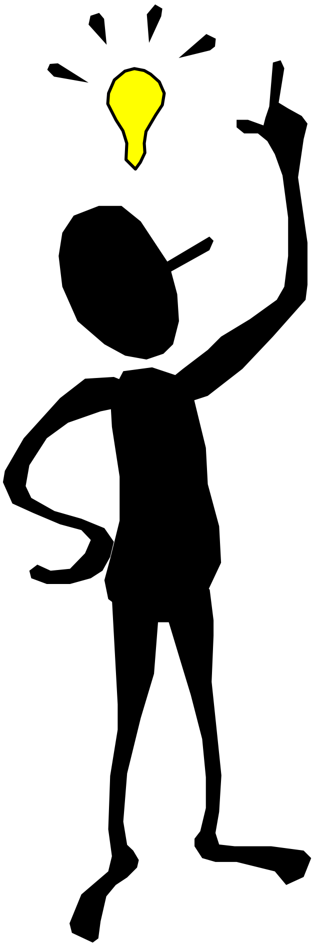


MODUL 8

Light-to-Frequency Converter

Version 1.0 Humer/Rothbart 1999

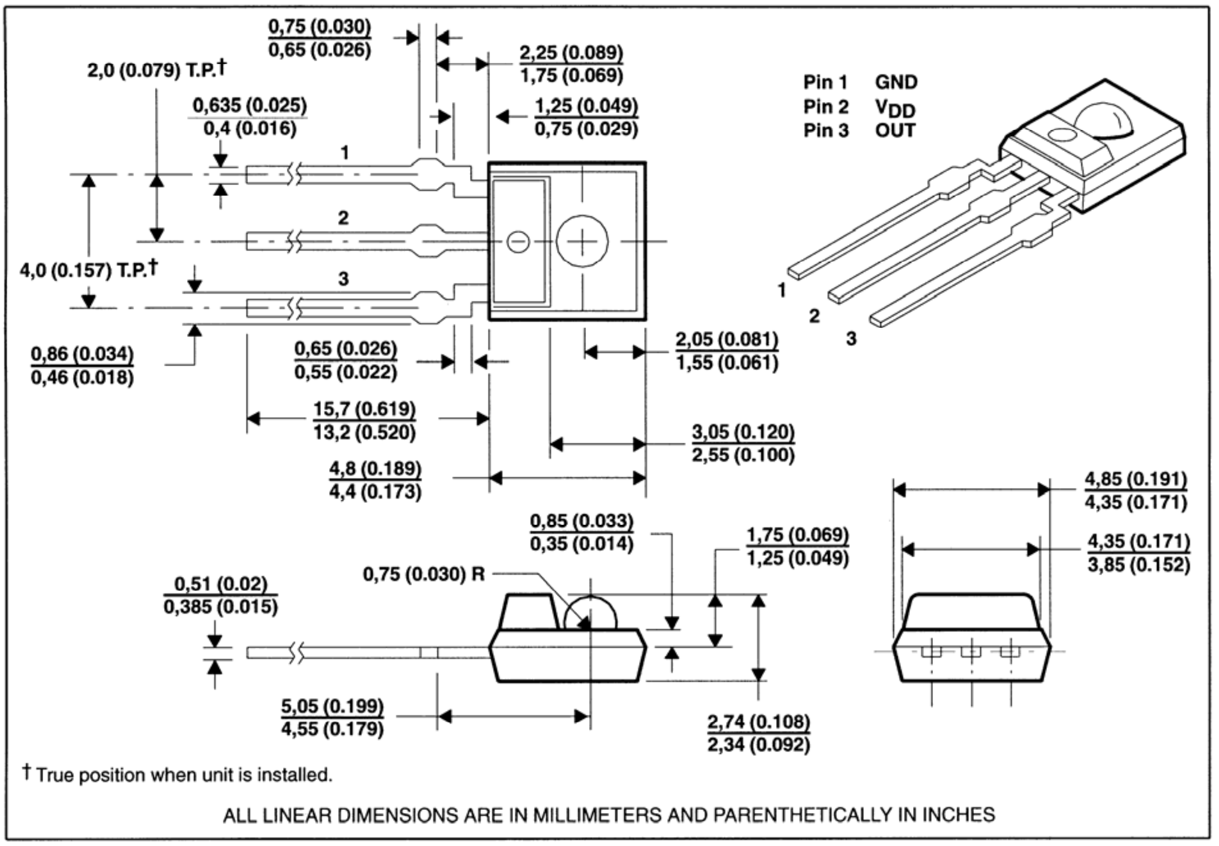
Inhaltsverzeichnis
Modul
Light-to-Frequency Converter

