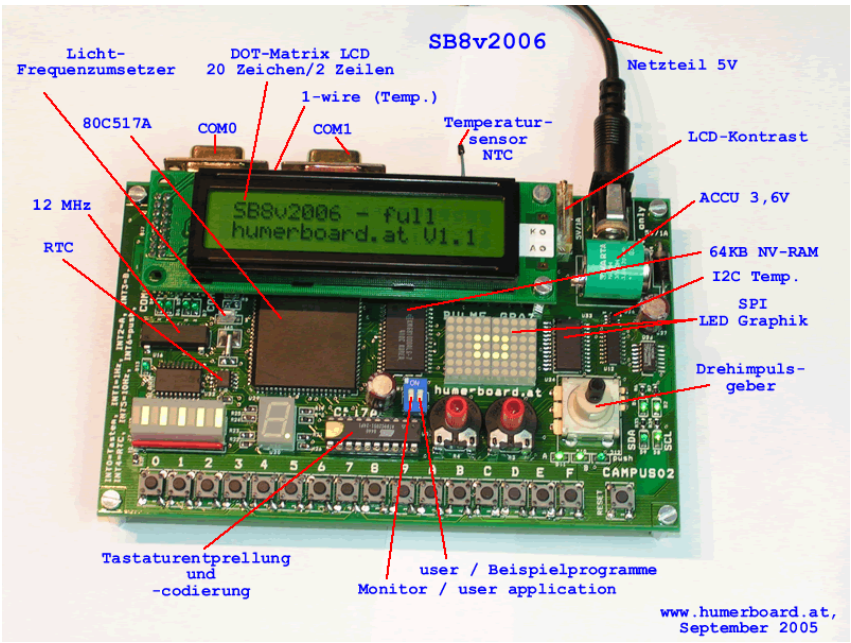
BULME GRAZ
INFINEON
CAMPUS02

Beispielprogramme
Keil – C51
SB8v2006



Version 1.1 / 20.11.2011

1 Inhaltsverzeichnis

2 LED Balken- und 7-Segmentanzeige

2.1 Visualisierungseinheit

Für die Visualisierung von Port 4 wird eine 10stellige LED Reihe zur Verfügung. LED1 wird dabei für den Sekundentakt und LED10 für den Tastaturinterrupt verwendet. Die dazwischenliegenden LED2..LED9 dienen zur Visualisierung von Port 4. Hier die Anzeige für den Wert 0xAA.

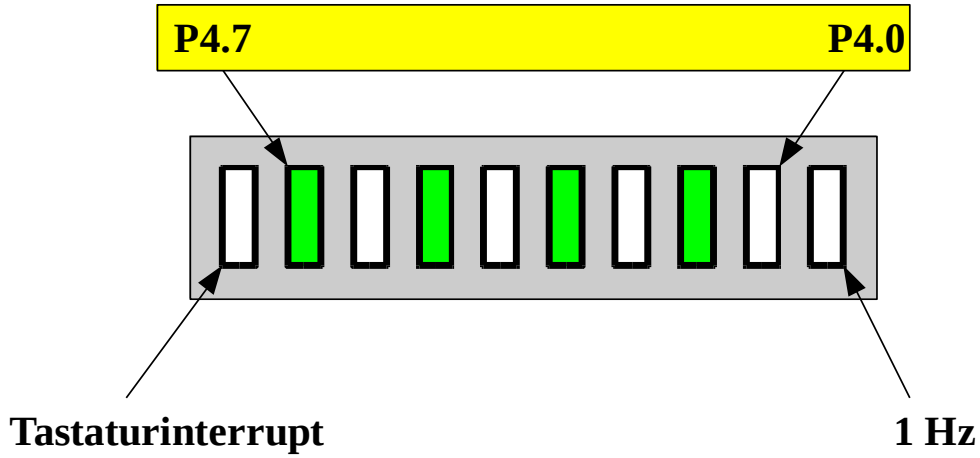


Bild: Visualisierung Port 4

2.2 LED-Balkenanzeige Schaltplan

Der Treiberbaustein U6 (Latch-Baustein) dient hier zur Selektion LED-Balken und der 7-Segmentanzeige (ebenfalls von P4 aus gesteuert). U6 wird mit der Portleitung P6.7 betrieben.

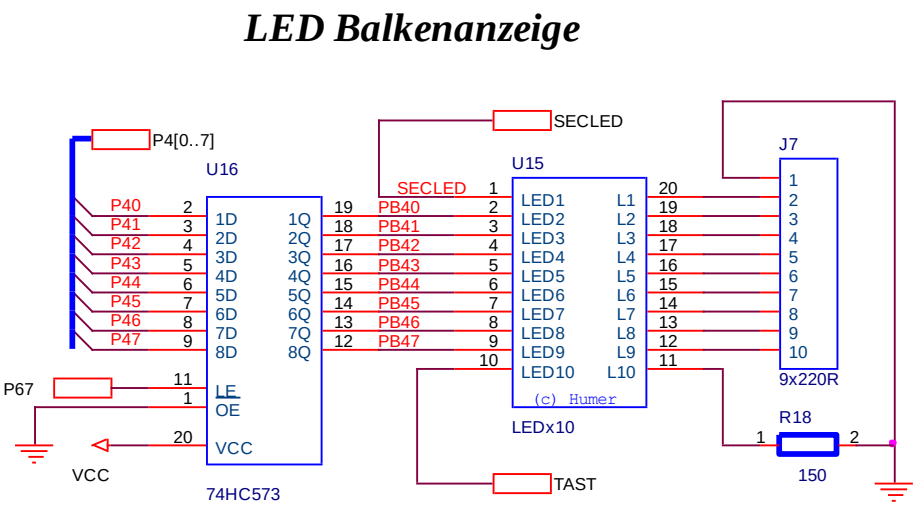
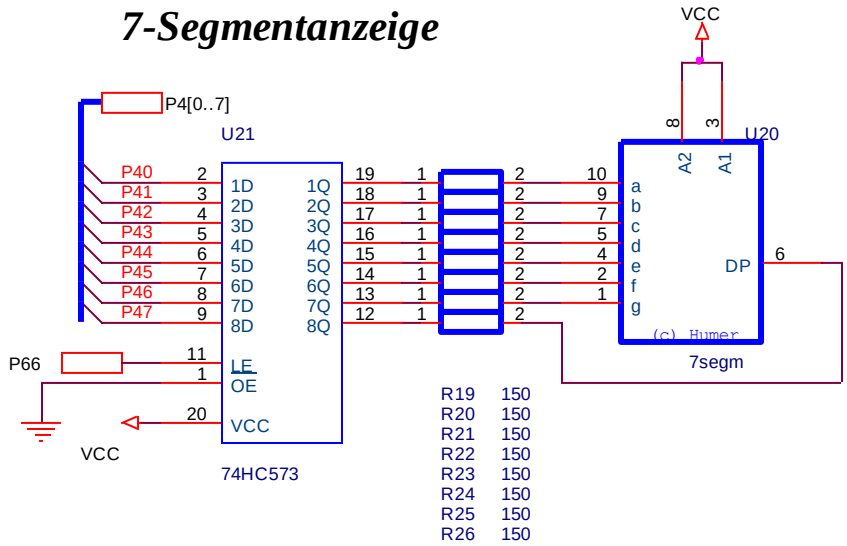


Bild: Schaltplan LED Balkenanzeige Board SB8V2006

2.3 7-Segmentanzeige Schaltplan



Die Daten liefert Port 4, die Steuerung selbst wird mit der Portleitung P6.6 betrieben. Die 7-Segmentanzeige wird mit negativer Logik betrieben. Die exakte Steuerung ist im nachfolgenden Programmbeispiel (als Funktion) dargestellt.

2.3.1 Funktion writebalken()

```
void writebalken(char balken)
{                           // P67...Balkenanzeige, P66... 7Seg. Anzeige
      P6=P6&0x3F;           //0011 1111 balken u 7Seg. Anz. sperren
      P6=P6|0x80;           //1000 0000 Oder-Verknüpfung - Balken active
      P4=balken;            // Daten anzeigen
      P6=P6&0x3F;           //0011 1111 balken u 7Seg. Anz. sperren
}
```

Nach dem RESET wird an der Balkenanzeige die Zahl 0x55 dargestellt und das Latch U6 transparent geschaltet.

2.3.2 Funktion write7Seg()

Für die Programmierung der 7-Segmentanzeige muss noch eine Konvertierungstabelle erstellt werden. Genaugenommen müssen 2 Tabellen wegen des Dezimalpunktes erstellt werden.

Tabelle ohne Dezimalpunkt

```
code char tab[16]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,
                   0x88,0x83,0xA7,0xA1,0x86,0x8E};
```

Tabelle mit Dezimalpunkt

```
code char tabm[16]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x18,
                   0x08,0x03,0x27,0x21,0x06,0x0E};
```

Funktion für die 7-Segmentanzeige

Beim Funktionsaufruf der Funktion write7Seg() werden 2 Parameter (Anzeigewert und Anzeige Dezimalpunkt) übergeben

```
void write7Seg(char SSeg, char dp)
{           P6=P6&0x3F; // Balken und 7Seg sperren
            P6=P6|0x40; // 7Seg freigeben
            if(dp) P4=tab[SSeg];
            else P4=tabm[SSeg];
            P6=P6&0x3F; // Balken und 7Seg sperren
}
```

Nach dem RESET-Zustand wird vom Monitorprogramm automatisch an der 7-Sementanzeige der Dezimalpunkt gesetzt und das Latch U21 gesperrt.

2.4 Beispiele Digital Out

2.4.1 Ausgabe an der Balkenanzeige

```
/*  ************************************************************** */
/*  *** NAME: HUMER, DATUM: 7.10.2010, DATEI: MUTZI.C   ********** */
/*  ****  LED ANSTEUERUNG PORT 4 BITMUSTER 01010101 UND 10101010 ******* */
/* ************************************************************** */

#include <reg517a.h>                    // SFR Definitionen
void mydelay(void);                     // Prototypdefinition
main()                                  // Beginn Hauptprogramm
{
      while(1)                          // Endlosschleife
      {
            P4=0x55;                    // Bitmuster 0101 0101
            mydelay();                  // Verzögerung
            P4=0xAA;                    // Bitmuster 1010 1010
            mydelay();                  // Verzögerung
      }
}


/* ******************* FUNKTION MYDELAY() ********** */

void mydelay(void)
{
      idata int i;      // Wertebereich -32768 .. 0 .. +32767
      for(i=0;i<20000;i++);
}
```

An der LED Balkenanzeige wird abwechseln die Bitkombination 01010101 und 10101010 angezeigt.

Die Verzögerung mydelay() wird durch eine Schleife realisiert. Der Wert 20000 kann verändert werden.

2.4.2 Einfaches Lauflicht

```
/* ************************************************ */
/* *** NAME: HUMER, DATUM: 7.10.2010, DATEI:  LAUFLI.C  ** */
/* ****   EINFACHES LAUFLICHT  ********************** */
/* ************************************************ */

#include <reg517a.h>
void mydelay(void);

void main(void)                         // Hauptprogramm
{
      char k;                           // Wertebereich -128..0..127
      P4=0x01;                          // Bitmuster 0000 0001
      while(1)                          // Endlosschleife
      {
            P4=0x01;                    // Bitmuster 0000 0001
            for(k=1;k<9;k++)
            {
                  mydelay();
                  P4<<=1;               // shift left
            }
      }
}

void mydelay(void)
{     int i;
      for(i=0;i<30000;i++);
}
```

2.4.3 Lauflicht (hin und her)

```
/* ************************************************ */
/* ** NAME: HUMER, DATUM: 7.10.2010, DATEI:  DREI.C   **** */
/* ***********   LAUFLICHT (HIN UND HER) ***************** */
/* ************************************************ */

#include <reg517a.h>                // SFR Definitionen
void mydelay(void);                 // Prototypendeklaration

void main(void)
{     char k;                       // Wertebereich -128..0..+127
      P4=0x01;                      //0000 0001
      while(1)                      // Endlosschleife
      {     P4=0x01;
            for(k=1;k<9;k++)
            {     mydelay();        // Funktionsaufruf
                  P4<<=1;           // shift left
            }
            P4=0x80;
            for(k=1;k<9;k++)
            {     mydelay();
                  P4>>=1; // shift rechts
            }
      }
}

void mydelay(void)
{     int i;
      for(i=0;i<30000;i++);
}
```

2.4.4 Binärzähler Balken- und 7Segmentanzeige

```
/* ********************************************** */
/*  NAME: HUMER, DATUM: 7.10.2010, DATEI:    VIER.C ****** */
/*  ****   BINÄRZÄHLER  BALK./7SEG      ***************** */
/* ********************************************** */

#include <reg517a.h>
void delay(void);
void write7Seg(char SSeg, char dp);
void writebalken(char balken);

code char tab[16]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,
                   0x88,0x83,0xA7,0xA1,0x86,0x8E};

code char tabm[16]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x18,
                   0x08,0x03,0x27,0x21,0x06,0x0E};

void main(void)
{     unsigned char counter;              //Variable WB 0..+255
      counter=0;
      writebalken(0);                     // Balkenanzeige löschen
      while(1)
      {     writebalken(counter++);       //counter++ = Zähler
            write7Seg(counter%16,0); //Modulo16 für die Adressierung tab[]
            delay();
      }
}
void delay(void)
{     unsigned int i;
      for(i=0;i<40000;i++);
}

void writebalken(char balken)
{                                 // P67...Balkenanzeige, P66... /Seg. Anzeige
      P6=P6&0x3F;                 //0011 1111 balken u 7Seg. Anz. sperren
      P6=P6|0x80;                 //1000 0000 Oder-Verknüpfung- Balkon active
      P4=balken;
      P6=P6&0x3F;                 //0011 1111 balken u 7Seg. Anz. sperren
}

void write7Seg(char SSeg, char dp)
{           P6=P6&0x3F;             // Balken und 7Seg sperren
            P6=P6|0x40;             // 7Seg freigeben
            if(dp)
                  P4=tab[SSeg];
            else
                  P4=tabm[SSeg];
            P6=P6&0x3F;             // Balken und 7Seg sperren
}
```

