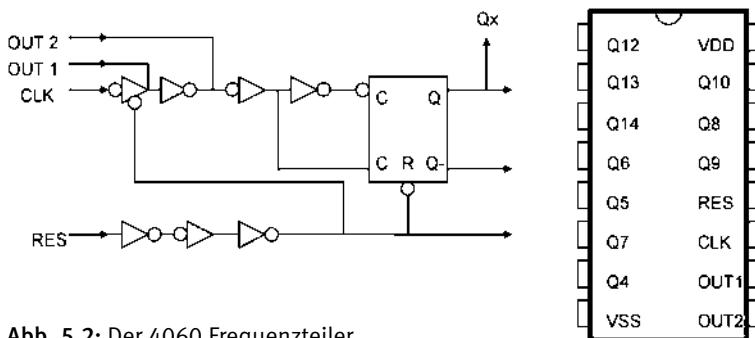


Abb. 5.2: Der 4060 Frequenzteiler

Ein typisches Beispiel für einen Frequenzteiler ist der 14-stufige MC14060B (oft auch 4060 genannt). Er unterstützt Frequenzen bis etwa 5 MHz. An den zehn Ausgängen kann man die Frequenz dann in einem bestimmten Teilverhältnis abgreifen.

Der IC 4060 kann mit der Spannung der C-Control betrieben werden. Achten Sie aber besonders auf die Polung. Im Praxistest zeigte sich, dass der Chip sehr anfällig gegen falsche Polung ist. Auf die Ausgänge 1 bis 3 wurde verzichtet, sodass das kleinste Teilungsverhältnis 1:16 ist.

# Sachverzeichnis

2-Wire-Bus 152

433-MHz-Band 139, 143

## A

Abbruchbedingung 35, 36

Acknowledge-Bit 153

ADC-Frequenz 110

Adressierung 181

Analogports 13, 31

AND 39, 40

Arbeitsspeicher 30, 41, 50, 149

Assembler 13, 21, 64

Autostart-Jumper 21, 24, 25

*Autostart-Modus* 24

## B

BASIC-RAM 172

Beep 17

Bit-Variablen 31

Bogenmaß 55, 56

BOOT-Option 151

Bustakt 13

Byte-Port 31

Byte-Variablen 31

## C

Chip-BOOT-Option 151

Chipkarte 149, 151

Code-Explorer 29

Code-Fenster 29

Code-Segment 170, 171

Condition-Code-Register 173, 178

CONFIG2-Register 136

Config-Objekt 133

## D

Dateihandling 16

Datenrichtung 64, 66

Datensegment 170

DCF77-Antenne 85

DCF-Synchronisation 16

Debugging 60

Digitalport 13, 31

Duty-Cycle 139

DYNAMIC 51

## E

Echtzeituhr 80, 81

EEPROM 149, 150, 151, 156, 157, 158

EEPROM-Objekt 106

Endlosschleife 36, 179

Ereignismodus 92

Extended-Funktion 21

Extended-Register 174

## F

Flash-Speicher 19, 105

Fließkomma-Arithmetik 19, 41, 51

FLOAT 51

Floatingpoint-Arithmetik 19

FPPRINT 52

Frequenzteiler 97

## G

GET 58, 59

globale Variable 30

GOSUB-Befehl 27

## H

H-12-Format 139

**I**

I<sup>2</sup>C-Bus 13, 21, 152, 153, 153, 153, 153, 154,  
 156, 156, 157, 166, 167  
 I<sup>2</sup>C-Datenspeicher 151  
 Index-Registerpaar 172  
 Internet Update 22  
 Interrupeingang 13  
 IRQ 57, 136

**K**

Konstante 32, 171  
 Kosinus-Funktion 55

**L**

Label 27  
 Laufzeit 50, 61  
 Laufzeitfehler 48  
 LC-Display 13, 21, 52  
 lokale Variable 38  
 LSA 174  
 LSB 167

**M**

MAP-Datei 41, 177  
 Master-Slave 153  
 MAX-Funktion 18  
 Meldungsfenster 29  
 Memloc 166, 167  
 Memory-Map 30  
 MIN-Funktion 18  
 MSB 163

**N**

NOT 39, 40  
 Nullmodem-Kabel 23, 24

**O**

Offset 31, 43, 50, 111, 167  
 ^-Operator 45  
 @-Operator 45  
 OR 39, 40

**P**

PAGE0 169, 170  
 PAGE1 169, 170  
 PageWrite 158  
 Parameter 38  
 PCB-Format 15  
 PD-Regler 127, 128  
 PID-Regler 127  
 PI-Regler 127, 128, 130, 132  
 Pointer 44  
 Porterweiterung 155  
 P-Regler 127  
 Projekt-Manager 29  
 pulsweitenmoduliertes Signal 121