Inhalt **Seite**

1. Allgemeines über den LFU

Der TSL235 Licht-Frequenz Umsetzer verbindet eine Silikon-Photodiode und einein Strom-zu-Frequenz-Umsetzer auf einer einzelnen CMOS-Baustein. Der Ausgang ist ein Rechtecksignal (50% duty cycle), wobei die Frequenz direkt proportional zur Lichtintensität steht. Da er TTL kompatibel ist, kann man den Ausgang direkt mit z.B. einem Microcontroller verbinden. Der Baustein wurde für den ultra-violetten bis zum sichtbaren Licht-Bereich (300 nm bis 700 nm) temperaturkompensiert und arbeitet im Wellenlängenbereich von 300 nm bis 1100 nm. Der TSL235 ist charakteristisch für Operationen im Temperaturbereich von –25 °C bis 70 °C.

1.1. Bauteilmaße



1.2. Funktionelles Blockschaltbild

1.3. Genauigkeitsbestimmende Faktoren

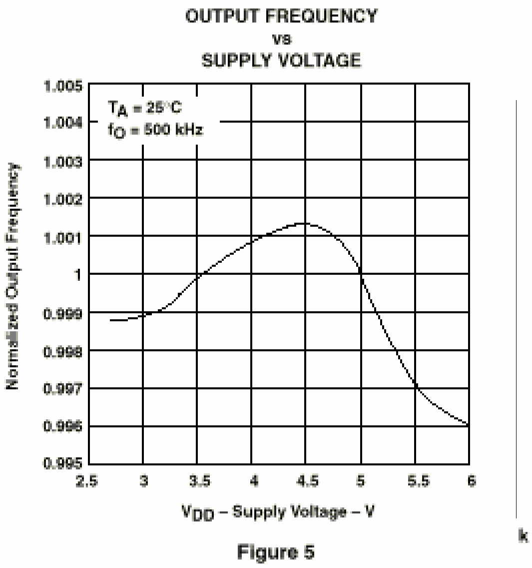


Figure 1

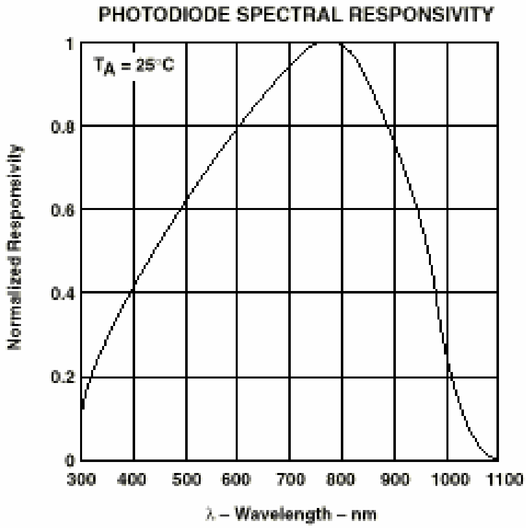


Figure 2

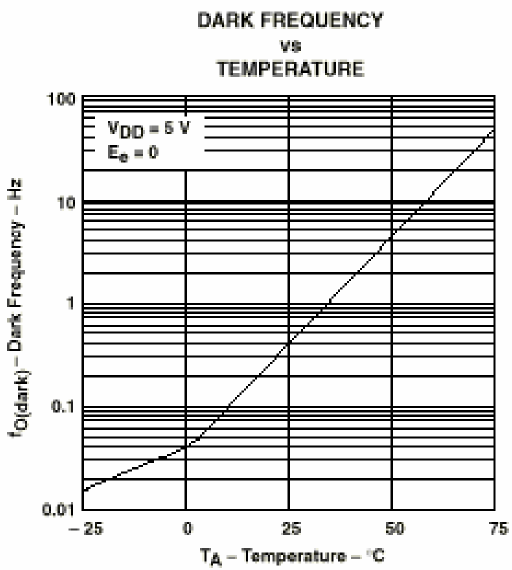


Figure 3

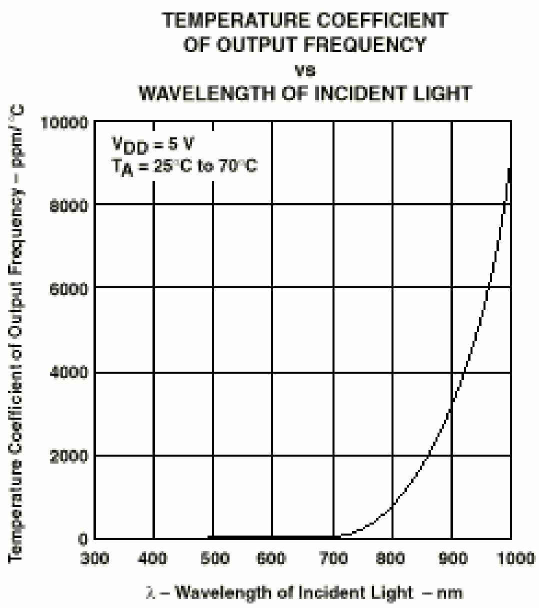


Figure 4

1.4. Kennlinie des Sensors

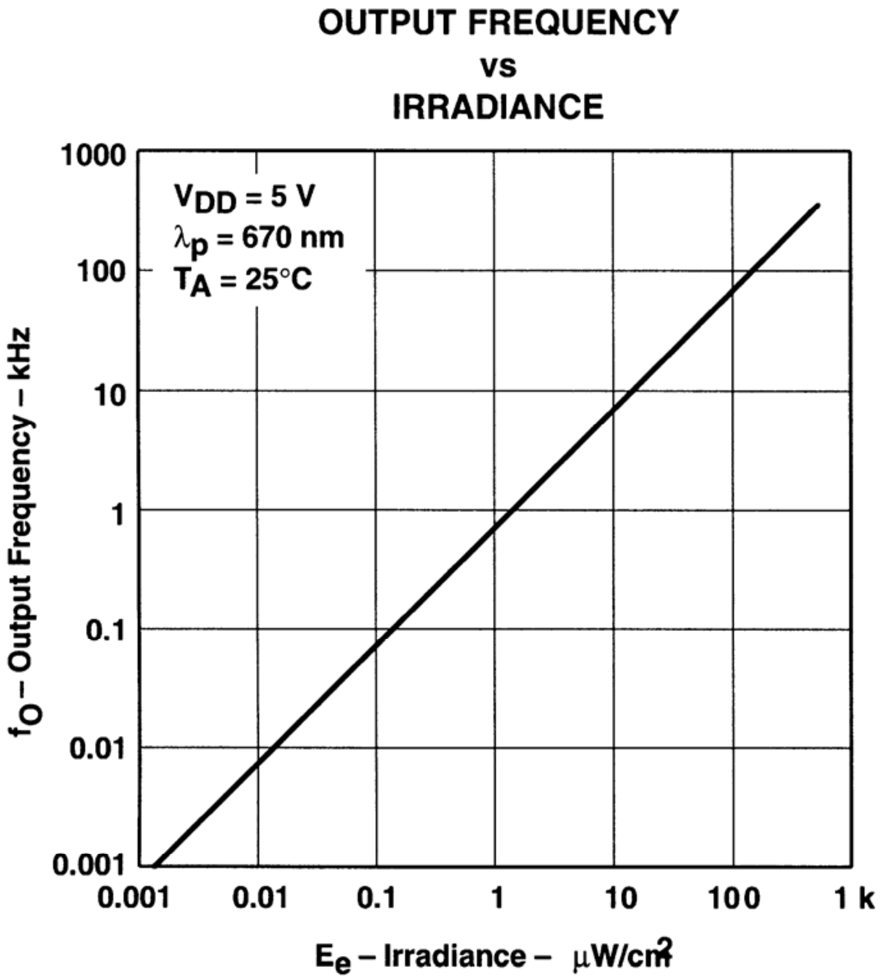


