



MODUL 6

TIMER UND COUNTER

V1.1 J. Humer 1997

INHALTSVERZEICHNIS

MODUL 6

TIMER UND COUNTER

Inhalt	Seite
1 Die Timer und Counter der Familie 8051	3
1.1 Timer Anwendung: Zeitschleife	4
1.2 Counter Anwendung: Drehzahlmessung	5
2 Die Capture Compare Unit	8
2.1 Begriffserklärung	8
2.1.1 Compare.....	8
2.1.2 Reload.....	8
2.1.3 Capture.....	8
2.2 Anwendungsbeispiel PWM	13
2.3 Musterlösung zu Übung 17	16
2.4 Lösung mit I2C Interrupt	19
2.5 Musterlösung zu Übung 18	22
2.6 Programmbeispiel LFU	26

1 Die Timer und Counter der Familie 8051

Der Mikrocontroller 8051 enthält drei universelle 16 bit Timer / Counter Blöcke: Timer0, Timer1 und Timer2, einen Capture/Compare Timer. Leistungsfähige Derivate wie der 80C517 enthalten zusätzlich eine komplette Capture Compare Einheit, welche in ihrer Leistungsfähigkeit noch um einiges beachtlicher ist als die drei Timer, welche zur Grundausstattung des 8051 gehören.

Die Timer 0 und 1 haben folgende mögliche Betriebsarten:

- **Timer Funktion:** Der Inhalt des Zählregisters des Timers wird jeden Maschinenzyklus um Eins erhöht. Da ein Maschinenzyklus 1/12 der Taktfrequenz ist, zählt der Timer/Counter mit 1/12 der Quarzoszillatorfrequenz hoch.
- **Counter Funktion:** Für diese Betriebsart steht für jeden der Counter ein Portpin (P3_4 für Counter 0 und P3_5 für Counter 1) als Eingang zur Verfügung. Wird an diesem Portpin eine Rechteckimpuls angelegt, so wird der Inhalt des Zählregisters des Timers bei jeder negativen Flanke um Eins erhöht. In dieser Betriebsart können also externe Ereignisse gezählt werden.

Zusätzlich zu dieser Auswahl lassen sich noch vier verschiedene Betriebsarten auswählen. Diese werden im special funktion register TMOD ausgewählt.

TMOD (89H), nicht bitadressierbar

MSB				LSB			
Timer 1				Timer 0			
Gate	C/T#	M1	M0	Gate	C/T#	M1	M0
Modusregister für Timer 0 und 1 (Timer Mode). Es enthält die Bits zur Einstellung der Betriebsarten für die Timer 0 und 1. In den Erläuterungen steht x für 0 oder 1, je nachdem, welcher Timer verwendet wird. Reset-Wert: 00H							
Bitsymbol	Funktion						
Gate	Wenn dieses Bit gesetzt ist, läuft der Timer nur dann, wenn er mit TRx freigegeben ist und gleichzeitig der Portpin P3.2/INT0# (Timer 0) bzw. P3.3/INT1# (Timer 1) auf High-Pegel ist.						
C/T#	Dieses Bit legt fest, ob Timer- oder Counter-Modus verwendet wird. C/T# = 0 wählt Timer- und C/T# = 1 wählt Counter-Modus.						
M0	M1	Arbeitsmodi					
0	0	THx dient als 8-bit-Timer; TLx bildet einen 5-bit-Vorteiler.					
0	1	THx und TLx bilden einen 16-bit-Timer.					
1	0	Auto-Reload-Timer (8 bit). Der Inhalt von THx wird beim Timerüberlauf nach TLx kopiert. THx selbst bleibt unverändert.					
1	1	Timer 0: Beide Register THx und TLx arbeiten als eigenständige 8-bit-Timer. TLx wird durch die Steuerbits von Timer 0, THx durch die Steuerbits von Timer 1 eingestellt. Timer 1: Timer 1 stoppt in dieser Betriebsart.					

1.1 Timer Anwendung: Zeitschleife

Ein typischer Anwendungsfall für einen Timer ist die Programmierung einer Zeitschleife, welche einen Interrupt auslöst.

Als Beispiel soll eine Zeitschleife mit Timer 0 programmiert werden, welche alle 10 ms einen Interrupt auslöst. Als Betriebsart wird die Betriebsart 1 gewählt, in welcher der Timer als 16 bit Timer arbeitet.

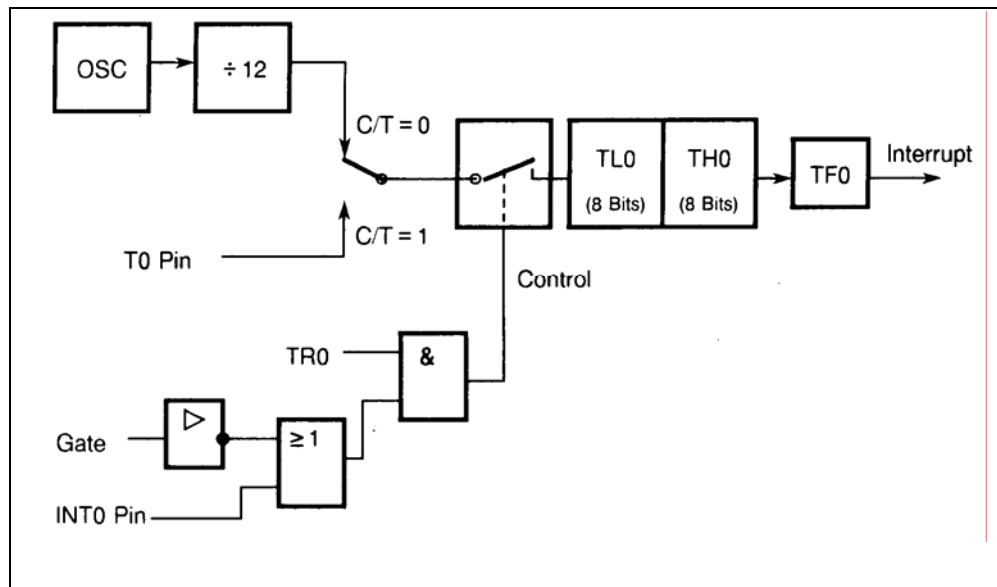


Abb. 2: Timer Betriebsart 1: 16 bit Timer

Die Quarzoszillatorfrequenz wird durch 12 geteilt. Der Zähler wird also mit 1 MHz getaktet.