2.4.5 Binärzähler – Erweiterung serielle Schnittstelle

```
/* ****   BINÄRZÄHLER MIT ANZEIGE UND     ******************** */
/* **** DATENÜBERTRAGUNG ÜBER DIE SERIELLE SCHNITTSTELLE ******** */
/* ************************************************************ */


#include <reg517a.h>
#include <stdio.h>
void delay(void);
void write7Seg(char SSeg, char dp);
void writebalken(char balken);

code char tab[16]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,
                   0x88,0x83,0xA7,0xA1,0x86,0x8E};

code char tabm[16]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x18,
                   0x08,0x03,0x27,0x21,0x06,0x0E};

void main(void)
{
unsigned char counter;                    //Variable WB 0..+255
      counter=0;
      writebalken(0);                           // Balkenanzeige löschen
      while(1)
      {     writebalken(counter++);             //counter++ = Zähler
            write7Seg(counter%16,0); //Modulo16 für die Adressierung tab[]
      printf("Counter:= %03u dez %02X hex \n",(int)counter,(int)counter);
            delay();
      }
}


void delay(void)
{     unsigned int i;
      for(i=0;i<40000;i++);
}


void writebalken(char balken)
{// P67...Balkenanzeige, P66... /Seg. Anzeige
      P6=P6&0x3F;       //0011 1111 balken u 7Seg. Anz. sperren
      P6=P6|0x80;       //1000 0000 Oder-Verknüpfung- Balkon active
      P4=balken;
      P6=P6&0x3F;       //0011 1111 balken u 7Seg. Anz. sperren
}

void write7Seg(char SSeg, char dp)
{           P6=P6&0x3F; // Balken und 7Seg sperren
            P6=P6|0x40; // 7Seg freigeben
            if(dp) P4=tab[SSeg];
            else P4=tabm[SSeg];
            P6=P6&0x3F; // Balken und 7Seg sperren
}
```

2.5 Beispiel Digital In/Out

2.5.1 Einlesen der Tastatur (Polling)

Nachfolgend ist die Beschaltung der Tastatureinheit (Tasten 0..F) dargestellt. Der IC U22 bedient die Tastaturmatrix, entprellt die Tasten und codiert sie in 4bit (ana0=P7.0 bis ana3=P7.3) Dieser Baustein liefert auch noch einen Interrupt, angeschlossen an den externen Interupt0. Weiters steht auch noch ein Taktsignal von einem Herz für den externen Interrupt 1 zur Verfügung und ein 10Hz Signal wird noch für den externen Interrupt 5 zur Verfügung gestellt.

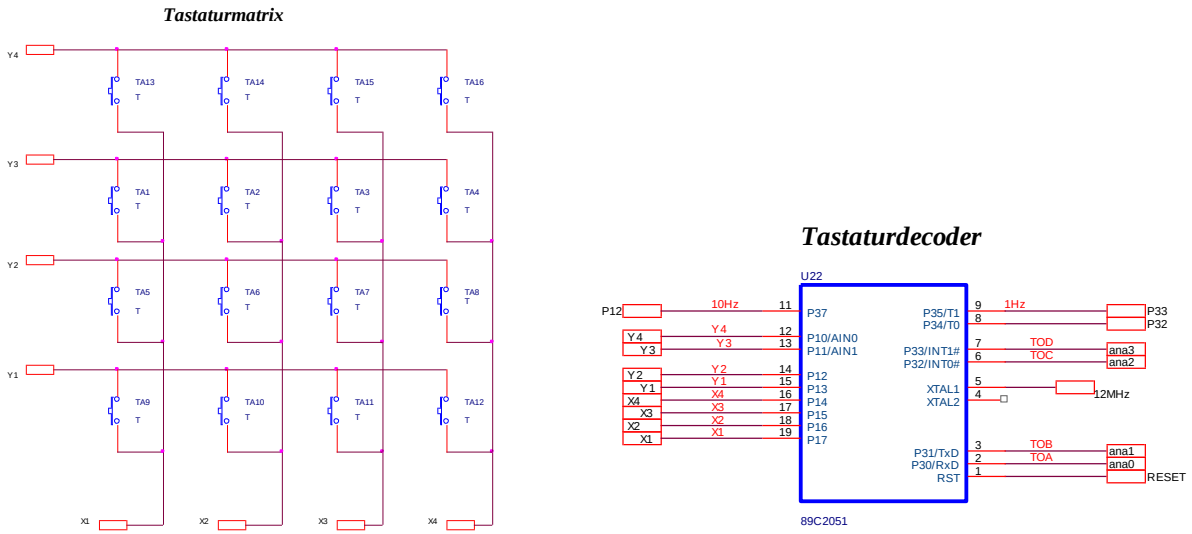


Bild: Schaltplan der Tastatur SB8v2006

```
/* ********************************************************** */
/* **** DATEI: SECHS.C,  DATUM: 14.10.2010, NAME: HUMER      *** */
/* ** EINLESEN DER TASTATUR UND ANZEIGE AM LED BALKEN      ********** */
/* ********************************************************** */

#include <reg517a.h>

char readTaste(void);                       // Prototyp

main()
{
      while(1)
      {
            P4=readTaste();                 // Anzeige im Polling-Verfahren
      }
}

char readTaste(void)
{
      return(P7&0x0F);                // Maskieren mit 00001111=0x0F
}
```

3 Interrupttechnik

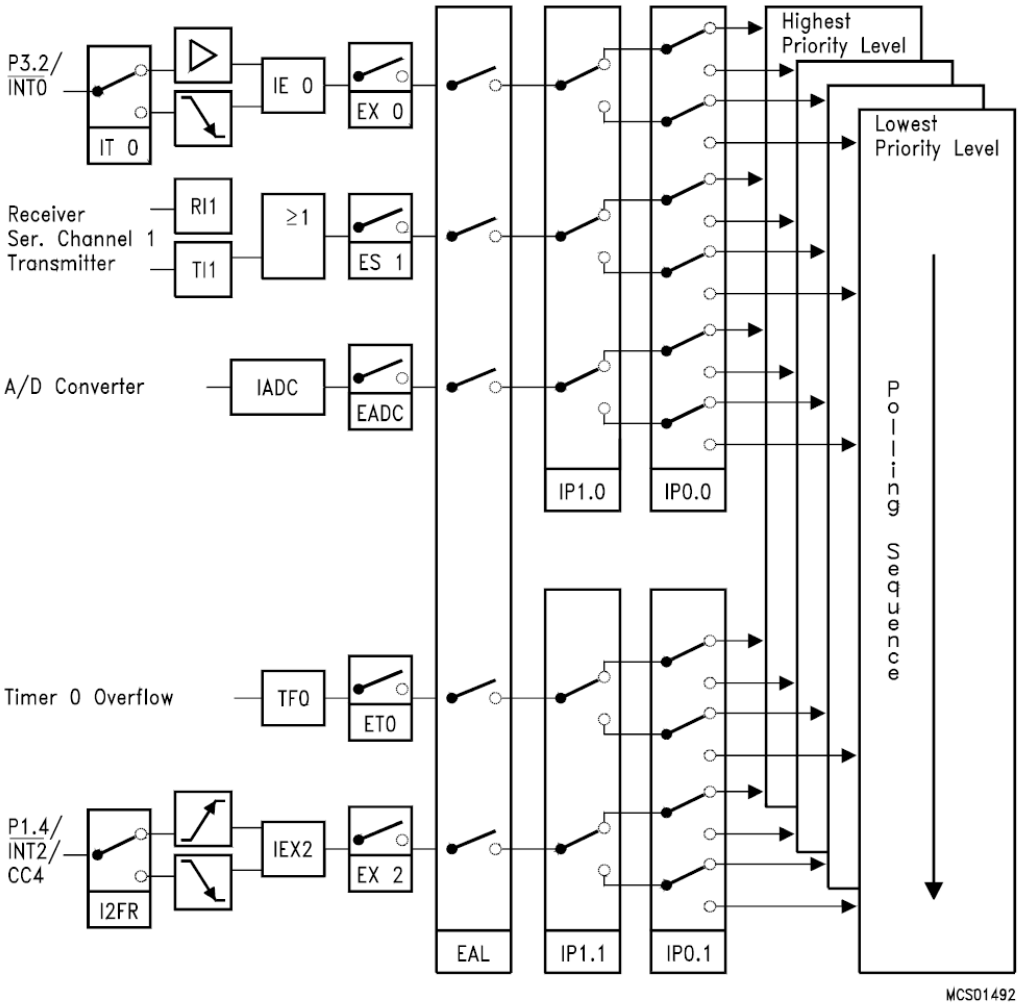
3.1 Allgemeines

Das Board SB8v2006 hat eine Reihe von bestückten Interruptquellen. Nachfolgend sind die externen Quellen tabellarisch aufgelistet.

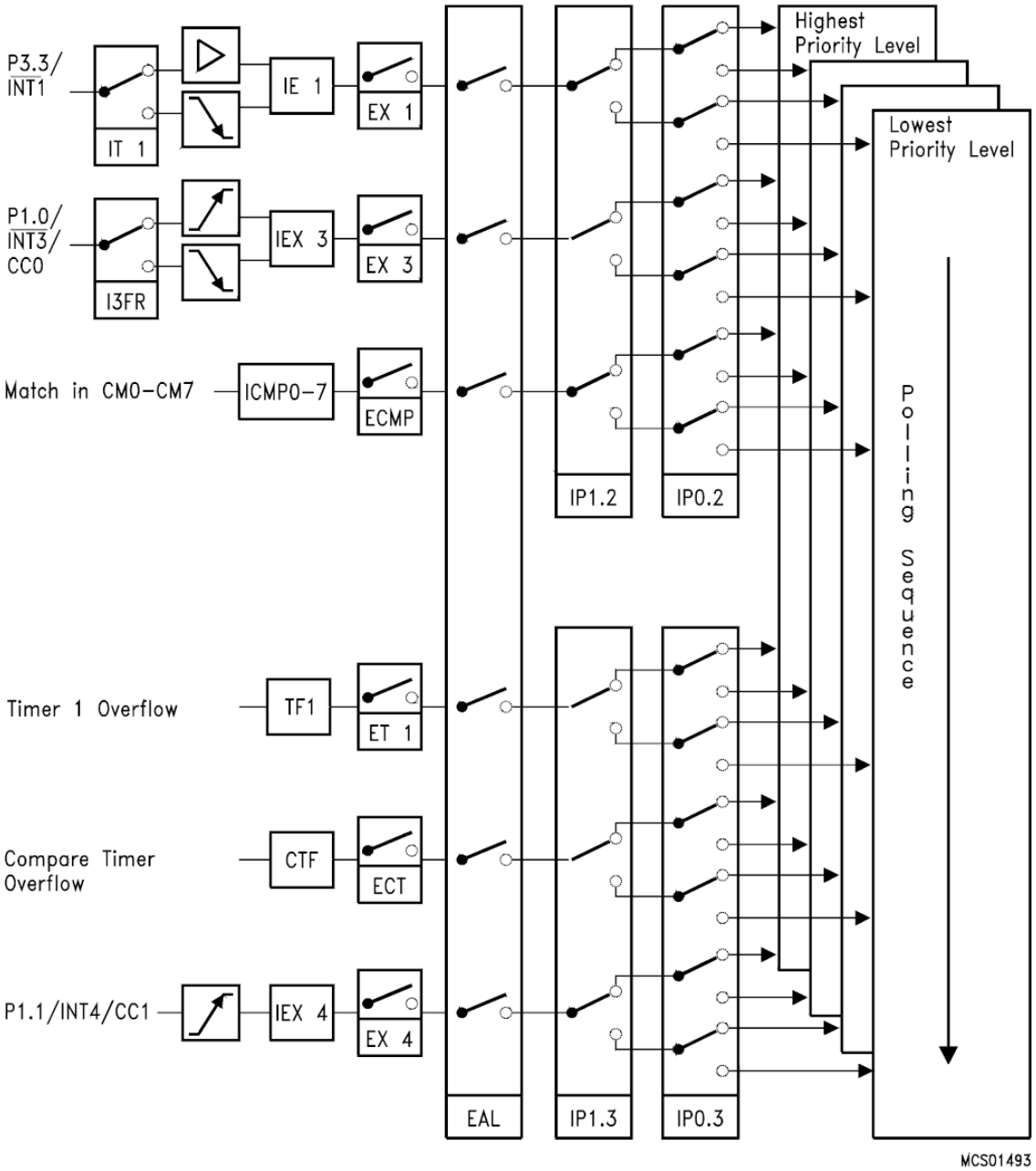
Interruptquelle	Externer Interrupt Nr.	Software Nummer
Tastatur	0	0
Sekundentakt	1	2
Drehimpulsgeber A	2	9
Drehimpulsgeber B	3	10
RTC	4	11
Takt 10Hz	5	12
Drehimpulsgeber push	6	13