Figure 1

2. Anschluß des TSL235 an einen Microcontroller

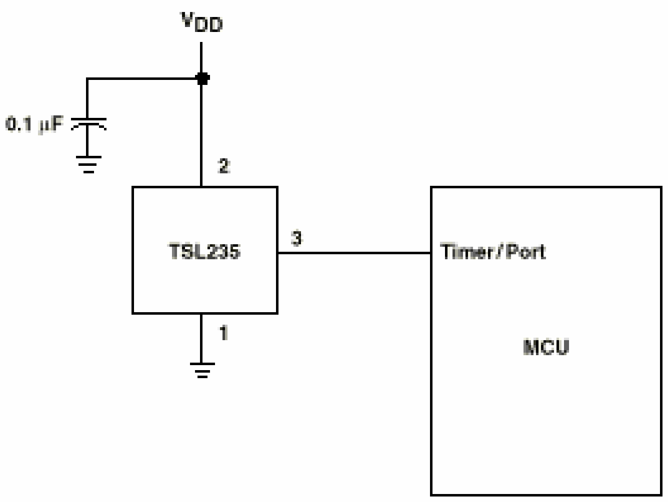
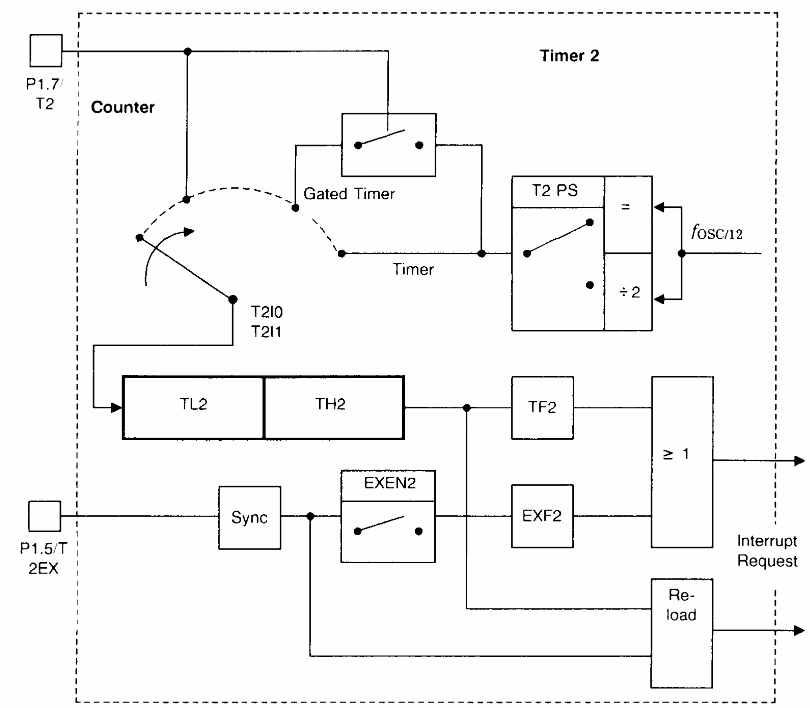


Figure 6. Typical TSL235 Interface to a Microcontroller

3. Microcontroller - Timer 2 Unit

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

Bitsymbol	Funktion
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)

Die Impulse des LFU liegen am Portpin T2 (P1_7) an. Bei jeder negativen Flanke wird der Zählerstand des 16 Bit Zählers um eins erhöht.