Das Bit C/T des sfr TMOD wird Null gesetzt und so der Timerbetrieb gewählt. Die beiden Bits M0 und M1 werden folgendermaßen gesetzt: M0 = 1 M1 = 0. In das sfr TMOD muß also der Wert 0x01 geschrieben werden. (***TMOD = 0x01***).

Der Timer wird über das Bit TR0 eingeschaltet.


Wenn der Timer zählt, werden 256 Impulse benötigt, um das 8 bit Register TL0 einmal zum Überlaufen zu bringen. Läuft das Register TL0 über, so gibt es einen Impuls an das Register TH0 aus, in welchem dadurch der Zählerstand um Eins erhöht wird.

Das heißt, durch das Register TL0 wird die Eingangsfrequenz von 1 MHz durch 256 geteilt. Dadurch ergibt sich eine Frequenz von 3906,25 Hz. Die 10 ms, in denen die Zeitschleife einen Interrupt auslösen soll, entsprechen einer Frequenz von 100 Hz. Das Register TH0 muß also die Frequenz von 3906,25 nochmals durch 39,0625 teilen. Dies wird erreicht, indem man das Register mit (256 - 39 =) **217** vorsetzt.

Wenn ein Interrupt ausgelöst wird, soll eine Variable *msec* um Eins erhöht werden. Wenn die Variable *msec 100* erreicht hat, soll zur Anzeige, das eine Sekunde voll ist, das Bit *secflag* gesetzt werden.

Das vollständige Interruptprogramm sieht folgendermaßen aus:

```
void INTTIM0 (void) interrupt 1
{
    msec++;
    if (msec == 100)
    {
        msec = 0;
        secflag = 1;
    }
    TH0 = 217;
    TL0 = 0;
}
```

 **Übung 17:** Schreiben Sie ein Programm, mit welchem eine Uhr auf dem Board realisiert werden kann. Die Uhrzeit soll in Stunden, Minuten und Sekunden auf dem LCD Display ausgegeben werden.

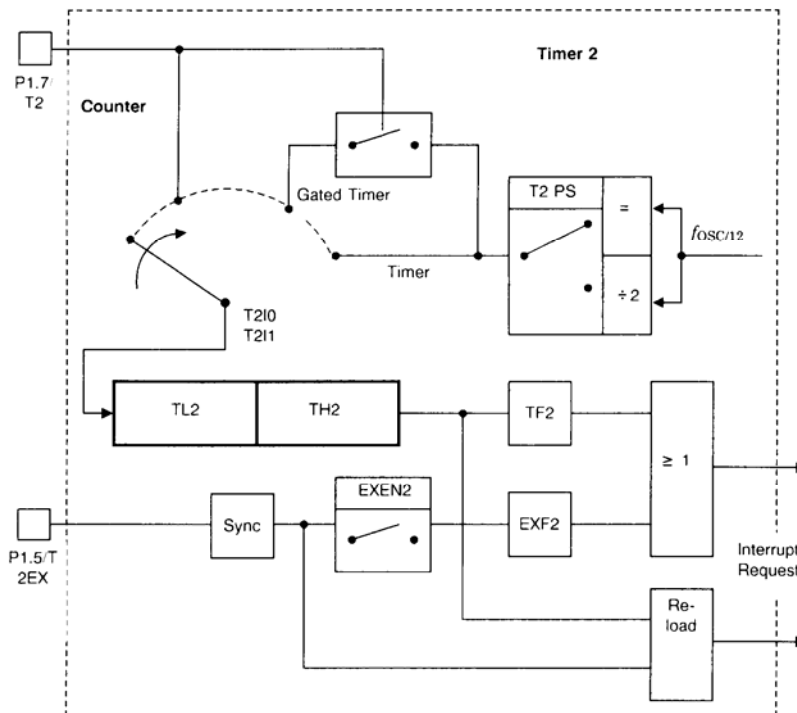
Die Stunden, Minuten und Sekunden sollen in der Interruptroutine berechnet werden und in den globalen Variablen *stu*, *min*, *sec* abgespeichert werden.

1.2 Counter Anwendung: Drehzahlmessung

/ only WIFI-BOARD */*

Mit Hilfe des Timers 2 soll die Drehzahl eines Elektromotors ermittelt werden. Dieser hat dazu einen Impulsgeber eingebaut, welcher pro Umdrehung 144 Impulse abgibt. Diese werden mit dem Zähler 2 gezählt und daraus die Drehzahl berechnet.

Der Timer/Counter 2 ist folgendermaßen aufgebaut:



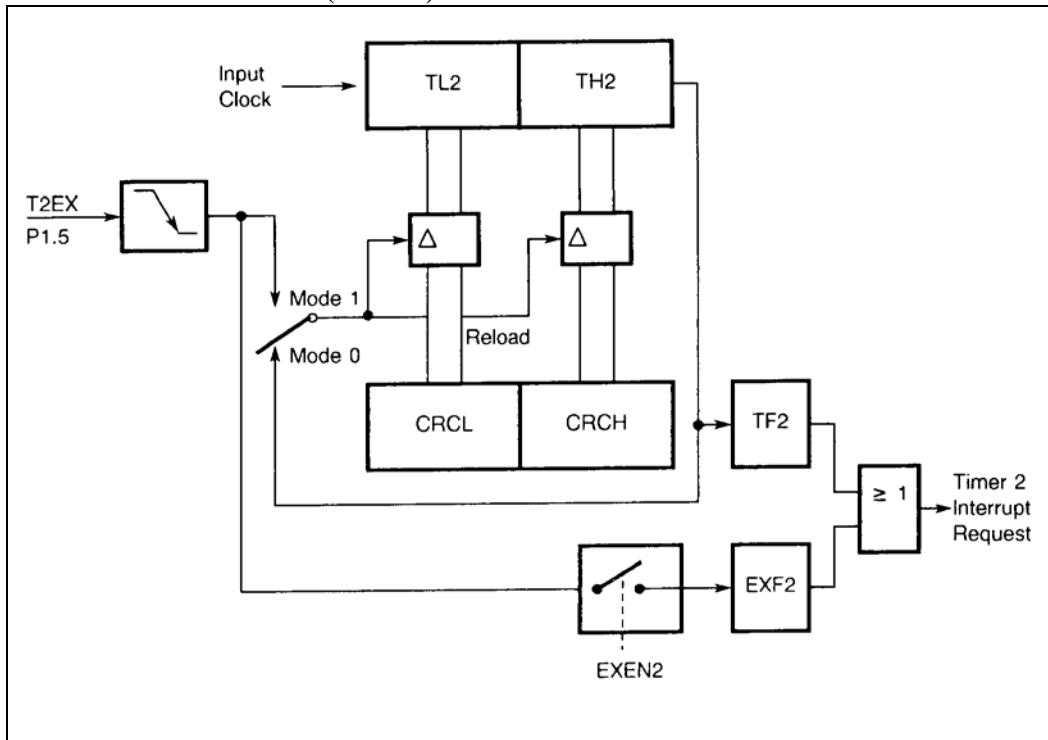
Timer 2:

T2CON (C8H), bitadressierbar

MSB							LSB	
T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0	
CFH	CEH	CDH	CCH	CBH	CAH	C9H	C8H	
Steuerregister des Timer 2 (Timer 2 Control Register). Es enthält Steuerbits für den Timer 2 sowie zwei Bits zur Interrupt-Steuerung. Reset-Wert: 00H								
Bitsymbol	Funktion							
T2PS	Timer-2-Vorteilerbit (Timer 2 Prescaler Bit). Zusammen mit T2PS1 (CTCON.7) läßt sich die Eingangsfrequenz und damit der Zähltakt des Timer 2 nach untenstehender Tabelle einstellen.							
	T2PS1 (CTCON.7)	T2PS (T2CON.7)	Eingangstakt	Zähltakt				
	0	0	f_{osz}	$f_{osz}/12$				
	0	1	$f_{osz}/2$	$f_{osz}/24$				
	1	0	$f_{osz}/4$	$f_{osz}/48$				
	1	1	$f_{osz}/8$	$f_{osz}/96$				
T2R1 T2R0	Bits zur Auswahl des Nachlademodus							
0	x	Nachladen abgeschaltet; der Timer 2 beginnt nach jedem Überlauf mit dem Wert 0000H.						
1	0	Modus 0: Autoreload nach jedem Überlauf						
1	1	Modus 1: Nachladen bei einer fallenden Flanke am Pin P1.5/T2EX						
T2CM	Auswahlbit des Compare-Modus für CRC/CC1/CC2/CC3. Für alle diese Register gleichzeitig legt es Compare-Modus 0 (T2CM = 0) oder Compare-Modus 1 (T2CM = 1) fest, wenn sie in der Compare-Funktion konfiguriert sind.							
T2I1 T2I0	Auswahlbits für die Arbeitsweise des Timers 2 nach untenstehender Tabelle.							
Taktspeisung:								
0	0	Keine Taktquelle freigegeben; der Timer 2 hält an.						
0	1	Timer 2 wird vom internen Oszillator gespeist, die Eingangsfrequenz hängt vom gewählten Vorteiler ab (Bits T2PS und T2PS1).						
1	0	Timer 2 arbeitet im Zählmodus und wird vom externen Signal am Pin P1.7/T2 getaktet.						
1	1	Timer 2 arbeitet als Gated-Timer; der intern gewählte Takt wird durchgelassen, solange am P1.7 High-Pegel anliegt.						

Bild 11-5: Special-Function-Register T2CON (C8H)

Nachladen des Timers 2 (Reload)



Die Impulse werden an Portpin T2 (P1_7) angelegt. Bei jeder negativen Flanke wird der Zählerstand des 16 Bit Zählers um Eins erhöht.

Zur Bestimmung der Drehzahl benötigt man den Timer 0, um eine 10 ms Zeitschleife auszulösen. In der Interruptroutine des Timers 0 werden die Werte der Register TH2 und TL2 ausgelesen und in den Variablen t2high und t2low abgespeichert. Anschließend müssen die Register TH2 und TL2 wieder auf Null gesetzt werden und die Zeitschleife wieder vorgesetzt werden.

Die Drehzahl berechnet sich nach folgender Formel:

$$N \text{ (U/min)} = (t2high * 256 + t2low) * 6000 / 144 = (t2high * 256 + t2low) * 41,66$$

/* only WIFI-BOARD */

2 Die Capture Compare Unit

2.1 Begriffserklärung

Im wesentlichen besteht die Compare/Capture-Einheit aus zwei Timer-Einheiten und insgesamt 13 Compare/Capture Registern. Ein Satz von 6 Steuerregistern erlaubt die Konfiguration der Compare/Capture-Einheit für die jeweilige Anwendung. Die 13 Register sind folgend eingeteilt:

- ⊗ ein 16-bit-Compare/Reload/Capture-Register (CRC)
- ⊗ drei 16-bit-Compare/Capture-Register (CC1 ... CC3)
- ⊗ ein 16-bit-Compare/Capture-Register (CC4) mit zusätzlicher Concurrent-Compare-Betriebsart.
- ⊗ acht 16-bit-Compare Register (CM0 ... CM7)

Je nach Konfiguration bieten die den beiden Timern T2 und CT (Compare Timer) angeschlossenen Register unterschiedliche Funktionen.