Tabelle: Externe Interruptquellen

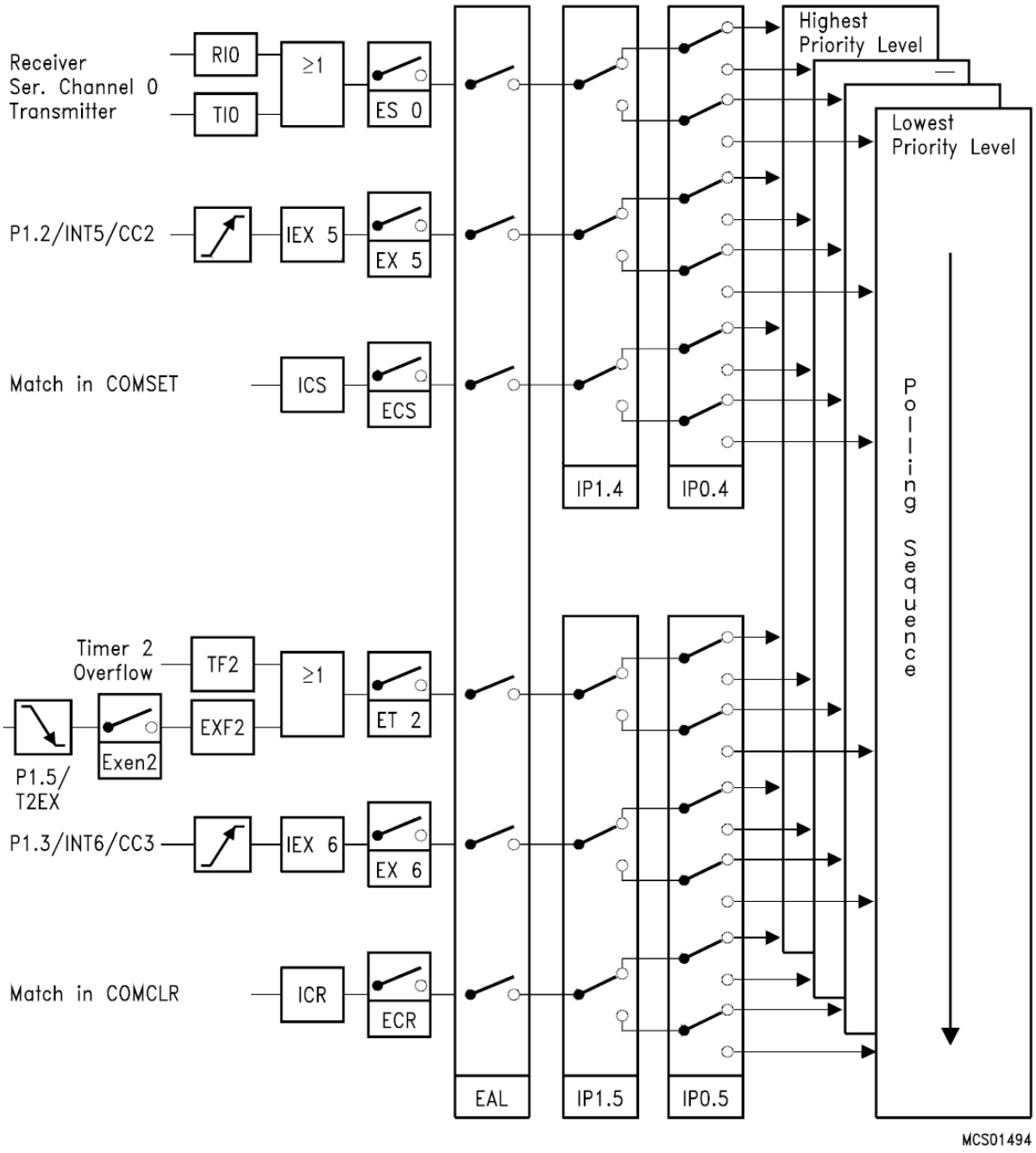
3.2 Blockschaltbild Interruptstruktur C517A (Teil1)



3.3 Blockschaltbild Interruptstruktur C517A (Teil2)



3.4 Blockschaltbild Interruptstruktur C517A (Teil 3)



3.5 Tabelle Interruptquellen und Vektoren

Interrupt Request Flags	Interrupt Vector Address	Interrupt Source
IE0	0003_H	External interrupt 0
TF0	$000B_H$	Timer 0 overflow
IE1	0013_H	External interrupt 1
TF1	$001B_H$	Timer 1 overflow
RI0 + TI0	0023_H	Serial channel 0
TF2 + EXF2	$002B_H$	Timer 2 overflow/ext. reload
IADC	0043_H	A/D converter
IEX2	$004B_H$	External interrupt 2
IEX3	0053_H	External interrupt 3
IEX4	$005B_H$	External interrupt 4
IEX5	0063_H	External interrupt 5
IEX6	$006B_H$	External interrupt 6
RI1/TI1	0083_H	Serial channel 1
ICMP0 to ICMP7	0093_H	Compare match interrupt of Compare Registers CM0-CM7 assigned to Timer 2
CTF	$009B_H$	Compare timer overflow
ICS	$00A3_H$	Compare match interrupt of Compare Register COMSET
ICR	$00AB_H$	Compare match interrupt of Compare Register COMCLR

3.6 Interrupt Softwarenummern für Keil C51

Int.-Quelle	Nr.	Auslösendes Moment	Request-Flag	Rücksetzen	Adresse
Ext. Interrupt 0	0	Neg. Flanke/Pegel	IE0	H	0x03
Timer 0	1	Timer 0 Überlauf	TF0	H	0x0B
Ext. Interrupt 1	2	Neg. Flanke/Pegel	IE1	H	0x13
Timer 1	3	Timer 1 Überlauf	TF1	H	0x1B
Serielle Schnittstelle	4	End In/Out	RI u. TI	S	0x23
Timer 2	5	Timer 2 Überlauf	TF2 u. EXF2	S	0x2B
A/D-Umsetzer	8	Ende der Wandlung	IADC	S	0x43
Ext. Interrupt 2	9	Neg./pos. Flanke	IEX2	H	0x4B
Ext. Interrupt 3	10	Neg./pos. Flanke	IEX3	H	0x53
Capture 0 In	10	Neg./pos. Flanke	IEX3	H	0x53
Capture 0 Out	10	Neg./pos. Flanke	IEX3	H	0x53
Ext. Interrupt 4	11	Pos. Flanke	IEX4	H	0x5B
Capture 1 In	11	Pos. Flanke	IEX4	H	0x5B
Capture 1 Out	11	Neg./pos. Flanke	IEX4	H	0x5B
Ext. Interrupt 5	12	Pos. Flanke	IEX5	H	0x63
Capture 2 In	12	Pos. Flanke	IEX5	H	0x63
Capture 2 Out	12	Neg./pos. Flanke	IEX5	H	0x63
Ext. Interrupt 6	13	Pos. Flanke	IEX6	H	0x6B
Capture 3 In	13	Pos. Flanke	IEX6	H	0x6B
Capture 3 Out	13	Neg./pos. Flanke	IEX6	H	0x6B

Tabelle: Keil C51 Softwarenummern für die Interruptfunktionen

Beispiel für eine Interruptfunktion für den Timer 0

```
Void zeitticker(void) interrupt 1
{
……..
}
```

3.7 Beispiele

3.7.1 Einlesen der Tastatur – Interruptgesteuert

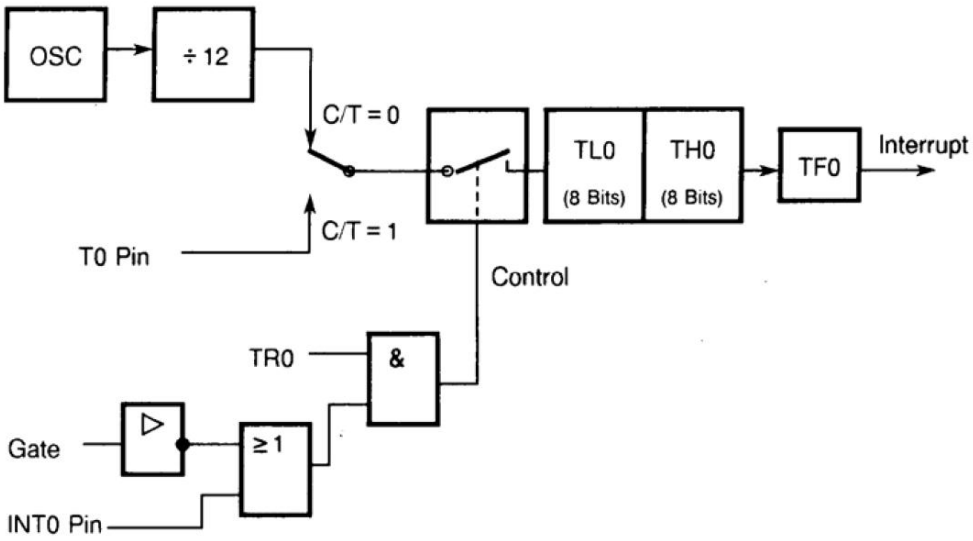
```
/* ************************************************************ */
/* **** Datei: sieben.c,  Datum: 21.10.2010, Name: Humer      **** */
/* ** Einlesen der Tastatur und Anzeige am LED Balken      ********** */
/* ************************************************************ */

#include <reg517a.h>

char readTaste(void);
void myinit(void);

void mitzitant(void) interrupt 0          // Interruptfunktion Ext. Nr.0
{
P4=readTaste();                           // Funktionsaufruf
}


main()
{     myinit();                           // Funktionsaufruf
      while(1);                           // Endlosschleife
}


void myinit(void)
{
      IT0=1;                              // neg. Flanke
      EX0=1;                              // enable externer Interrupt 0
      EAL=1;                              // Generelle Freigabe
}


char readTaste(void)
{
      return(P7&0x0F);                    // Maskiertes Einlesen
}
```

3.7.2 Binärzähler (LED Balken) und Einlesen der Tastatur (7Segmentanzeige)

```
/* ********************************************************** */
/* ***** Datei: acht.c,  Datum: 21.10.2010, Name: Humer      **** */
/* ***** Einlesen der Tastatur und Anzeige an 7Seg           **** */
/* ***** Binärzähler mit 1Hz an der Balkenanzeige             *** */
/* ********************************************************** */

#include <reg517a.h>
char readTaste(void);
unsigned char counter;
void myinit(void);
void write7Seg(char SSeg, char dp);
void writebalken(char balken);
code char tab[16]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,
                   0x88,0x83,0xA7,0xA1,0x86,0x8E};
code char tabm[16]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x18,
                   0x08,0x03,0x27,0x21,0x06,0x0E};
void mitzitant(void) interrupt 0
{
      write7Seg(readTaste(),1);          // Anzeige nach Tastendruck
}
void secticker(void) interrupt 2         // Sekundenticker
{
      counter++;
      writebalken(counter);              // Anzeige am Balken
}

main()
{     myinit();
      while(1);                          // Endlosschleife
}
void myinit(void)
{
      counter=0;
      IT0=1;                             // neg. Flanke
      EX0=1;                             // enable externer Interrupt 0
      EAL=1;                             // Generelle Freigabe
      IT1=1;
      EX1=1;
}
char readTaste(void)
{
      return(P7&0x0F);                   // Einlesen und maskieren
}
void writebalken(char balken)
{                                // P67...Balkenanzeige, P66... /Seg. Anzeige
      P6=P6&0x3F;                //0011 1111 balken u 7Seg. Anz. sperren
      P6=P6|0x80;                //1000 0000 Oder-Verknüpfung- Balkon active
      P4=balken;
      P6=P6&0x3F;                //0011 1111 balken u 7Seg. Anz. sperren
}
void write7Seg(char SSeg, char dp)
{           P6=P6&0x3F; // Balken und 7Seg sperren
            P6=P6|0x40; // 7Seg freigeben
            if(dp) P4=tab[SSeg];
            else P4=tabm[SSeg];
            P6=P6&0x3F; // Balken und 7Seg sperren
}
```

4 Timer

4.1 Beispiel Binärzähler, gesteuert mit Timer



Blockschaltbild Timer 0 und Timer 1 C517A

4.2 TMOD Register

Diese Beschreibung dient nur als Ergänzung zum Datenblatt.