4. Programmbeispiel 1

Allgemeines:

Für die Programmbeispiele sind folgende Anschlüsse verwendet:

a) Ein Sekundentakt (extern) an INT1
b) Sensor (LFU) an PIN T2

In Beispiel 1 zählt der Timer2 die Impulse (16bit), die an P1.7 (T2) ankommen. Die vom LFU gesendeten Impulse werden direkt in den Timer 2 geleitet. Jede Sekunde werden der alte, der aktuelle Timer2-Stand und der Übertrag auf den Computer-Bildschirm ausgegeben. Danach wird die Frequenz, uW/cm² und Lux errechnet und auf das LCD ausgegeben.
Da der Timer2 nicht bei jeder Messung auf Null gesetzt wird, muß der alte Zählerstand gespeichert werden.
In der Interruptroutine „t2int“ werden die Überläufe gezählt.

```
/* ******************************************* */
/*           Programm: LICHTm1.c               */
/*    gibt Hz, uW/cm² und lx auf LCD aus       */
/* ******************************************* */
/* Steueranweisung an den Compiler */
#pragma debug pagelength (54) INTVECTOR(0x8000)

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

/* Globale Variablendefinitionen */
bit secflag;
unsigned int messung_old,messung;
char buffer[20];
char uebold,ueb;
float frequenz;



/* Funktionen */
```

```
void secticker(void) interrupt 2
{
     messung=T2;
     uebold=ueb;             // Überläufe in uebold speichern
     ueb=0;                  // Überläufe rücksetzen
     secflag=1;
}
```

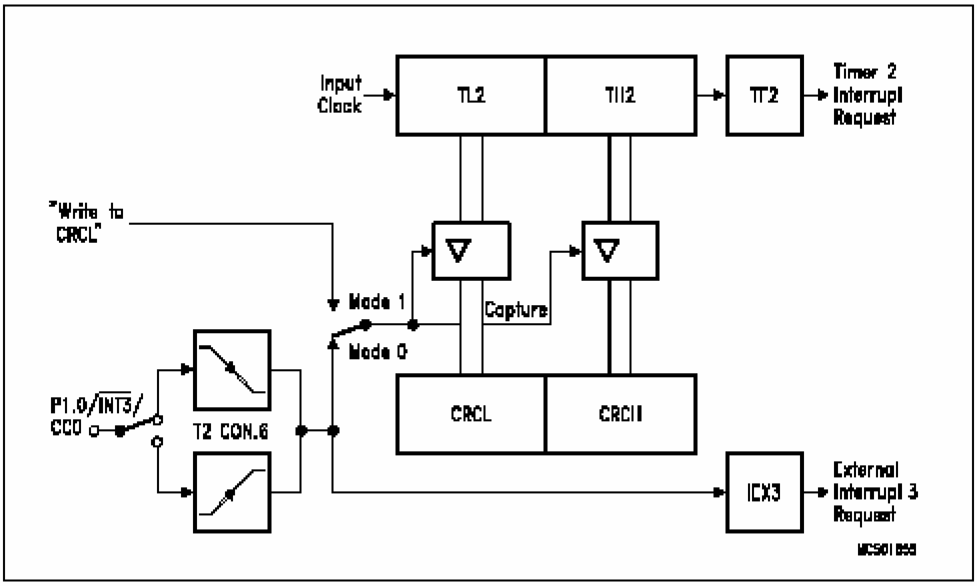
```
void t2int(void) interrupt 5
{
     TF2=0;                  // Interrupt-Request-Flag
rücksetzen
     ueb++;                  // Überläufe zählen
}
```

```
/* Hauptprogramm */
main()
{
 EX1=1;
 IT1=1;
 ET2=1;
 T2CON=0x02;     // Timer2 im Zählmodus; von Signal an P1.7 getaktet
 ueb=0;
 uebold=0;
 TF2=0;
 IP0=4;
 init_lcd();
 blank_lcd();
 EAL=1;

      while(1)
      {
      if(secflag==1)
       {
            printf("T2=%5u T2old=%5u uebold=%3bd
                                    ueb=%3bd\n",messung,messung_old,uebold,ueb);
            frequenz=((float)messung-(float)messung_old)+uebold*65536.0;
            sprintf(buffer,"LFU hat %7.0f Hz   ",frequenz);
            print_lcd(1,1,buffer);
            sprintf(buffer,"%7.3fuW",frequenz*.375/250);
            print_lcd(2,1,buffer);
            sprintf(buffer,"%5d lx",(int)(frequenz*.375*6.8/250));
            print_lcd(2,13,buffer);
            secflag=0;                  // Sekundenflag rücksetzen
            P4++;                       // Port4 = LEDs
            messung_old=messung;        // Zählerstand merken
       }
      }
}
```