2.1.1 Compare

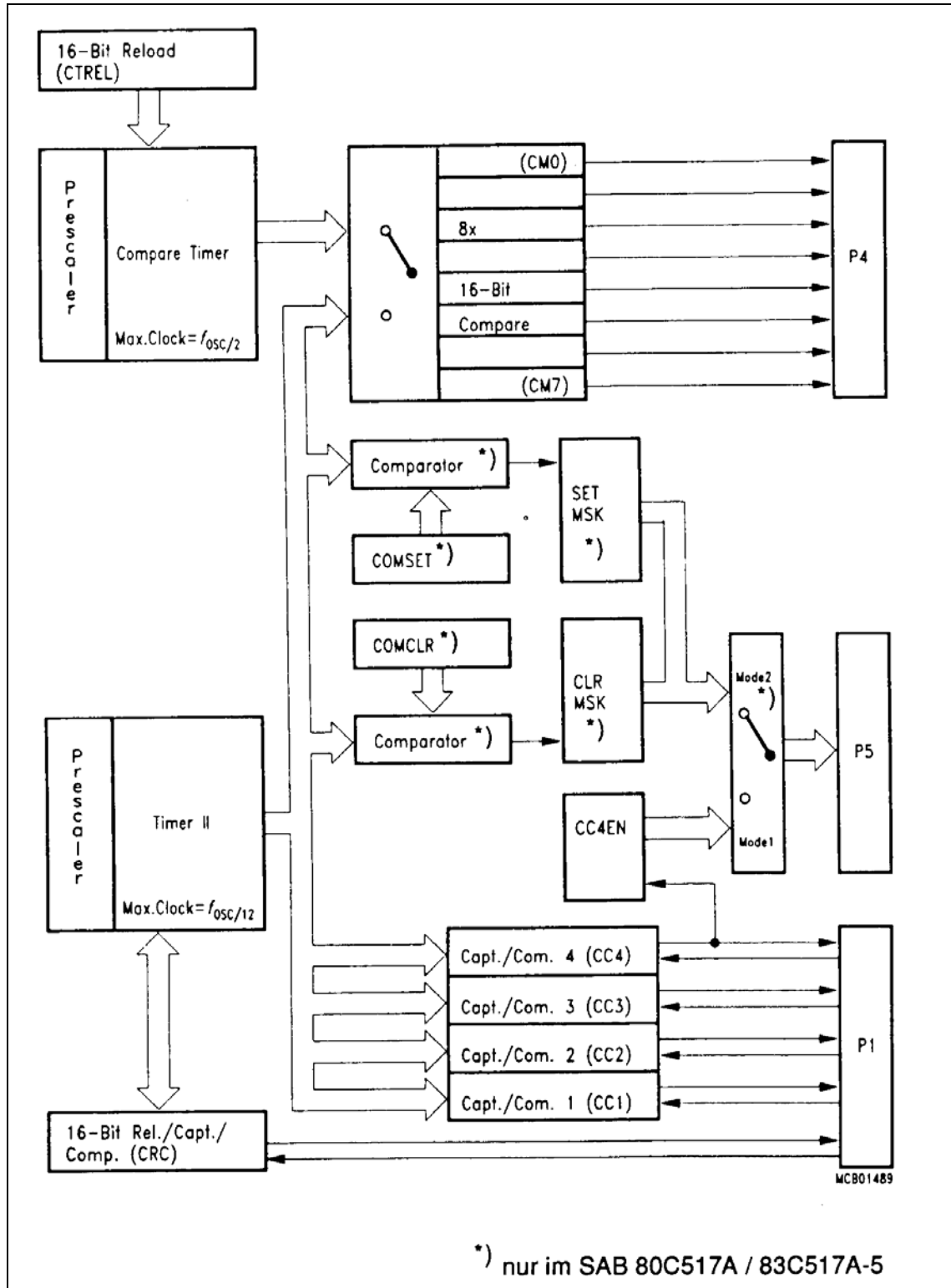
Der Inhalt des zugeordneten Timers wird ständig mit dem Inhalt des Compare-Registers verglichen. Bei Übereinstimmung werden ein oder mehrere Ereignisse (je nach Konfiguration) ausgelöst. Damit kann durch geeignete Wahl des Compare-Werts der Zeitpunkt für den Eintritt des (der) Ereignisse(s) präzise festgelegt werden.

2.1.2 Reload

Nach einem Überlauf läuft der Timer nicht mit 0000H weiter, sondern mit dem im jeweiligen Reload-Register abgelegten Nachladewert. Dadurch kann die Timer-Periode (die Zeit von Überlauf zu Überlauf) über den Reload-Wert variiert werden.

2.1.3 Capture

Auf ein bestimmtes Ereignis hin wird der Inhalt des laufenden Timers in das Capture-Register übernommen. Dadurch läßt sich der Zeitpunkt des Ereigniseintritts exakt festhalten.



Die Compare/Capture-Einheit des 80C517/80C517A

Konfigurationsmöglichkeiten der Capture/Compare-Einheit:

Timer	Compare-Register	Compare-Ausgang	Compare-Modi
Timer 2	CRCH/CRCL	P1.0/INT3#/CC0	Modus 0 und 1, Reload
	CCH1/CCL1	P1.1/INT4/CC1	Modus 0 und 1
	CCH2/CCL2	P1.2/INT5/CC2	Modus 0 und 1
	CCH3/CCL3	P1.3/INT6/CC3	Modus 0 und 1
	CCH4/CCL4	P1.4/INT2#/CC4	Modus 0 und 1
	CCH4/CCL4	P5.0/CCM0 bis P5.7/CCM7	Concurrent-Compare-Modus
	COMSET/COMCLR	P5.0/CCM0 bis P5.7/CCM7	Set/Reset-Modus (nur im 80C517A)
	CMH0/CML0	P4.0/CM0	Modus 1
	CMH1/CML1	P4.1/CM1	Modus 1
	CMH2/CML2	P4.2/CM2	Modus 1
	CMH3/CML3	P4.3/CM3	Modus 1
	CMH4/CML4	P4.4/CM4	Modus 1
	CMH5/CML5	P4.5/CM5	Modus 1
	CMH6/CML6	P4.6/CM6	Modus 1
CMH7/CML7	P4.7/CM7	Modus 1	
Compare-Timer	CMH0/CML0	P4.0/CM0	Modus 0
	CMH1/CML1	P4.1/CM1	Modus 0
	CMH2/CML2	P4.2/CM2	Modus 0
	CMH3/CML3	P4.3/CM3	Modus 0
	CMH4/CML4	P4.4/CM4	Modus 0
	CMH5/CML5	P4.5/CM5	Modus 0
	CMH6/CML6	P4.6/CM6	Modus 0
	CMH7/CML7	P4.7/CM7	Modus 0