MSB LSB

Timer 1				Timer 0			
Gate	**C/T**	**M1**	**M0**	**Gate**	**C/T**	**M1**	**M0**

Tabelle TMOD Registeraufbau

M1	M0	Funktion
0	**0**	THx dient als 8-bit Timer, TLx bildet einen 5-bit Vorteiler
0	**1**	THx und TLx bilden gemeinsam einen 16-bit Timer
1	**0**	Auto-Reload Modus (8bit), THx wird in TLx bei Timerüberlauf kopiert
1	**1**	THx und TLx bilden beide einen 8bit Timer

Tabelle Funktionen von M0 und M1

4.3 Beispiel

4.3.1 Zeitticker mit Timer 0, 50ms

```
/* ********************************************************** */
/* **** Datei: BCT0.c **** Datum: 04.11.2010 *** Name:  Humer ***** */
/* **** Binärzähler auf der Basis Timer 0  ********************** */

#include <reg517a.h>

void myinit(void);

void ticker(void) interrupt 1                          // Timer0-Interrupt
{                       // Es soll alle 50ms ein Interrupt erzeugt werden
                        // Timer0 auf 15535 vorsetzen = -50000
                        // so nicht! T0=15535; auf 2x8bit Werte zerlegen
      TL0=0xAF;
      TH0=0x3C;
      P4++;
}

main()
{     myinit();
      while(1);
}

void myinit(void)
{
      P4=0;
      TR0=1;
      TMOD=0x01;                          // 16bit Timer
      ET0=1;                              // enable Timer0 Interrupt
      EAL=1;                              //Generelle Interruptfreigabe
}
```

4.3.2 Elektronischer Würfel (Zufallszahlermittlung)

Durch einen Tastendruck soll eine Zufallszahl zwischen 1 und 6 ermittelt werden. Die Anzeige soll auf dem 7Segment erfolgen. Die Zufallszahl wird durch das Auslesen von Timer 0 erreicht.

```
/* ******************************************************** */
/* ******  ZUFALLSZAHLENGENERATOR WERTE 1..6          ********** */
/* ******  TIMER0 (TH DIENT ALS QUELLE), ANZ. 7 SEG   ********** */
/* ******  DATUM: 19.11.10, HUMER, ZUFALL.C           ********** */
/* ******************************************************** */

#include <reg517a.h>
bit flag;

void write7Seg(char SSeg);
code char tab[16]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x18,
                                0x08,0x03,0x27,0x21,0x06,0x0E};

void myinit(void);

void pressbutton(void) interrupt 0
{
      flag=1;
}
main()
{     myinit();                                    // Funktionsaufruf myinit()
      while(1)
      {
            if(flag)
            {     flag=0;
                  write7Seg((TH0%6)+1);
            }
      }
}

void myinit(void)
{
      EX0=1;
      IT0=1;
      EAL=1;
      TMOD=0x01;
      TR0=1;
}

void write7Seg(char SSeg)
{
      P6=P6&0x3F;                          // 0011 1111.. Anzeigen sperren
      P6=P6|0x40;                          // 0100 0000 7Seg aktivieren
      P4=tab[SSeg];                        // Aufruf der Tabelle
      P6=P6&0x3F;                          // 0011 1111.. Anzeigen sperren
}
```

5 Serielle Schnittstelle

5.1 Initialiserung

Durch die Verwendung der modifizierten Keil-Monitor-Software wird die serielle Schnittstelle bereits auf 9600 baud konfiguriert. Der Microcontroller braucht somit nicht mehr konfiguriert werden. Für ein Standalone-Programm muss natürlich laut Datenblatt die Schnittstelle konfiguriert werden.

In unserem Beispiel wurde der Baudrategenerator mit 12 MHz Taktfrequenz verwendet:

```
void ini_ser0(void)      // Initialisierung laut Datenblatt m. Baudrategen.
{                        // 80C517A-Board, 12MHz, sb8V2006-humerboard.at
      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 */
}
```

Für die Anwendungungen der seriellen Schnittsellen können alle ANSI-C-Befehle verwendet werden.

```
Printf()
Scanf()
Putchar()
Getchar()
Etc.
```

Mit dem Arbeiten ist es notwendig für Eingaberoutinen den Programmcode der modifizierten Funktion getkey.c einzubinden. UVision sendet periodisch ein ESC-Zeichen zur Hardware, diese versteht dies aber als Eingabezeichen. Durch den modifizierten Programmcode wird dies abgeblockt.

```
char _getkey ()
{
      char c;
      do
            {
            while (!RI);
            c = SBUF;
            RI = 0;
            }
      while(c==0x11);         // Abfrage auf ESC-Zeichen
      return (c);
}
```

Sollte ein ESC-Zeichen empfangen werden, wird es nicht weitergegeben!

5.2 Beispiel

5.2.1 Diverse Ein-/Ausgaben

```
/* ******************************************************** */
/* ********* BEISPIELPROGRAMM FÜR DIE SERIELLE SCHNITTSTELLE ****** */
/* ********* 4CHELI, 28.10.2010, HUMER ********************** */
/* ******* DATEI: RS232.C *********** C51 UVISION C517ABOARD ** */
/* ***** DIE DATEI GETKEY.C NICHT VERGESSEN !!! *************** */

#include <reg517a.h>                    // SFR Registerdefinitionen
#include <stdio.h>                      // Standard_In_Out Bibl.
#include <string.h>                     // String Bibliothek

idata char a;                           // max 128 Byte im idata
idata int b;                            // 16bit signed Variable
idata float c;                          // Speicherort indirekt adressierbar
char text[20];                          // Array mit 10 char Variablen
void ini_ser0(void);                    // Prototypendeklaration

main()
{
      a=5;                              // Wertzuweisung char
      b=17;                             // Wertzuweisung integer
      c=5.34;                           // Wertzuweisung float
      ini_ser0();                       // Funktionsaufruf Initialisierung RS232

      while(1)
      {
            printf("Hello world\n");
            printf(" Wert char = %bd\n",a);
            printf(" Wert integer = %d\n",b);
            printf ("Wert float = %5.2f\n",c);
                                        // Einlesen von Zahlen und Wörtern
            printf("Geben Sie eine Zahl (Ganzzahl) ein:\n");
            scanf("%d",&b);
            printf("\nEingegebener Wert: %d\n",b);
            printf("Geben Sie ein Wort ein:\n");
            scanf("%s",&text);
            printf("Das Wort lautet: %s\n",text);
            printf("Anzahl der Buchstaben=%d\n",strlen(text));
            Printf("************************************************\n");
      }
}

void ini_ser0(void)      // Initialisierung laut Datenblatt m. Baudrategen.
{                        // 80C517A-Board, 12MHz, sb8V2006-humerboard.at
      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 */
}
```

5.2.2 Simpler Taschenrechner

Die PC-Console dient als Eingabeeinheit, die Daten werden über die RS232-Schnittstelle an das Board geschickt, dort das Ergebnis ermittelt und wieder über die RS232 an den PC geschickt. Mit einem Terminalprogramm oder uVision kann der Datenfuss an der seriellen Schnittstelle visualisiert werden.

```
/* ************************************************************ */
/* ********* DATEI:    RECHNER.C ********************************** */
/* ********* DATUM:    28.10.2010    ***************************** */
/* ********** COMPILER: KEIL - C 51 VERSION 6.0 ****************** */
/* ************************************************************ */
#include <reg517a.h>
#include <stdio.h>
/* ******************* Variablendefinitionen ***************** */
float zahl1,zahl2;
char op;
float erg;
/* *********** Prototypen ************************************ */
void ini_ser0(void);

main()
{
      ini_ser0();             // Initialisierung der seriellen Schnittstelle
while(1)
{
        printf("Geben Sie zwei Zahlen ein: \n");
        printf("Zahl1 Operator Zahl2 <return>, Komma mit Punkt\n");
        scanf("%f %c %f",&zahl1,&op,&zahl2);
        printf("Zahl 1    = %7.2f \n",zahl1);
        printf("Operator = %c\n",op);
        printf("Zahl 2    = %7.2f \n",zahl2);
        printf("--------------------------\n");
        switch(op)
        {
                case '+':
                        printf("Ergebnis = %7.2f\n",(float)zahl1+zahl2);
                        break;
                case '-':
                        printf("Ergebnis = %7.2f\n",(float)zahl1-zahl2);
                        break;
                case '/':
                        printf("Ergebnis = %7.2f\n",(float)zahl1/zahl2);
                        break;
                case '*':
                        printf("Ergebnis = %7.2f\n",(float)zahl1*zahl2);
                        break;
        }  /* end switch */
        printf("\n");
} /* end while */
} /* end main */
void ini_ser0(void)     // 80C517A-Board, 12MHz, sb8V2006-humerboard.at
{     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 */
}
```

5.2.3 Ein- Ausgaben mit Pointer

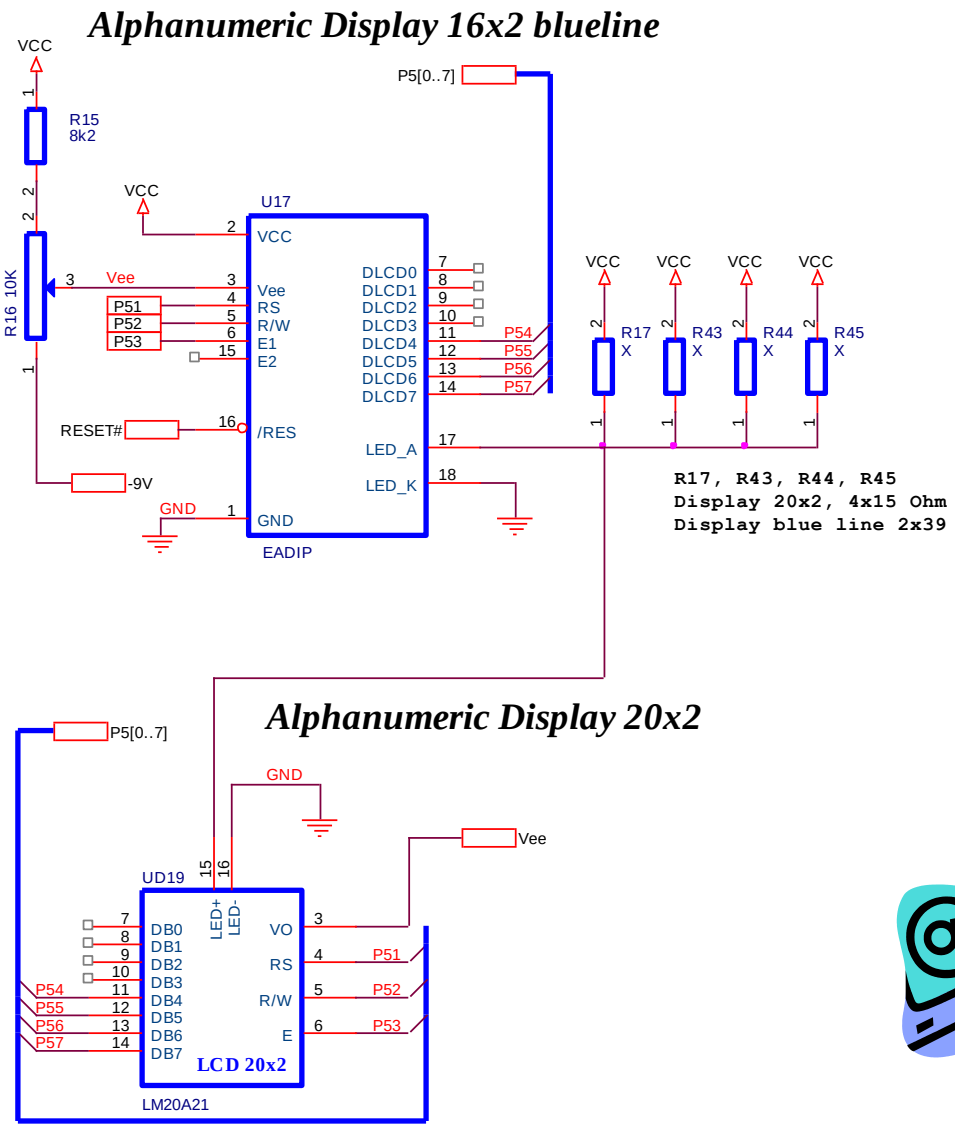
```
/* ************************************************************ */
/* ******* BEISPIELPROGRAMM FÜR DIE SERIELLE SCHNITTSTELLE *************** */
/* ******* 4CHELI, 11.11.2010, HUMER ***************************** */
/* ** DATEI: RS232.C *********** C51 UVISION C517ABOARD ************* */
/* ************************************************************ */

#include <reg517a.h>                          // SFR Registerdefinitionen
#include <stdio.h>                            // Standard_In_Out Bibl.
#include <string.h>                           // String Bibliothek
idata char text[20];                        //Array mit 10 char Variablen
char i;
char *zeiger;
void ini_ser0(void);                          // Prototypendeklaration

main()
{
      ini_ser0();             // Funktionsaufruf Initialisierung RS232
      while(1)
      {
            printf("Hello world\n");
            printf("Geben Sie ein Wort ein:\n");
            scanf("%s",&text);
            printf("Das Wort lautet: %s\n",text);
            printf("Anzahl der Buchstaben=%d\n",strlen(text));
            printf("************************************************\n");
            zeiger=&text[strlen(text)-1];

            for(i=strlen(text);i>0;i--)
            {
                  printf("%c",*(zeiger--));
            }

            printf("\n");

// Vergleich
/*          for(i=strlen(text);i>0;i--)
                        {
                              printf("%c",text[i-1]);
                        } */


            printf("\n");
      }
}

void ini_ser0(void)
{                       // 80C517A-Board, 12MHz, sb8V2006-humerboard.at
      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 */
}
```

6 LCD

6.1 Allgemeines

Am Board SC8v2006 wurde ein alphanum. Display mit 20x2 Zeichen verwendet. Der standardisierte Microcontroller von Hitachie am LCD garantiert für viele Bibliotheksfunktionen im Internet. Für die Programmierung der Boards verwende ich eine eigene, bitorientierte Version. Diese ist zwar zeitlich sehr langsam – garantiert aber bezüglich ihres Einsatzes maximale Universalität. Das Board kann mit 2 unterschiedlichen Bauformen bestückt sein (U17 oder UD19).

6.2 Schaltplan



Im Schaltplan ist der verwendete 4-bit Betrieb erkennbar.