5. Capture-Funktion der Capture-Compare Unit:

Jeder der vier compare/capture Register CC1 bis CC4 und das CRC-Register kann verwendet werden um den 16-bit Wert des Timer2 Registers (TL2 und TH2) zu latchen. Mit den zwei verschiedenen Modi kann diese Funktion eingestellt werden. In Modus 0 latcht ein externes Ereignis den Timer2-Inhalt in ein Capture-Register. In Modus 1 kann dies durch Schreiben in das Low-Order-Byte CRCL (vom 16 bit Capture Register) erfolgen. Also kann mit diesem Modus der Timer2-Inhalt fliegend ausgelesen werden.



$0C1_H$	COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0	CCEN

Compare/capture enable register selects compare or capture function for register CRC, CC1 to CC3.

Bit		Function
COCAH0	**COCAL0**	**Compare/capture mode for CRC register**
0	0	Compare/capture disabled
0	1	Capture on falling/rising edge at pin P1.0/$\overline{\text{INT3}}$/CC0
1	0	Compare enabled
1	1	Capture on write operation into register CRCL
COCAH1	**COCAL1**	**Compare/capture mode for CC register 1**
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.1/INT4/CC1
1	0	Compare enabled
1	1	Capture on write operation into register CCL1
COCAH2	**COCAL2**	**Compare/capture mode for CC register 2**
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.2/INT5/CC2
1	0	Compare enabled
1	1	Capture on write operation into register CCL2
COCAH3	**COCAL3**	**Compare/capture mode for CC register 3**
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.3/INT6/CC3
1	0	Compare enabled
1	1	Capture on write operation into register CCL3

6. Programmbeispiel 2

Hier wird jede Sekunde der Timer2-Inhalt in das Capture-Register gelatcht.

```
/* ****************************************** */
/*           Programm: lichtm2.c              */
/*      gibt Hz, uW/cm² und lx mittels        */
/*      Capture-Register auf LCD aus.         */
/* ****************************************** */

/* Steueranweisung an den Compiler */
#pragma debug pagelength (54) INTVECTOR(0x8000)

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

/* Globale Variablendefinitionen */
bit secflag;
unsigned int messung_old,messung;
char buffer[20];
char uebold,ueb;
float frequenz;

/* Funtionen */
```

```
void secticker(void) interrupt 2
{
      CRCL=0;                  // Timer2-Inhalt in Captureregister latchen
      messung=CRCH*256+CRCL;
      uebold=ueb;              // Überläufe in uebold speichern
      ueb=0;                   // Überläufe rücksetzen
      secflag=1;

}
```

```
void t2int(void) interrupt 5
{
      TF2=0;                   // Interrupt-Request-Flag rücksetzen
      ueb++;                   // Überläufe zählen
}
```

```
/* Hauptprogramm */
main()
{
 CCEN=0x03;     // Capture-Einheit auf Mode 1
 EX1=1;
 IT1=1;
 ET2=1;
 T2CON=0x02;    // Timer2 im Zählmodus; von Signal an P1.7 getaktet
 ueb=0;
 uebold=0;
 TF2=0;
 IP0=4;
 init_lcd();
 blank_lcd();
```

```
 EAL=1;

      while(1)
      {
      if(secflag==1)
       {

            printf("T2=%5u T2old=%5u uebold=%3bd
                                 ueb=%3bd\n",messung,messung_old,uebold,ueb);
            frequenz=((float)messung-(float)messung_old)+uebold*65536.0;
            sprintf(buffer,"LFU hat %7.0f Hz   ",frequenz);
            print_lcd(1,1,buffer);
            sprintf(buffer,"%7.3fuW",frequenz*.375/250);
            print_lcd(2,1,buffer);
            sprintf(buffer,"%5d lx",(int)(frequenz*.375*6.8/250));
            print_lcd(2,13,buffer);
            secflag=0;                      // Sekundenflag rücksetzen
            P4++;                           // Port4 = LEDs
            messung_old=messung;            // Zählerstand merken
       }
      }
}
```