Bild 11-9: Konfigurationsmöglichkeiten der CCU

CTCON (E1H), nicht bitadressierbar

MSB				LSB			
T2PS1	-	ICR/-	ICS/-	CTF	CLK2	CLK1	CLK0
Compare-Timer-Steuerregister (Compare Timer Control Register). Es enthält die Auswahlbits für den Zähltakt des Compare-Timers und des Timers 2 sowie mehrere Interrupt-Request-Flags. Reset-Wert: 0xxx 0000B (80C517), 0x00 0000B (80C517A)							
Bitsymbol	Funktion						
T2PS1	Timer-2-Vorteilerbit 1 (Timer 2 Prescaler Bit 1). Zusammen mit T2PS (T2CON.7) läßt sich die Eingangsfrequenz und damit der Zähltakt des Timer 2 nach der Tabelle in Bild 11-5 einstellen.						
CTF	Compare-Timer-Überlaufflag. Dieses muß durch Software gelöscht werden. Falls der Interrupt des Compare-Timers freigegeben ist, löst CTF = 1 einen Interrupt aus.						
CLK2 CLK1 CLK0	Auswahl des Zähltakts für den Compare-Timer (siehe nachstehende Tabelle):						
	CLK2	CLK1	CLK0	Funktion			
	0	0	0	Zähltakt für den Compare-Timer ist $f_{osz}/2$			
	0	0	1	Zähltakt für den Compare-Timer ist $f_{osz}/4$			
	0	1	0	Zähltakt für den Compare-Timer ist $f_{osz}/8$			
	0	1	1	Zähltakt für den Compare-Timer ist $f_{osz}/16$			
	1	0	0	Zähltakt für den Compare-Timer ist $f_{osz}/32$			
	1	0	1	Zähltakt für den Compare-Timer ist $f_{osz}/64$			
	1	1	0	Zähltakt für den Compare-Timer ist $f_{osz}/128$			
	1	1	1	Zähltakt für den Compare-Timer ist $f_{osz}/256$			

Bild 11-8: Special-Function-Register CTCON (E1H)

Compare-Modus mit Timer 2

Compare-Modus 0 mit Compare-Timer und Register CMx

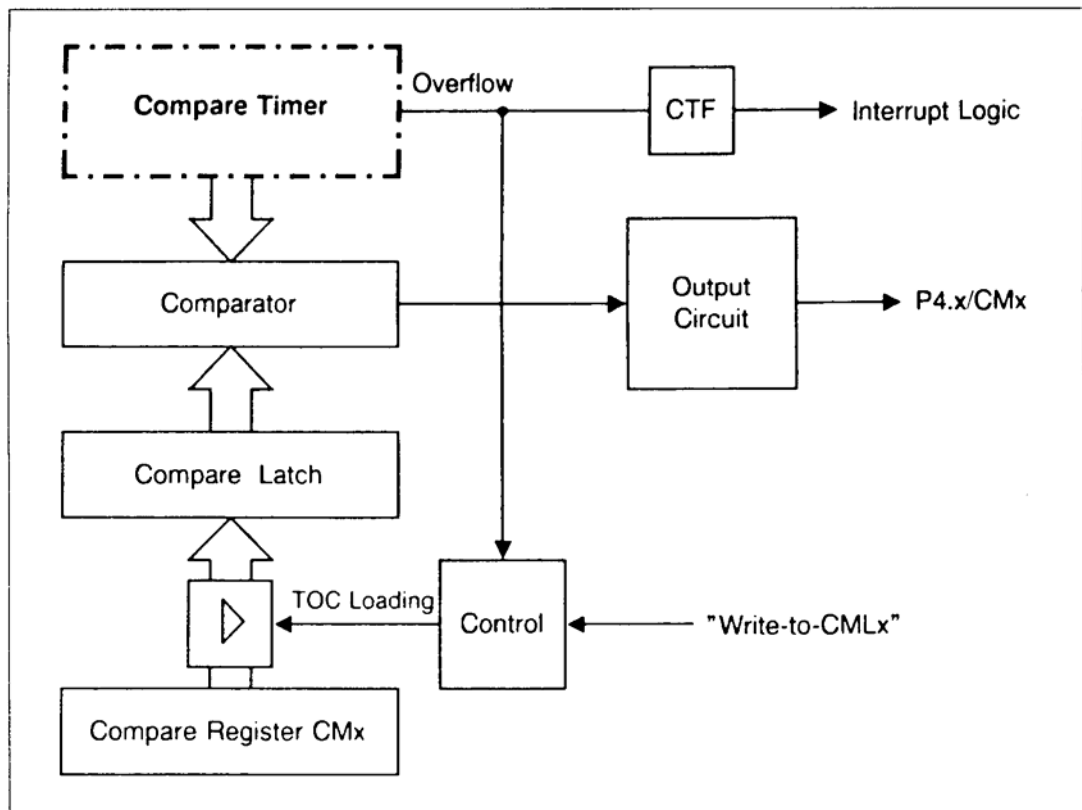


Bild 11-17: Compare-Modus 0 mit Compare-Timer und Register CMx

2.2 Anwendungsbeispiel PWM

Mit Hilfe der Capture Compare Einheit läßt sich einfach eine Pulsweitenmodulation aufbauen. Dazu verwendet man den Compare Timer.