6.3 Bibliotheksfunktionen LCD

Für das LCD stehen in der Library folgende Funktionen zur Verfügung:

```
void warte(unsigned int msec);
void busy_lcd();
void init_lcd();
void blank_lcd();
void char_lcd(char zeile, char spalte, char ascii);
void print_lcd(char zeile, char spalte, char *str);
void graf_lcd (char adresse, char muster);
void spezial_graf(void);
void def_balken (void);
void pos(char zeile,char spalte);
void lcd_write(char c);
```

6.4 Anbindung für andere 8051er - Systeme

Im Programmteil lcd.c müssen am Beginn nur die Anschlüsse definiert werden.

```
/* LCD-Anschluesse an Ports */

  sbit RS = P5^1;
  sbit RW = P5^2;
  sbit EN = P5^3;
  sbit D4 = P5^4;
  sbit D5 = P5^5;
  sbit D6 = P5^6;
  sbit D7 = P5^7;
```

6.5 Beispiel

6.5.1 Hello world am LCD

```
/* ************************************************ */
/* **** Datei: lcd_word.c, humer, Datum: 25.11.10 ******* */
/* **** Einfache Textausgabe an LCD ******************** */
/* ************************************************ */

#include <reg517a.h>
#include "lcd.h"                    // Datei im Arbeitsverzeichnis

// Datei lcd.c muss sich ebenfalls im AV befinden und im Projekt
// eingebunden sein

void myinit(void);


main()
{
      myinit();
      char_lcd(1,18,'A');           // Ausgabe string  1. Zeile, 18. Spalte
      print_lcd(1,1,"Hello world");
      print_lcd(2,3,"hi freaks!");

      while(1);

}


void myinit(void)
{
      init_lcd();
      blank_lcd();

}
```

6.5.2 Datenausgabe am LCD

```
/* ********************************************** */
/* *** Datei: lcd_daten.c, humer, Datum: 25.11.10 ******* */
/* *** Einfache Textausgabe an LCD ******************** */
/* ********************************************** */


#include <reg517a.h>
#include "lcd.h"
#include <stdio.h>

void myinit(void);
idata float var_a;
data int var_b;
idata char text_fix[]={"Minki"};                    // String
idata char text_buffer[21];                         // Ausgabebuffer


main()
{
      myinit();
      var_a=3.141592;
      var_b=-123;
      print_lcd(1,1,text_fix);                      // Stringausgabe
      sprintf(text_buffer,"a=%6.3f",var_a);         // schreibe in Buffer
      print_lcd(1,7,text_buffer);                   // Ausgabe vom Buffer
      sprintf(text_buffer,"b=%05d",var_b);          // schreibe in Buffer
      print_lcd(2,4,text_buffer);                   // Ausgabe in Buffer
      while(1);

}

void myinit(void)
{
      init_lcd();                             // Initialisierung LCD
      blank_lcd();                            // Gesamtes LCD löschen

}
```

6.6 LCD –Ausgabe von Graphikzeichen am DOT-LCD

6.6.1 Allgemeines

Für diesen Displaytyp können die ersten 8 ASCII-Zeichen des LCD selbst definiert werden. Diese Zeichen sind nicht druckbar und darum für den USER freigeschaltet. Sie müssen nur in den Speicher des Dispalys geladen werden. 2 x 8 Zeichen sind bereits in der lcd.c Datei bereitgestestellt.

6.6.2 Zeichengruppe 1

```
void spezial_graf(void)
{

 /*** Symbol Nr. 0hex ***/
  graf_lcd ( 0, 0x00);      /*   . . . . .   */
  graf_lcd ( 1, 0x00);      /*   . . . . .   */
  graf_lcd ( 2, 0x00);      /*   . . . . .   */
  graf_lcd ( 3, 0x00);      /*   . . . . .   */
  graf_lcd ( 4, 0x00);      /*   . . . . .   */
  graf_lcd ( 5, 0x00);      /*   . . . . .   */
  graf_lcd ( 6, 0x00);      /*   . . . . .   */
  graf_lcd ( 7, 0x00);      /*   . . . . .   */

 /*** Symbol Nr. 1hex ***/
  graf_lcd ( 8, 0x00);      /*   . . . . .   */
  graf_lcd ( 9, 0x00);      /*   . . . . .   */
  graf_lcd (10, 0x04);      /*   . . * . .   */
  graf_lcd (11, 0x1E);      /*   * * * * .   */
  graf_lcd (12, 0x1F);      /*   * * * * *   */
  graf_lcd (13, 0x1E);      /*   * * * * .   */
  graf_lcd (14, 0x04);      /*   . . * . .   */
  graf_lcd (15, 0x00);      /*   . . . . .   */

 /*** Symbol Nr. 2hex ***/
  graf_lcd (16, 0x00);      /*   . . . . .   */
  graf_lcd (17, 0x00);      /*   . . . . .   */
  graf_lcd (18, 0x04);      /*   . . * . .   */
  graf_lcd (19, 0x0F);      /*   . * * * *   */
  graf_lcd (20, 0x1F);      /*   * * * * *   */
  graf_lcd (21, 0x0F);      /*   . * * * *   */
  graf_lcd (22, 0x04);      /*   . . * . .   */
  graf_lcd (23, 0x00);      /*   . . . . .   */

 /*** Symbol Nr. 3hex ***/
  graf_lcd (24, 0x00);      /*   . . . . .   */
  graf_lcd (25, 0x04);      /*   . . * . .   */
  graf_lcd (26, 0x0E);      /*   . * * * .   */
  graf_lcd (27, 0x1F);      /*   * * * * *   */
  graf_lcd (28, 0x0E);      /*   . * * * .   */
  graf_lcd (29, 0x0E);      /*   . * * * .   */
  graf_lcd (30, 0x0E);      /*   . * * * .   */
  graf_lcd (31, 0x00);      /*   . . . . .   */
```

```
 /*** Symbol Nr. 4hex ***/
  graf_lcd (32, 0x00);       /*   . . . . .   */
  graf_lcd (33, 0x0E);       /*   . * * * .   */
  graf_lcd (34, 0x0E);       /*   . * * * .   */
  graf_lcd (35, 0x0E);       /*   . * * * .   */
  graf_lcd (36, 0x1F);       /*   * * * * *   */
  graf_lcd (37, 0x0E);       /*   . * * * .   */
  graf_lcd (38, 0x04);       /*   . . * . .   */
  graf_lcd (39, 0x00);       /*   . . . . .   */

 /*** Symbol Nr. 5hex  representiert ENTER Taste    ***/
  graf_lcd (40, 0x1F);       /*   * * * * *   */
  graf_lcd (41, 0x01);       /*   . . . . *   */
  graf_lcd (42, 0x01);       /*   . . . . *   */
  graf_lcd (43, 0x05);       /*   . . * . *   */
  graf_lcd (44, 0x0D);       /*   . * * . *   */
  graf_lcd (45, 0x1F);       /*   * * * * *   */
  graf_lcd (46, 0x0C);       /*   . * * . .   */
  graf_lcd (47, 0x04);       /*   . . * . .   */

 /*** Symbol Nr. 6hex ***/
  graf_lcd (48, 0x04);       /*   . . * . .   */
  graf_lcd (49, 0x0E);       /*   . * * * .   */
  graf_lcd (50, 0x1F);       /*   * * * * *   */
  graf_lcd (51, 0x0E);       /*   . * * * .   */
  graf_lcd (52, 0x0E);       /*   . * * * .   */
  graf_lcd (53, 0x0E);       /*   . * * * .   */
  graf_lcd (54, 0x0E);       /*   . * * * .   */
  graf_lcd (55, 0x0E);       /*   . * * * .   */


   /*** Symbol Nr. 7hex ***/
  graf_lcd (56, 0x08);       /*   . * . . .   */
  graf_lcd (57, 0x14);       /*   * . * . .   */
  graf_lcd (58, 0x08);       /*   . * . . .   */
  graf_lcd (59, 0x03);       /*   . . . * *   */
  graf_lcd (60, 0x04);       /*   . . * . .   */
  graf_lcd (61, 0x04);       /*   . . * . .   */
  graf_lcd (62, 0x03);       /*   . . . * *   */
  graf_lcd (63, 0x00);       /*   . . . . .   */
  busy_lcd();
}
```

6.6.3 Zeichengruppe 2

```
void def_balken(void)
{

 /*** Symbol Nr. 0hex ***/
  graf_lcd ( 0, 0x00);       /*   . . . . .   */
  graf_lcd ( 1, 0x00);       /*   . . . . .   */
  graf_lcd ( 2, 0x00);       /*   . . . . .   */
  graf_lcd ( 3, 0x00);       /*   . . . . .   */
  graf_lcd ( 4, 0x00);       /*   . . . . .   */
  graf_lcd ( 5, 0x00);       /*   . . . . .   */
  graf_lcd ( 6, 0x00);       /*   . . . . .   */
  graf_lcd ( 7, 0x00);       /*   . . . . .   */

 /*** Symbol Nr. 1hex ***/
  graf_lcd ( 8, 0x10);       /*   * . . . .   */
  graf_lcd ( 9, 0x10);       /*   * . . . .   */
  graf_lcd (10, 0x10);       /*   * . . . .   */
  graf_lcd (11, 0x10);       /*   * . . . .   */
  graf_lcd (12, 0x10);       /*   * . . . .   */
  graf_lcd (13, 0x10);       /*   * . . . .   */
  graf_lcd (14, 0x10);       /*   * . . . .   */
  graf_lcd (15, 0x10);       /*   *. . . . .  */

 /*** Symbol Nr. 2hex ***/
  graf_lcd (16, 0x18);       /*   * * . . .   */
  graf_lcd (17, 0x18);       /*   * * . . .   */
  graf_lcd (18, 0x18);       /*   * * . . .   */
  graf_lcd (19, 0x18);       /*   * * . . .   */
  graf_lcd (20, 0x18);       /*   * * . . .   */
  graf_lcd (21, 0x18);       /*   * * . . .   */
  graf_lcd (22, 0x18);       /*   * * . . .   */
  graf_lcd (23, 0x18);       /*   * * . . .   */


 /*** Symbol Nr. 3hex ***/
  graf_lcd (24, 0x1C);       /*   * * * . .   */
  graf_lcd (25, 0x1C);       /*   * * * . .   */
  graf_lcd (26, 0x1C);       /*   * * * . .   */
  graf_lcd (27, 0x1C);       /*   * * * . .   */
  graf_lcd (28, 0x1C);       /*   * * * . .   */
  graf_lcd (29, 0x1C);       /*   * * * . .   */
  graf_lcd (30, 0x1C);       /*   * * * . .   */
  graf_lcd (31, 0x1C);       /*   * * * . .   */


 /*** Symbol Nr. 4hex ***/
  graf_lcd (32, 0x1E);       /*   * * * * .   */
  graf_lcd (33, 0x1E);       /*   * * * * .   */
  graf_lcd (34, 0x1E);       /*   * * * * .   */
  graf_lcd (35, 0x1E);       /*   * * * * .   */
  graf_lcd (36, 0x1E);       /*   * * * * .   */
  graf_lcd (37, 0x1E);       /*   * * * * .   */
  graf_lcd (38, 0x1E);       /*   * * * * .   */
  graf_lcd (39, 0x1E);       /*   * * * * .   */
```

```
 /*** Symbol Nr. 5hex ***/
  graf_lcd (40, 0x1F);       /*   * * * * *   */
  graf_lcd (41, 0x1F);       /*   * * * * *   */
  graf_lcd (42, 0x1F);       /*   * * * * *   */
  graf_lcd (43, 0x1F);       /*   * * * * *   */
  graf_lcd (44, 0x1F);       /*   * * * * *   */
  graf_lcd (45, 0x1F);       /*   * * * * *   */
  graf_lcd (46, 0x1F);       /*   * * * * *   */
  graf_lcd (47, 0x1F);       /*   * * * * *   */

 /*** Symbol Nr. 6hex ***/
  graf_lcd (48, 0x04);       /*   . . . . .   */
  graf_lcd (49, 0x04);       /*   * . . . .   */
  graf_lcd (50, 0x04);       /*   * * . . .   */
  graf_lcd (51, 0x04);       /*   * * * . .   */
  graf_lcd (52, 0x1F);       /*   * * * * .   */
  graf_lcd (53, 0x0E);       /*   * * * . .   */
  graf_lcd (54, 0x04);       /*   * * . . .   */
  graf_lcd (55, 0x00);       /*   * . . . .   */
  busy_lcd();

}
```

6.7 Beispiel

6.7.1 Sonderzeichen am LCD darstellen

```
/* ************************************************************** */
/* ***************  SONDERZEICHEN AM LCD DARSTELLEN      *************** */
/* **** DATEI: LCD_GRAPHIK.C     DATUM: 2.12.10         HUMER   4CHELI **** */
/* ************************************************************** */


#include <reg517a.h>
#include "lcd.h"

void myinit(void);
data char i;

main()
{
      myinit();                          // Initialisieren
      def_balken();                      // 8 Sonderzeichen laden
// alternativ spezial_graf();
      for(i=0;i<8;i++)
      char_lcd(2,3+i,i);
      while(1);
}

void myinit(void)
{
      init_lcd();
      blank_lcd();
}
```

6.7.2 Zähler mit LCD-Balkenanzeige

```
/* Binärzähler 0..100 Balkenanzeige - Pseudographik am LCD */

#include <reg517a.h>
#include <stdio.h>
#include "lcd.h"
void mygraphic(char zeile, char wert);
void myinit();
bit flag;
idata char counter;
idata char buffer[21];

void sec_ticker(void) interrupt 2
{     flag=1;
      counter++;
      if(counter==101) counter=0;
}

void main()
{     myinit();
      while(1)
      {
      if (flag)
            {
            flag=0;
            mygraphic(2,counter);
            sprintf(buffer,"Count: %03bd",counter);
            print_lcd(1,4,buffer);
            }
      }
}

void myinit()
{
      init_lcd(); blank_lcd(); def_balken();
      EX1=1;      IT1=1;      EAL=1;                  // Interruptfreigabe
}

void mygraphic(char zeile, char wert)
{     char i;
      for(i=wert/5;i>0;i--)
            {char_lcd(zeile,i,5);}
                              // Anzahl der vollen Elemente beschreiben
      char_lcd(zeile,(wert/5)+1,wert%5);              // nicht volle Anzeige
      for(i=(wert/5)+2;i<21;i++)                      // Rest löschen
            char_lcd(zeile,i,0);
}
```