**R**

RAM-Speicher 19  
 RC5-Protokoll 144  
 RC-Glied 122  
 Register A 172  
 Register H 172  
 Rekursionen 50  
 Relais 13, 76, 77  
 RollOver 159  
 RS232 13

**S**

Sample-and-Hold-Schaltung 109  
 SCL 152, 153, 153, 156, 159  
 SDA 152, 153, 156, 159  
 serielle Ringleitung 104  
 serielle Schnittstelle 13, 47, 99  
 Servo Ansteuerung 16  
 SERVO-Modus 134  
 SET 58, 59  
 SHL 39  
 SHR 39  
 Sinus-Funktion 55  
 Spannungsregler 15  
 Speicher-Map 30  
 Sprungbefehle 179  
 SSA 174  
 Stacktiefe 53  
 Startbedingung 36

Stoppbedingung 153

SYS-Call 171

SYSCODE 175

System-Timer 91, 92, 136

Systemtreiber 16, 21, 166, 169

## T

Tabelle 117

Tastatur 13

Terminierungszeichen 46

Tieffpass 122

Time-out 100, 103

TIMER INTERRUPT 136

Timer 80

Token 23

Tokencode 13, 61

Torzeit 94

Transistorschaltung 74

## U

Unterprogramme 27

USB-Modul 99

USB-Steckplatz 24

## V

Vorwiderstand 67

## W

Winkel 55

Winkelmaß 56

WORD-Datentyp 31

WORD-Port 31

WriteProtected 159

## X

XOR 39, 40

## Z

Zeichenketten 46

Zeigeroperatoren 44



Stefan Tappertzhofen

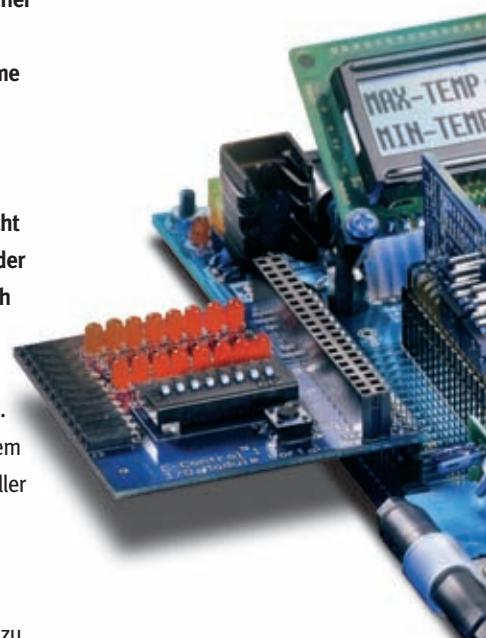
## Messen, Steuern und Regeln mit C-Control M-Unit 2

Mikrocontroller sind im Automatisierungsbereich tonangebend. Wo früher teure und komplex aufgebaute Schaltungen verwendet wurden, sind heute leistungsstarke und extrem preisgünstige Mikrocontrollersysteme im Einsatz. Die breite Produktpalette bietet dabei Lösungen für den privaten, aber auch professionellen und industriellen MSR-Bereich an. Ein wichtiges Mikrocontroller-Auswahlkriterium ist neben dem Preis-Leistungs-Verhältnis auch die erforderliche Einarbeitungszeit. Hier sticht besonders das seit Jahren bewährte C-Control-Mikrocontrollersystem der Firma Conrad Electronic hervor. In der Generation 2.0 präsentiert es sich hinsichtlich Hard- und Software nun mit einer geradezu unglaublichen Leistungsvielfalt.

Das vorliegende Buch wurde komplett überarbeitet und neu strukturiert. Der Anwender lernt neben der neuen M-Unit, der C-Control Micro und dem umfangreichen Zubehör auch die leistungsstarken Advanced Mikrocontroller für den professionellen Einsatz kennen. Die praxisorientierten Beispiele führen nicht nur in die Programmierung mit der einfachen und flexiblen Programmiersprache BASIC++ 2006 ein, sondern vermitteln auch die nötigen Erfahrungen, um sich den Herausforderungen der MSR-Technik zu stellen. Die auf der CD-ROM beigelegten Anwendungsprogramme und Demoapplikationen runden den Einstieg in die Welt der Mikrocontrollersteuerungen ab.

### Aus dem Inhalt:

- Einführung in BASIC++ 2006 und CCASM
- Digital- und Analogports
- Zeitsteuerung
- Messschaltungen
- Die serielle Schnittstelle
- Extended Module wie LCD-, I<sup>2</sup>C- oder Config-Modul
- Advanced Mikrocontroller mit 32-Bit-Fließkommaarithmetik



### Auf CD-ROM:

- Anwendungsprogramme
- Beispiele
- Datenblätter

Besuchen Sie uns im Internet: [www.franzis.de](http://www.franzis.de)

ISBN 978-3-7723-5488-5



9 783772 354885

Euro 39,95 [D]