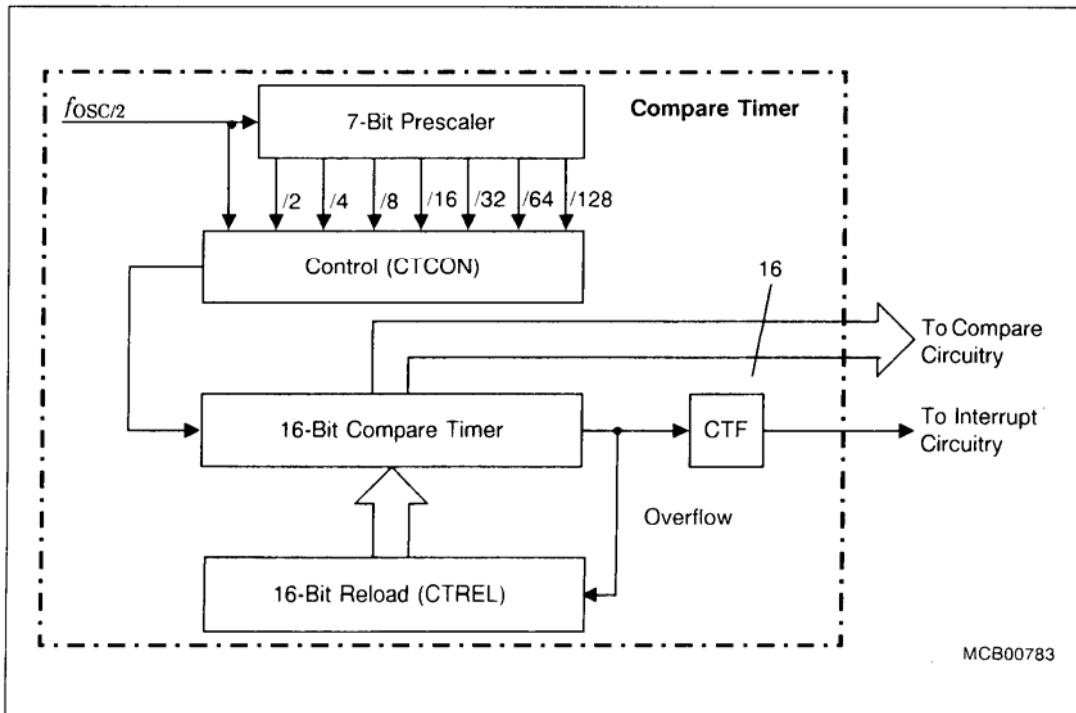


Bild 11-7: Blockschaltbild des Compare-Timers

Compare Timer

Mit Hilfe des sfr **CTCON** wird die Zählfrequenz eingestellt. In das sfr **CTRELL** und **CTRELH** werden die reload Werte eingeschrieben. Für eine Genauigkeit von 10 bit muß **CTRELH = 0xFC** eingegeben werden. **CTRELL = 0x00**. Damit lassen sich 1024 verschiedene Pulsweiten einstellen.

Der Zählerstand wird laufend mit einem anderen Register, dem Register **CMx** (**x = 0..7**), verglichen. Ist der Zählerstand des Registers kleiner als der des Compare Timers, so wird am Portpin **P4_x** eine 1 ausgegeben.

Das Register wird über das sfr **CMSEL** ausgewählt. Das Übungsboard soll auf Pin **P4_0** ausgeben, daher ist **CMSEL = 0x01**.

Die Initialisierung für die PWM ist in der Routine **PWM_INIT** programmiert. Das setzen eines neuen Werts wird mit der Funktion **PWM_VAL** durchgeführt.

Verwendete SFR:

CMEN (F6H), nicht bitadressierbar

MSB							LSB
CMEN7	CMEN6	CMEN5	CMEN4	CMEN3	CMEN2	CMEN1	CMEN0
Freigabe für CM-Register (CM-Register Enable). Bei Setzen des entsprechenden Bits wird die Compare-Funktion des jeweiligen CM-Registers freigegeben und dessen Ausgangssignal an den Portpin geleitet. Reset-Wert: 00H							
Bitsymbol	Funktion						
CMEN7	Freigabe-Bit für Register CM7						
CMEN6	Freigabe-Bit für Register CM6						
CMEN5	Freigabe-Bit für Register CM5						
CMEN4	Freigabe-Bit für Register CM4						
CMEN3	Freigabe-Bit für Register CM3						
CMEN2	Freigabe-Bit für Register CM2						
CMEN1	Freigabe-Bit für Register CM1						
CMEN0	Freigabe-Bit für Register CM0						

Bild 11-16: Special-Function-Register CMEN (F6H)

CMSEL (F7H), nicht bitadressierbar

MSB							LSB
CMSEL7	CMSEL6	CMSEL5	CMSEL4	CMSEL3	CMSEL2	CMSEL1	CMSEL0
Auswahl der CM-Register (CM Register Selection). Das Register legt fest, mit welchem Timer das jeweilige CM-Register arbeiten soll. CMSELx = 1 schlägt das Register CMx dem Compare-Timer zu und läßt es im Compare-Modus 0 arbeiten (PWM-Modus). CMSELx = 0 ordnet das Register CMx dem Timer 2 zu und läßt es im Compare-Modus 1 arbeiten. Reset-Wert: 00H							
Bitsymbol	Funktion						
CMSEL7	Auswahlbit für Register CM7						
CMSEL6	Auswahlbit für Register CM6						
CMSEL5	Auswahlbit für Register CM5						
CMSEL4	Auswahlbit für Register CM4						
CMSEL3	Auswahlbit für Register CM3						
CMSEL2	Auswahlbit für Register CM2						
CMSEL1	Auswahlbit für Register CM1						
CMSEL0	Auswahlbit für Register CM0						

Bild 11-15: Special-Function-Register CMSEL (F7H)

```


void pwm_init(unsigned int val) /*Compare mode 0, val = 0 - 1023*/
{
    CTCON = 0x00;          /* f_OSC/2 (default)          */
    CMSEL = 0x01;          /* Assign CM0 to the compare timer */
    CTRELH = 0xFC;         /* 10 bit reload -> 1024 steps     */
    CTRELL = 0x00;         /* Start T0C loading              */

    if (val >= 1024) {
        CMH0 = 0xFF;
        CML0 = 0xFF;
    } else if (val <= 0) {
        CMH0 = CTRELH;
        CML0 = CTRELL;
    } else {
        CMH0 = CTRELH + (unsigned char)(val / 256);
        CML0 = CTRELL + (unsigned char)(val % 256);
    }
    CMEN = 0x01;
}

void pwm_val(unsigned int val)
{
    if (val >= 1024) {
        CMH0 = 0xFF;
        CML0 = 0xFF;
    } else if (val <= 0) {
        CMH0 = CTRELH;
        CML0 = CTRELL;
    } else {
        /* load new PWM value into CM0 */
        CMH0 = CTRELH + (unsigned char)(val / 256);
        CML0 = CTRELL + (unsigned char)(val % 256);
    }
    CMEN = 0x01;
}

```

/* only WIFI-BOARD */

 **Übung 18:** Schreiben Sie ein Programm, welches mittels PWM auf Portpin P4_0 die Drehzahl eines Gleichstrommotors verändert. Die Drehzahl des Motors soll mit Hilfe des Timers 0 und des Timers 2 erfasst werden. PWM Wert und Drehzahl sollen auf der LCD Anzeige ausgegeben werden.

Erweiterung: Es sollen sich über die Tasten 0 bis 3 die PWM Werte 25% 50% 75% und 100% einstellen lassen. Wird eine andere Taste gedrückt, soll der Motor Abgeschaltet werden.