6.7.3 Digitaluhr (Std., Min., Sek. – Anzeige am LCD und RS232)

```
/* ************************************************************** */
/* **********************D I G I T A L U H R*********************** */
/* *************** Datei: vier.c, Humer, 14.10.2010 ****************** */
/* ************************************************************** */

#include <reg517a.h>
#include <stdio.h>
#include "lcd.h"
char std,min,sec,tastenwert;
bit flag;
char buffer[21];
void lesetasten(void) interrupt 0
{     tastenwert=P7&0x0F;
      switch (tastenwert)
      {     case 0: EX1=1; break;
            case 1:     EX1=0; break;
            case 2: std=0;min=0;sec=0;flag=1;break;
            case 3: if(std<24)
                              {std++; flag=1;}break;
            case 4: if(std>0)
                              {std--; flag=1;}break;
            case 5: if(min<59)
                              {min++; flag=1;}break;
            case 6: if(min>0)
                              {min--; flag=1;}break;
            case 7: if(sec<59)
                              {sec++; flag=1;}break;
            case 8: if(sec>0)
                              {sec--; flag=1;}break;
      }
}
void secticker(void) interrupt 2
{     sec++;
      if(sec==60) {sec=0; min++;}
      if(min==60) {min=0; std++;}
      if(std==24) {std=0;}
      flag=1;
}
void init517a(void);
main()
{     init517a();
      while(1)
      {
            if(flag)
            {
                  printf("Time=%02bd:%02bd:%02bd\r",std,min,sec);
                  sprintf(buffer,"%02bd:%02bd:%02bd",std,min,sec);
                  print_lcd(1,6,buffer);
                  flag=0;
}   }   }
void init517a(void)
{     EX1=1; //enable Externer Interrupt Nr. 1
      IT1=1; // neg. Flake für Interrupt Nr. 1
      EAL=1; // Generelle Interruptfreigabe
      EX0=1;
      IT0=1;      init_lcd(); blank_lcd();
}
```

6.7.4 Digitaluhr – Anzeige auch Zehntelsekunden

```
/* **********************D I G I T A L U H R*************************** */
/* ***** DATEI: DIGITAL_UHR_ZEHNTEL.C, HUMER, 04.11.2010 ****************** */
/* ********************************************************************** */

#include <reg517a.h>
#include <stdio.h>
#include "lcd.h"
char std,min,sec,zehntel,tastenwert;
bit flag;
char buffer[21];
void lesetasten(void) interrupt 0
{     tastenwert=P7&0x0F;
      switch (tastenwert)
      {     case 0: EX1=1; break;
            case 1:     EX1=0; break;
            case 2: std=0;min=0;sec=0;flag=1;break;
            case 3: if(std<24)
                              {std++; flag=1;}break;
            case 4: if(std>0)
                              {std--; flag=1;}break;
            case 5: if(min<59)
                              {min++; flag=1;}break;
            case 6: if(min>0)
                              {min--; flag=1;}break;
            case 7: if(sec<59)
                              {sec++; flag=1;}break;
            case 8: if(sec>0)
                              {sec--; flag=1;}break;
      }
}
void secticker(void) interrupt 12
{     zehntel++;
      if(zehntel==10) {zehntel=0; sec++;}
      if(sec==60) {sec=0; min++;}
      if(min==60) {min=0; std++;}
      if(std==24) {std=0;}
      flag=1;
}
void init517a(void);
main()
{     init517a();
      while(1)
      {
            if(flag)
            {

      printf("\rTime=%02bd:%02bd:%02bd:%01bd",std,min,sec,zehntel);

      sprintf(buffer,"%02bd:%02bd:%02bd:%01bd",std,min,sec,zehntel);
                  print_lcd(1,6,buffer);
                  flag=0;
}   }   }
void init517a(void)
{     EX5=1;                              //enable Externer Interrupt Nr. 5
      EAL=1;                              // Generelle Interruptfreigabe
      EX0=1;
      IT0=1; init_lcd(); blank_lcd();
}
```

6.7.5 Reaktionszeitmessgerät

Hier ist ein kleines Projekt realisiert. Der Anwender drückt eine beliebige Taste, dann erscheint an der 7Segm-Anzeige die Taste welche möglichst schnell gedrückt werden soll. Die Reaktionszeit wird dabei in Zehntelsekunden gemessen.

```
/* ****************   REAKTIONSZEITMESSGERÄT    ******************* */
/* DATEI: RANDOM.C     DATUM: 11.11.10       HUMER   4CHELI     * */
/* ************************************************************** */
#include <reg517a.h>
#include <stdio.h>
idata int i,zsec,j;
char taste,tsoll;
code char tab[16]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,
                   0x88,0x83,0xA7,0xA1,0x86,0x8E};
code char tabm[16]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x18,
                   0x08,0x03,0x27,0x21,0x06,0x0E};
bit flag;
void write7Seg(char SSeg, char dp);
void readtasten (void) interrupt 0              // Interruptfunktion Taste
{
      flag=1;
      taste=P7&0x0F;
}
void ticker (void) interrupt 12                 // Ext. Int. 5 = 10Hz
{
      zsec++;
}
void myinit (void);
void mywarte (char anzahl);

main()
{
      myinit();                                 // Initialisierung
      while(1)
      {     printf("Bitte eine Taste drücken\n");     // warte auf Tastendr.
            while(!flag);
            flag=0;
mywarte(TL0);
            tsoll=TH0&0x0F;               // Für den Zufall - Timer 0
            write7Seg((tsoll),1);         // Ermittlung der richtigen Taste
            EX5=1;
            printf("Warte auf Tastendruck\n");        // warte auf Tastendr.
            while(!flag);
            printf("Taste gedrückt\n");
            flag=0;

if(taste==tsoll)
                  {
printf("Wertung: Dauer=%d00 msec\n",zsec);
                        zsec=0; EX5=0;
                  }
                  else
                  {
                  EX5=0;                                  // Int. 5 sperren
zsec=0;                                 // zsec rücksetzen
                  printf("Falsche Taste, ungültig, nicht genügend!\n");
                  }
}}
```

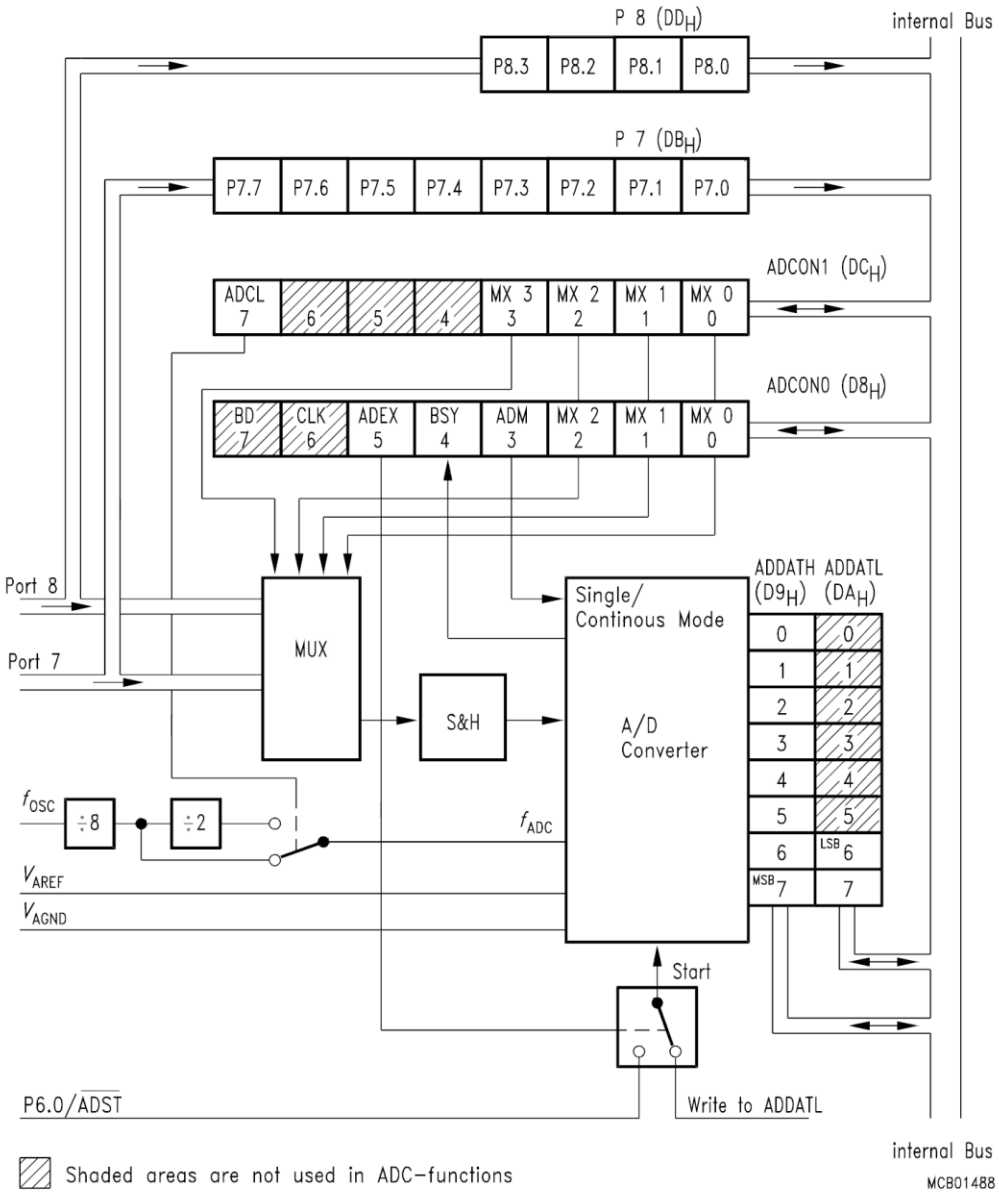
```
void write7Seg(char SSeg, char dp)
{           P6=P6&0x3F; // Balken und 7Seg sperren
            P6=P6|0x40; // 7Seg freigeben
            if(dp) P4=tab[SSeg];
            else P4=tabm[SSeg];
            P6=P6&0x3F; // Balken und 7Seg sperren
}

void myinit (void)
{           TMOD=0x01; // 16bit Timer Mode
            IT0=1; // neg. Flanke Interrupt 0
            EX0=1; // Freigabe Externer Interrupt 0 (Taste)
            EAL=1; // Generelle Interruptfreigabe
            TR0=1; // Timer0 run
            EX5=0;
}

void mywarte (char anzahl)
{
      for(i=0; i<anzahl; i++)
      for(j=0; j<2000; j++);
}
```

6.7.6 Wie lange hat jemand eine Taste gedrückt?

```
#include <reg517a.h>
#include <stdio.h>

sbit porttaste = P3^2;

char taste;
bit flag1=0;
idata int T0ueb;
idata int dauer;
void myinit(void);

void pressbutton (void) interrupt 0
{
      TR0=1;
      taste=P7&0x0F;
      flag1=1;
}

void T0ueberlauf(void) interrupt 1
{
      T0ueb++;
}

main()
{
      myinit();
      while(1)
      {
            if(flag1)
            {
                        //while(((P3&0x04)>>2)==0);// xxxx xTxx (P3.2)
                  while(!porttaste);
                  TR0=0;
                  flag1=0;
                  dauer = (int)((TH0/4)+65.5*T0ueb);
                  printf("Taste= %bd, dauer= %05d msec \n", taste, dauer);
                  T0ueb=0;
                  TH0=0;
            }
      }
}

void myinit(void)
{
      TR0=0; // Timer sperren
      TH0=0; // Timer rücksetzen
      TL0=0;
      TMOD=0x01; //Timer initialisieren
      ET0=1;            // Timer0 Interr. freigabe
      IT0=1;            // neg. Flanke Tastatur
      EX0=1;            // Interrupt 0 Freigabe
      EAL=1;            // Generelle Freigabe
}
```

7 Analog-Digital-Umsetzer

7.1 Blockschaltbild AnalogDigitalUmsetzer C517A



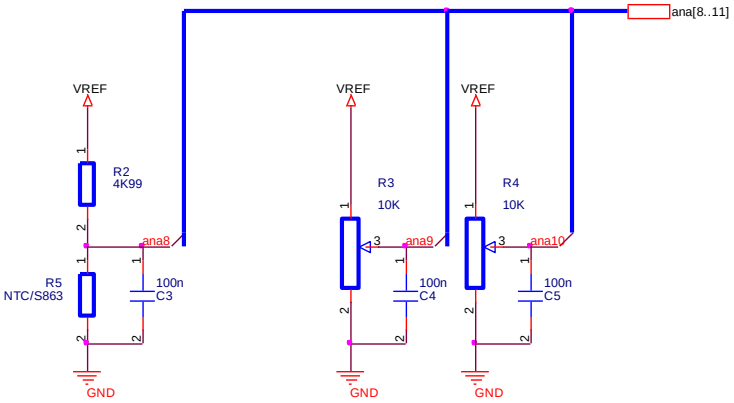
Blockschaltbild ADU C517A

7.2 Funktionsaufruf read_adc()

```
int read_adc(char kanal)
{
      ADCON1=kanal; // Kanal wählen
      ADDATL=0; // Start ADC
      while(BSY); // Warte bis fertig
      return((ADDATH<<2)+(ADDATL>>6));// 10 bit Wert retour
}
```