2.3 Musterlösung zu Übung 17

```

/*****
/*
/*          Übung 17
/*          Interruptgesteuerte Uhr
/*
/*
/*****

/****  Steueranweisungen an den Compiler      ****/
#pragma iv(0x8000)
#pragma mod517
#pragma code debug pl(61)

/****  Angabe der Include Dateien            ****/
#include <reg517.h>
#include <stdio.h>
#include "lcd.h"

#define uchar unsigned char
#define uint unsigned int

/****  Globale Variablen                    ****/
bit secflag;
uchar stu, min, sec, msec;
char buffer[21];

/****  Timer Interrupt 0 wird von der        ****/
/****  von der Zeitschleife ausgelöst      ****/
/****  und erzeugt alle 100 ms einen Interrupt ****/

void INTTIM0 (void) interrupt 1
{
    TH0 = -39;
    TL0 = 0;

    msec++;
    if ( msec == 100 )
    {
        msec = 0;
        sec++;
        secflag = 1;
    }
    if ( sec == 60 )
    {
        sec = 0;
        min++;
    }
    if ( min == 60 )
    {
        min = 0;
        stu++;
    }
    if ( stu == 24 ) stu = 0;
}

```



```

}

/**** Prototypen der Funktionen ****/
void set_time (uchar stunde, uchar minute, uchar secunde, uchar msecunde);
void ini_ser0(void);
void ini_ser1(void);

/**** Funktionen ****/
void set_time (uchar stunde, uchar minute, uchar secunde, uchar msecunde)
{
    stu = stunde;
    min = minute;
    sec = secunde;
    msec = msecunde;
}

/*****
/*      function ini_ser0 initialisiert serielle Schnittstelle 0      */
*****/

void ini_ser0(void)
{
    data unsigned int OLD;

    S0CON = 0x72; /* Serieller Mode 1, 8bit, 1 Stopbit, TI0 gesetzt */
    BD = 1;          /* Baudrate enable */
    OLD = PCON;     /* Baudrate 9600 bit */
    PCON = OLD | 0x80; /* SMOD = 1 */
}

/*****
/*      function ini_ser1 initialisiert serielle Schnittstelle 1      */
*****/

void ini_ser1(void)
{
    S1CON = 0xF2; /* Serieller Mode B, 8bit, 1 Stopbit, TI0 gesetzt */
    S1REL = 0xD9; /* Ladewert 217 fuer 9600 baud */
}

/*****
/*      Hauptprogramm      */
*****/

main ()
{
    TMOD = 0x01; /***** Timer 0 Mode 1 16 Bit *****/
    TH0 = -39; /***** Vorgesetzter Wert, alle 10 ms Int.*****/
    TL0 = 0; /***** Vorgesetzter Wert *****/
    ET0 = 1; /***** Timer 0 Interrupt disabled *****/
}

```

```
EAL = 1;           /**** Interrupt allgemein erlaubt      ****/  
TR0 = 1;          /**** Einschalten der Zeitschleife      ****/  
  
init_lcd();  
blank_lcd();  
ini_ser0();  
ini_ser1();  
  
set_time (20,30,0,0);  
  
while (1)  
{  
  
    while (!secflag);  
    secflag = 0;  
    printf ("%2u:%2u:%2u \n", (uint)stu, (uint)min, (uint)sec);  
    sprintf(buffer, "Uhr mit Timer 0 Int.");  
    print_lcd(1,1,buffer);  
    sprintf (buffer, "%2u:%2u:%2u", (uint)stu, (uint)min, (uint)sec);  
    print_lcd(2,4,buffer);  
  
}  
  
}
```

2.4 Lösung mit I2C Interrupt

```

/*****
/*          Uebung 17a          */
/*          Interruptgesteuerte Uhr          */
/*          mit I2C-Interrupt          */
*****/

/****  Steueranweisungen an den Compiler  *****/
#pragma iv(0x8000)
#pragma mod517
#pragma code debug pl(61)

/****  Angabe der Include Dateien  *****/
#include <reg517.h>
#include <stdio.h>
#include "lcd.h"

#define uchar unsigned char
#define uint unsigned int

/****  Globale Variablen  *****/
bit secflag;
uchar stu, min, sec, msec;
char buffer[21];

void INTI2C (void) interrupt 2
{
    sec++;
    secflag = 1;

    if ( sec == 60 )
    {
        sec = 0;
        min++;
    }
    if ( min == 60 )
    {
        min = 0;
        stu++;
    }
    if ( stu == 24 ) stu = 0;
}

/****  Prototypen der Funktionen  *****/
void set_time (uchar stunde, uchar minute, uchar secunde, uchar msecunde);
void ini_ser0(void);
void ini_ser1(void);

/****  Funktionen  *****/

```

```

void set_time (uchar stunde, uchar minute, uchar secunde, uchar msecunde)
{
    stu = stunde;
    min = minute;
    sec = secunde;
    msec = msecunde;
}

/*****
/*      function ini_ser0 initialisiert serielle Schnittstelle 0      */
*****/

void ini_ser0(void)
{
    data unsigned int OLD;

    S0CON = 0x72; /* Serieller Mode 1, 8bit, 1 Stopbit, TI0 gesetzt */
    BD = 1;      /* Baudrate enable */
    OLD = PCON;  /* Baudrate 9600 bit */
    PCON = OLD | 0x80; /* SMOD = 1 */
}

/*****
/*      function ini_ser1 initialisiert serielle Schnittstelle 1      */
*****/

void ini_ser1(void)
{
    S1CON = 0xF2; /* Serieller Mode B, 8bit, 1 Stopbit, TI0 gesetzt
*/
    S1REL = 0xD9; /* Ladewert 217 fuer 9600 baud */
}

/*****
/*      Hauptprogramm */
*****/

main ()
{
    EX1 = 1;      /***** Interrupt 1 enabled      *****/
    EAL = 1;      /***** Interrupt allgemein erlaubt      *****/
    IT1 = 1;      /* Flankensteuerung */

    init_lcd();
    blank_lcd();
    ini_ser0();
    ini_ser1();

    set_time (20,30,0,0);

    while (1)
    {

```

```
    while (!secflag);
    secflag = 0;
    printf ("%2u:%2u:%2u \n", (uint)stu, (uint)min, (uint)sec);
    sprintf(buffer, "Uhr mit I2C Interr.");
    print_lcd(1,1,buffer);
    sprintf (buffer, "%2u:%2u:%2u", (uint)stu, (uint)min, (uint)sec);
    print_lcd(2,6,buffer);

}

}
```

2.5 Musterlösung zu Übung 18

```

/*****
/*                               Übung 18                               */
/*   Pulsweitenmodulierte Ansteuerung eines Gleichstrommotors   */
/*           /* WIFI - BOARD only */                               */
/*****

/****  Steueranweisungen an den Compiler      ****/
#pragma noiv
#pragma mod517
#pragma code debug pl(61)

/****  Angabe der Include Dateien              ****/
#include <reg517.h>
#include <stdio.h>
#include "lcd.h"

#define uchar unsigned char
#define uint unsigned int

sbit P1_0 = 0x90;
sbit P1_1 = 0x91;

/****  Globale Variablen                      ****/
uchar t2low,t2high,taste;
char lcd_ser, x, y;

/****  Prototypen der Funktionen              ****/
void pwm_init(unsigned int val);
void pwm_val(unsigned int val);

```

```

/*****
/****          Timer Interrupt 0 wird von der          */
/****          von der Zeitschleife ausgelöst        */
/****          und erzeugt alle 10 ms einen Interrupt */
/*****

void INTTIM0 (void) interrupt 1
{
    t2low  =  TL2;
    t2high =  TH2;
    TH2    =   0;
    TL2    =   0;
    TH0    = -39;
    TL0    =   0;
}

/****  Externer Interrupt 1 wird von der          ****/
/****  Tastatur ausgelöst                        ****/
/****  Tastencode wird von Port 6 eingelesen     ****/

```

```

void EXTINT1 (void) interrupt 2
{
    taste = P6;
    taste >>= 3;

    switch (taste)
    {
    case 0: pwm_val(256);
            P1_1 = 0;
            break;
    case 1: pwm_val(512);
            P1_1 = 0;
            break;
    case 2: pwm_val(768);
            P1_1 = 0;
            break;
    case 3: pwm_val(1024);
            P1_1 = 0;
            break;
    default: pwm_val(0);
            P1_1 = 1;
            break;
    }
}
    
```

```

/**** Funktionen ****/

/*****
/*      PWM_INIT initialisiert den PWM, Auflösung 10 bit      */
/*=====
/* Parameter: unsigned int val: der Wert des PWM 0-1024 (0x0-0x3FF) */
/*****

void pwm_init(unsigned int val) /* Compare mode 0, val = 0 - 1023 */
{
    CTCON = 0x00;          /* f_OSC/2 (default) */
    CMSEL = 0x01;         /* Assign CM0 to the compare timer */
    CTRELH = 0xFC;        /* 10 bit reload -> 1024 steps */
    CTRELL = 0x00;        /* Start TOC loading */

    if (val >= 1024) {
        CMH0 = 0xFF;
        CML0 = 0xFF;
    } else if (val <= 0) {
        CMH0 = CTRELH;
        CML0 = CTRELL;
    } else {
        CMH0 = CTRELH + (unsigned char)(val / 256);
        CML0 = CTRELL + (unsigned char)(val % 256);
    }
    CMEN = 0x01;
}
    
```

```
}

```

```

/*****
/*      PWM_VAL laedt PWM mit neuem Wert      */
/*=====*/
/* Parameter: unsigned int val: der Wert des PWM 0-1024 (0x0-0x3FF) */
/*****
void pwm_val(unsigned int val)
{
    if (val >= 1024) {
        CMH0 = 0xFF;
        CML0 = 0xFF;
    } else if (val <= 0) {
        CMH0 = CTRELL;
        CML0 = CTRELL;
    } else {
        /* load new PWM value into CM0      */
        CMH0 = CTRELL + (unsigned char)(val / 256);
        CML0 = CTRELL + (unsigned char)(val % 256);
    }
    CMEN = 0x01;
}

```

```

/*****
/*      Hauptprogramm      */
/*****

```

```

main ()
{
    uint i,stellwert,speed;

    TMOD = 0x01;          /***** Timer 0 Mode 1 16 Bit      ****/
    TH0 = -39;           /***** Vorgesetzter Wert, alle 100ms Int.****/
    TL0 = 0;             /***** Vorgesetzter Wert      ****/
    ET0 = 1;            /***** Timer 0 Interrupt enabled  ****/
    T2CON = 0x02;       /***** Timer 2 Counter Einstellung ****/
    TH2 = 0;            /***** Timer 2 Startwert High Byte ****/
    TL2 = 0;            /***** Timer 2 Startwert Low Byte  ****/
    IT1 = 1;            /***** Externer Interrupt 1 negative Flanke */
    EX1 = 1;            /***** Externer Interrupt 1 enabled  ****/
    EAL = 1;            /***** Interrupt allgemein erlaubt  ****/
    TR0 = 1;            /***** Einschalten der Zeitschleife ****/

    init_lcd();
    blank_lcd();

    pwm_init (0);

    lcd_ser = 1;         /***** Ausgabe auf LCD Anzeige      ****/
    P1_0 = 0;           /***** Drehrichtung      ****/
    P1_1 = 0;           /***** Enable Leistungsteil      ****/
    taste = 4;

    while (1)

```



```
{
    speed = (41.66 * ( (float)t2low + 256.0 * (float)t2high ));
    stellwert = taste * 25.0 + 25;
    if (stellwert > 100) stellwert = 0;
    pos (1,1);
    printf ("PWM: %u%%    ",stellwert);
    pos (2,1);
    printf ("%u U/min    ",speed);
    warte (200);
}
}
```

2.6 Programmbeispiel LFU

```
/* ***** */
/* ** Programm: licht.c */
/* ***** */

#pragma debug pagelength (54) INTVECTOR(0x8000)
#include <stdio.h>
#include <reg517.h>
#include "lcd.h"

bit secflag,mess;
unsigned int Licht;
char buffer[20],ueb;
float licht2;

/* Funtionen */

void secticker(void) interrupt 2
{
    if (mess==1) /*letzte Sekunde = MESSUNG */
    {
        T2CON=0; /* Timer 2 sperren */
        secflag=1; /* Meszauswertung freigeben */
        mess=0;
    }
    else
    {
        T2CON=0x02; /* Timer 2 freigeben */
        ueb=0; /* šberl„ufe ruecksetzen*/
        mess=1; /* bit mess setzen */
    }
}

void t2int(void) interrupt 5
{
    TF2=0;
    ueb++;
}

/* Hauptprogramm */
main()
{
    EX1=1;
    IT1=1;
    ET2=1;
    T2CON=0;
    ueb=0;
    TF2=0;
    IP0=4;
    init_lcd();
```

```
blank_lcd();
EAL=1;

while(1)
{
if(secflag==1)
{
Licht=T2;
licht2=ueb*65536.0+Licht;
sprintf(buffer,"LFU hat %8.0f Hz    ",licht2);
print_lcd(1,1,buffer);
secflag=0;
T2=0;
P4++;
}
}
}
```