7.3 Beschaltung von NTC, Poti 1 und 2

ADU - Einheit



NTC-Temperatursensor ***Potentiometer***

7.4 Beispiel

7.4.1 Messung der Spannung an 2 Potis – Ausgabe am LCD

```
/* ********************************************************************** */
/* ******** ANALOG DIGITAL UMSETZER, EINFACHE STRUKTUR   ****************** */
/* ***************** DATEI: ADU_SIMPLE.C, HUMER, 9.12.2010 ************** */
/* ********************************************************************** */

#include <reg517a.h>
#include "lcd.h"
#include <stdio.h>

void myinit(void);
int read_adc(char kanal);
idata char buffer[21];
main()
{
      myinit();
      while(1)
      {
sprintf(buffer,"ch 9: %04d=5.3fV",read_adc(9),read_adc(9)*5.0/1024);
print_lcd(1,1,buffer);
sprintf(buffer,"ch10: %04d=%5.3fV",read_adc(10),read_adc(10)*5.0/1024);
print_lcd(2,1,buffer);
      }
}

void myinit(void)
{
      init_lcd();
      blank_lcd();

}
int read_adc(char kanal)
{
      ADCON1=kanal; // Kanal wählen
      ADDATL=0; // Start ADC
      while(BSY); // Warte bis fertig
      return((ADDATH<<2)+(ADDATL>>6));// 10 bit Wert retour
}
```

7.4.2 Ermittlung der Spannung am POTI – Visualisierung als Balken

```
/* ********************************************************************** */
/* ******  Analog Digital Umsetzer, LCD- Balkenanzeige  ******************** */
/* ***************** Datei: adubalken.c, Humer, 9.12.2010 ***************** */
/* ********************************************************************** */

#include <reg517a.h>
#include "lcd.h"
#include <stdio.h>

void myinit(void); // Prototyp
int read_adc(char kanal), aduwert;
idata char buffer[21],balkenwert;
void mygraphic(char zeile, char wert);

main()
{     myinit();
      while(1)
      {     aduwert=read_adc(9);    // Einlesen Kanal9 = Potentiometer
            balkenwert=(int)aduwert/10;
// Ganzzahligen Prozentwert ermitteln
            sprintf(buffer,"ch 9: %04d = %5.3fV",aduwert,aduwert*5.0/1024);
            print_lcd(2,1,buffer);
            mygraphic(1,(char)balkenwert);            // Funktionsaufruf
}   }

void myinit(void)  // Initialisierung
{     init_lcd();
      blank_lcd();
      def_balken();

}

int read_adc(char kanal)  // Lesen ADU
{     ADCON1=kanal; // Kanal wählen
      ADDATL=0; // Start ADC
      while(BSY); // Warte bis fertig
      return((ADDATH<<2)+(ADDATL>>6));          // 10 bit Wert retour
}

void mygraphic(char zeile, char wert)           // Balkenanzeige 0..100%
{     char i;
      for(i=wert/5;i>0;i--)
            {char_lcd(zeile,i,5);}
                        // Anzahl der vollen Elemente beschreiben
      char_lcd(zeile,(wert/5)+1,wert%5);        // nicht volle Anzeige
      for(i=(wert/5)+2;i<21;i++)  // Rest löschen
            char_lcd(zeile,i,0);
}
```

7.4.3 Lauflicht „kit“ Geschwindigkeit mit Poti gesteuert

```
/* ********************************************** */
/* ** NAME:HUMER, DATUM: 16.12.2010, DATEI: ADU_KIT.C *** */
/*  ****   LAUFLICHT-KIT GESCHW. UEBER POTI       ********* */
/* ********************************************** */

#include <reg517a.h>
void delay(void);
unsigned int potiwert;
int read_adc(char kanal);
void main(void)
{
      char k;
      P4=0x01;                                          //0000 0001
      while(1)
      {
            P4=0x01;
            for(k=1;k<9;k++)
            {
                  delay();
                  P4<<=1;                               // shift left
            }

            P4=0x80;
            for(k=1;k<9;k++)
            {
                  delay();
                  P4>>=1;                               // shift rechts
            }

      }
}

void delay(void)
{     int i;
      potiwert=read_adc(9);
      for(i=0;i<(200+5*potiwert);i++);
}

int read_adc(char kanal)
{
      ADCON1=kanal;                                 // Kanal wählen
      ADDATL=0;                                     // Start ADC
      while(BSY);                                   // Warte bis fertig
      return((ADDATH<<2)+(ADDATL>>6));              // 10 bit Wert retour
}
```

7.4.4 Beispiel Ermittlung der Spannung am NTC

```
// Einlesen der Spannung am NTC Widerstand
// Datum: 1.4.11, Humer. 4AHELI/4NHELT

#include <reg517a.h>                        // SFR Definitionen
#include "lcd.h"                            // LCD Prototypendeklarationen
#include <stdio.h>                          //LIB  Std. In-Out
void myinit();
code float r_ref = 4990;                    // Referenzwiderstand
int readadc(char kanal);                    //Prototyp mit Rückgabewert
idata float r_temp;                         // Widerstand NTC
idata int ntcwert;            // Int Variable Ort indirekt Adr. Speicher
idata char lcdbuffer[21];                   // Bufferinitalisierung

main()
{     myinit(); // Initialisierung
      while(1)
      {
            ntcwert=readadc(8);                 // Einlesen der NTC-Spannung
            sprintf(lcdbuffer,"%3d",ntcwert);
            print_lcd(1,1,lcdbuffer);           // Ausgabe des ADU-Wertes
            sprintf(lcdbuffer,"%5.3f[V]",ntcwert*5.0/1024);
            print_lcd(1,5,lcdbuffer);           // Ausgabe der NTC-Spannung
            r_temp = r_ref*ntcwert/(1023-ntcwert);
            sprintf(lcdbuffer,"%5.0f[Ohm]",r_temp);
            print_lcd(2,1,lcdbuffer);     // Ausgabe des NTC-Widerstandes
      }
}

int readadc(char kanal)
{
      ADCON1=kanal;                         // Kanalwahl
      ADDATL=0;                             // Start ADU
      while(BSY);                           // Warte bis ADU fertig
      return((ADDATH<<2)+(ADDATL>>6));
}

void myinit(void)                           // Funktion myinit
{     init_lcd();                           // LCD Initialisierung
      blank_lcd();                          // Löschen, Anzeige an der LCD
}
```

7.4.5 Temperaturmessung NTC

```
// Einlesen der Spannung am NTC Widerstand+Temperatur
// Datum: 7.4.11, Humer. 4CHELI

#include <reg517a.h>  // SFR Definitionen
#include "lcd.h"  // LCD Prototypendeklarationen
#include <stdio.h>      //LIB  Std. In-Out
#include <math.h> //LIB Mathematik
void myinit();
code float r_ref = 4990; // Referenzwiderstand
int readadc(char kanal); //Prototyp
idata float r_temp;     // Widerstand NTC
idata int ntcwert; // Int Variable Ort indirekt Adr. Speicher
idata char lcdbuffer[21]; // Bufferinitalisierung
code float        a=0.0012814814,
                  b=0.000236678791,
                  c=0.0000000908736406;
idata float lnr,temp;

main()
{     myinit(); // Initialisierung
      while(1)
      {
            ntcwert=readadc(8);                 // Einlesen der NTC-Spannung
            sprintf(lcdbuffer,"%3d",ntcwert);
            print_lcd(1,1,lcdbuffer);           // Ausgabe des ADU-Wertes
            sprintf(lcdbuffer,"%5.3f[V]",ntcwert*5.0/1024);
            print_lcd(1,5,lcdbuffer);           // Ausgabe der NTC-Spannung
            r_temp = r_ref*ntcwert/(1023-ntcwert);
            sprintf(lcdbuffer,"%5.0f[Ohm]",r_temp);
            print_lcd(2,1,lcdbuffer);     // Ausgabe des NTC-Widerstandes
            lnr = log(r_temp);
            temp = (1.0/(a+b*lnr+c*lnr*lnr*lnr));
            temp-=273.2;
            sprintf(lcdbuffer,"%4.1f",temp);
            print_lcd(2,14,lcdbuffer);
}}

int readadc(char kanal)
{
      ADCON1=kanal; // Kanalwahl
      ADDATL=0; // Start ADU
      while(BSY); // Warte bis ADU fertig
      return((ADDATH<<2)+(ADDATL>>6));
}

void myinit(void) // Funktion myinit
{     init_lcd();       // LCD Initialisierung
      blank_lcd();      // Löschen, Anzeige an der LCD
}
```

7.4.6 Messung der Temperatur im 300ms Intervall

```
// Einlesen der Spannung am NTC Widerstand+Temperatur
// Datum: 7.4.11, Humer. 4CHELI
// Messungsintervall 300ms, (Timer0), serielle Schnittst.+LCD

#include <reg517a.h> // SFR Definitionen
#include "lcd.h"   // LCD Prototypendeklarationen
#include <stdio.h>       //LIB  Std. In-Out
#include <math.h>  //LIB Mathematik

void myinit();
bit flag;
code float r_ref = 4990; // Referenzwiderstand
int readadc(char kanal); //Prototyp
idata float r_temp;     // Widerstand NTC
idata int ntcwert; // Int Variable Ort indirekt Adr. Speicher
idata char lcdbuffer[21],counter; // Bufferinitalisierung
code float  a=0.0012814814,
                  b=0.000236678791,
                  c=0.0000000908736406;
idata float lnr,temp;

void ticker(void) interrupt 1
{
      TH0=0x3C;
      TL0=0xAF;
      counter++;
      if(counter==6)
            {counter=0;flag=1;}
}

main()
{     myinit(); // Initialisierung
      while(1)
      {
            if(flag)
            {
            ntcwert=readadc(8); // Einlesen der NTC-Spannung
            sprintf(lcdbuffer,"%3d",ntcwert);
            print_lcd(1,1,lcdbuffer); // Ausgabe des ADU-Wertes
            sprintf(lcdbuffer,"%5.3f[V]",ntcwert*5.0/1024);
            print_lcd(1,5,lcdbuffer); // Ausgabe der NTC-Spannung
            r_temp = r_ref*ntcwert/(1023-ntcwert);
            sprintf(lcdbuffer,"%5.0f[Ohm]",r_temp);
            print_lcd(2,1,lcdbuffer); // Ausgabe des NTC-Widerstandes
            lnr = log(r_temp);
            temp = (1.0/(a+b*lnr+c*lnr*lnr*lnr));
            temp-=273.2;
            sprintf(lcdbuffer,"%4.1f",temp);
            printf("%4.1f\n",temp);
            print_lcd(2,14,lcdbuffer);
            flag=0;
            }
}}
```

```
int readadc(char kanal)
{
      ADCON1=kanal; // Kanalwahl
      ADDATL=0; // Start ADU
      while(BSY); // Warte bis ADU fertig
      return((ADDATH<<2)+(ADDATL>>6));
}
void myinit(void) // Funktion myinit
{     init_lcd();       // LCD Initialisierung
      blank_lcd();      // Löschen, Anzeige an der LCD
      EAL=1;
      ET0=1;
      TR0=1;
      TMOD=0x011;
```

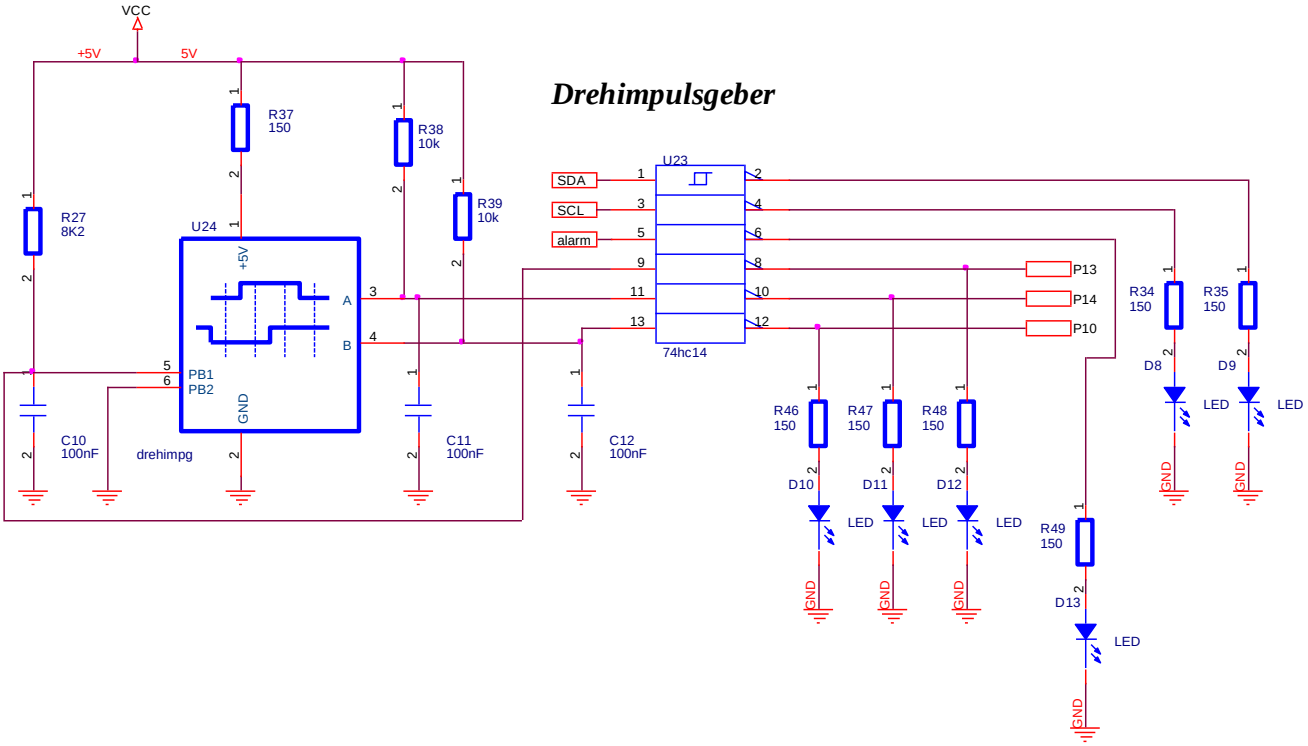
```
}
```

8 Drehimpulsgeber

8.1 Allgemeines

Der Drehimpulsgeber auch Ein-Knopfbedienung genannt ist eine Eingabemöglichkeit bei Microcontrollersystemen. Die 2 Mäandersignal sin nicht entprellt und müssen extra behandelt werden. Eine Visualisierung der Pegel von A und B erfolgt durch LEDs D10 und D11. Der Schalter hat aber auch noch eine Tastenfunktion (push) – Pegel durch die LED12 visualisiert. Die Signale wurden durch RC-Kombination in der Prellung gedämpft und stehen voll dem Interruptsystem zur Verfügung. Somit können die Signale push, A und B im Polling oder per Interrupt betrieben werden.

8.2 Schaltplan



Interruptquelle	Externer Interrupt Nr.	Software Nummer
Tastatur	0	0
Sekundentakt	1	2
Drehimpulsgeber A	2	9
Drehimpulsgeber B	3	10
RTC	4	11
Takt 10Hz	5	12
Drehimpulsgeber push	6	13

Tabelle Einbindung des Drehimpulsgebers in der Interruptstruktur

8.3 Beispiel

8.3.1 Drehimpulsgeber

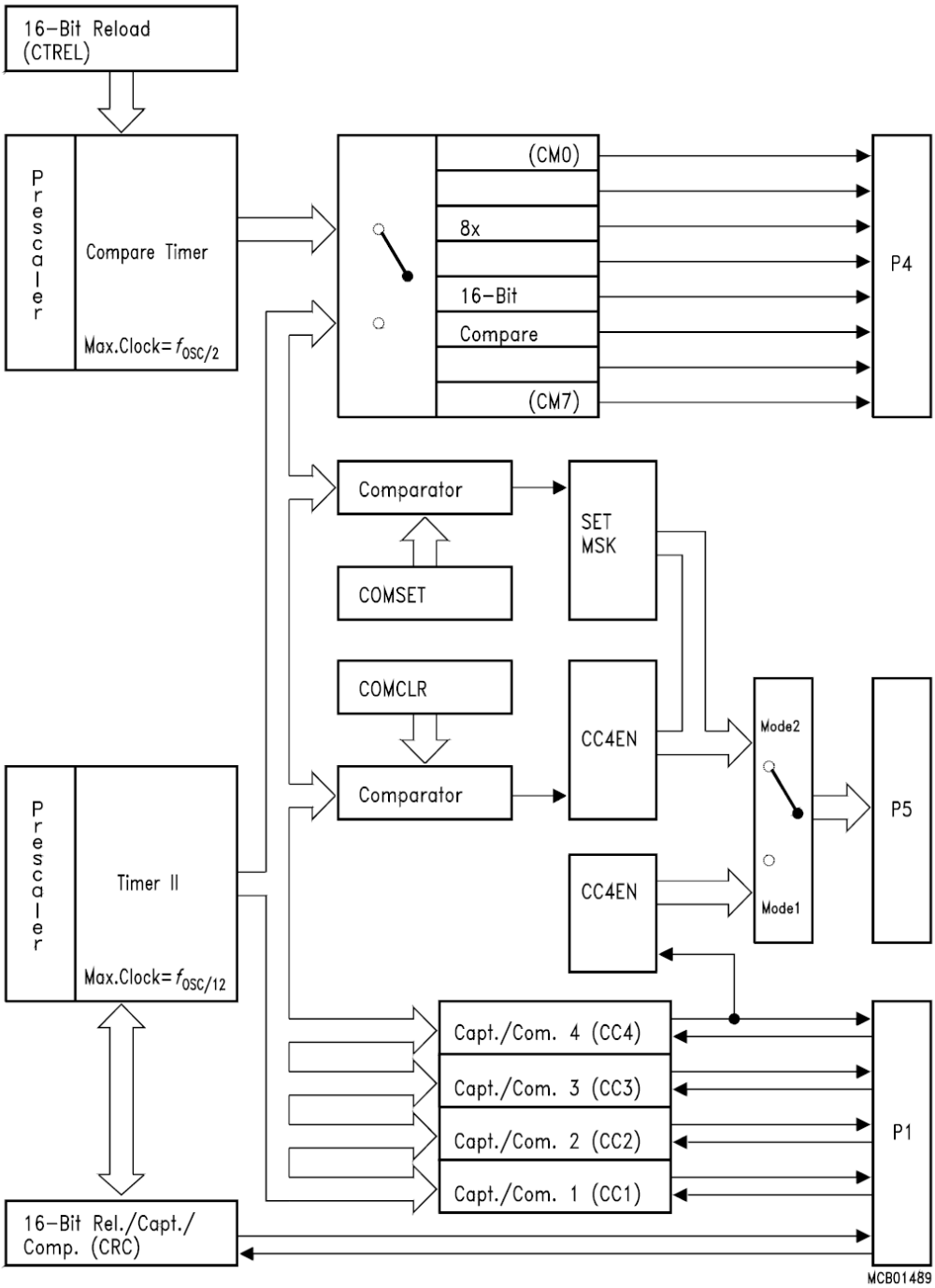
```
// ************************************************************
// Programmbeispiel: Drehimpulsgeber
// Datum: 20.9.2005: mpa
// Drehimpulsgeber wird abgefragt und als laufender Pfeil am LCD angezeigt
// Eingangsleitungen sind P1.0/P1.4 ( Richtung ) und P1.3 ( Knopf )

#include<reg517a.h>
#include<stdio.h>
#include<lcd.h>

// Globale variablen (beginnen mit g...)
char gpos;                          // position des Zeichens auf dem LCD
char galt;                          // Alter Wert des Drehimpulsgebers


// Deklaration der Funktionen
char Richtung(char);                // Bestimmt die Richtung des Drehgebers

main()
{
char  in;                                   // 8 bit Eingangsvariable
char rich;                                  // Richtung des Drehimpulsgeber

P1 = 0xFF;                                  // Port P1 als Eingang setzen
gpos = 8;                                   // Startposition des Zeigers
init_lcd();                                 // LCD initialisieren
spezial_graf();               // Eigene Zeichen auf LCD schreiben (0-7)
blank_lcd();                          // LCD loeschen

while(1)                              // Endlosschleife
      {
      in = P1;                        // Port 1 einlesen

      if(in & 0x08 )                  // P1.3 ausmaskieren mit 00001000
            { print_lcd(1,1,"Knopf gedrueckt"); }
      else
            { print_lcd(1,1,"               "); }

      char_lcd(2,gpos,' ');           // altes Zeichen loeschen

      rich = Richtung(in);    // Richtung des Drehimpulsgebers bestimmen
      if(rich == 1 )                        // falls rechts gedreht
            {
            gpos = gpos + 1;                // Position im eins erhoehen
            if(gpos > 20) { gpos = 20; }  // Limit setzten
            }
      if(rich == 2 )                        // falls links gedreht
            {
            gpos = gpos - 1;                // Position im eins erhoehen
            if(gpos < 1 ) { gpos =  1; }  // Limit setzten
            }
      char_lcd(2,gpos,0x03);
// Zeichen (Pfeil oben) auf neue Position schreiben
      }                                     // of while
}
```

```
// Funktionen
// Richtung: bestimmt die Richtung in die der Drehgeber bewegt wurde
// Weicht der neue Wert um mehr als eine Stelle vom alten Wert ab, wird
keine Aenderung zurueckgegeben
// return:  0...keine Aenderung
//                1...Rechts
//                2...Links
char Richtung(char neu)
{
char ret;               // rueckgabe wert

                        // Drehimpulsgeber erzeugt folgende Muster an P1 :
                        // xxx0xxx0
                        // xxx0xxx1
                        // xxx1xxx1
                        // xxx1xxx0

neu = neu & 0x11;       // P1.0 und P1.4 ausmaskieren
switch(neu)       // Unterschied zwichen altem und neuem Wert abfragen
  {
  case 0x00:
       if(galt == 0x00)       // wenn galt gleich war
              {ret = 0;}
       if(galt == 0x10)       // wenn galt eins darunter war hochzaehlen
              {ret = 1;}
       if(galt == 0x01)       // wenn galt eins darueber war runterzaehlen
              {ret = 2;}
  break;
  case 0x01:
       if(galt == 0x01)       // wenn galt gleich war
              {ret = 0;}
       if(galt == 0x00)       // wenn galt eins darunter war hochzaehlen
              {ret = 1;}
       if(galt == 0x11)       // wenn galt eins darueber war runterzaehlen
              {ret = 2;}
  break;
  case 0x11:
       if(galt == 0x11)       // wenn galt gleich war
              {ret = 0;}
       if(galt == 0x01)       // wenn galt eins darunter war hochzaehlen
              {ret = 1;}
       if(galt == 0x10)       // wenn galt eins darueber war runterzaehlen
              {ret = 2;}
  break;
  case 0x10:
       if(galt == 0x10)       // wenn galt gleich war
              {ret = 0;}
       if(galt == 0x11)       // wenn galt eins darunter war hochzaehlen
              {ret = 1;}
       if(galt == 0x00)       // wenn galt eins darueber war runterzaehlen
              {ret = 2;}
  break;
    default:
            return(0);
// Keine Aenderung weil kein neuer code oder ein code uebersprungen wurde
    break;
  }         // of switch
galt = neu;                   // Neuen galt wert bestimmen
return(ret);
}
```

9 PWM-Captur/Compare

9.1 Beispiel PWM

```
// ***************************************************************
// Einlesen des Potentiometers R4 über ADC Eingelesener Datenwert steuert
// Helligkeit des LED-Balken und 7-Segment an Port 4 mit PWM
// Datum: 8.12.2005: mpa (Dr. Manfred Pauritsch)
```



```
#include<reg517a.h>
#include<stdio.h>
#include<lcd.h>

                                        // Globale variablen (beginnen mit g...)
char gTXT[25];                                  // Allgemeiner Textbuffer
                                                // Deklaration der Funktionen
int aduwert(char);

main()
{
int     adu1;                                   // Ergebnisse der AD-Umsetzer
float pwmwert;                                  // pwm in prozent
init_lcd();                                     // LCD initialisieren
blank_lcd();                                    // LCD loeschen
            // Konfigurieren der Compare-Timer-Einheit ////////////////////
CTCON  = 0x03;          // Einstellen des Vorteilers f=fclk/16
CMSEL  = 0xFF;          // CM0-CM7 gehört zu Compare-Timer im PWM modus
CTRELH = 0xFC;          // Reload wert für 1024 Stufen =
CTRELL = 0x00;          // 65536 - 1024 = 64512 = 0xFC00
CMEN   = 0xFF;
// CM0-CM7 Register freigeben und Ausgang an Port P4.1-P4.7  weiterleiten
P6 = 0xFF;
// LE aktivieren mit 11111111. ( P6.7 = LEDBar = 1 , P6.6 = 7Seg = 1)

while(1)                                                // Endlosschleife
      {
      adu1 = aduwert(10);                               // ADU  Kanal 10 lesen
      pwmwert = (1024.0 - adu1)*100/1024.0;
                                        // Berechnung des Prozentwertes der PWM

      CMH0 = CTRELH + (adu1 >> 8);
            // Obere 8 bit extrahieren und zu Reloadwert zuzaehlen
      CML0 = CTRELL + (0x00FF & adu1);
            // Untere 8 bit extrahieren und zu Reloadwert zuzaehlen
      CMH1 = CMH0;                                      // Reload Wert kopieren
      CML1 = CML0;                                      // Reload Wert kopieren
      CMH2 = CMH0;                                      // Reload Wert kopieren
      CML2 = CML0;                                      // Reload Wert kopieren
      CMH3 = CMH0;                                      // Reload Wert kopieren
      CML3 = CML0;                                      // Reload Wert kopieren
      CMH4 = CMH0;                                      // Reload Wert kopieren
      CML4 = CML0;                                      // Reload Wert kopieren
      CMH5 = CMH0;                                      // Reload Wert kopieren
      CML5 = CML0;                                      // Reload Wert kopieren
      CMH6 = CMH0;                                      // Reload Wert kopieren
      CML6 = CML0;                                      // Reload Wert kopieren
      CMH7 = CMH0;                                      // Reload Wert kopieren
      CML7 = CML0;                                      // Reload Wert kopieren

      sprintf(gTXT,"PWM=%4.1f Prozent ",pwmwert);
                                                        // Messwerte ausgeben
      print_lcd(1,1,gTXT);
                                        // String auf LCD ausgeben. Zeile 1
      warte(100);                                                 // Warte
      }
}
```

```
                                    // Starten einer Analog-Digital-Umsetzung
                                    // Parameter:     char...Eingabekanal (0-11)
                                    // Rueckgabe:     ADU-Wert int

int aduwert(char kanal)
{
      int result;
      unsigned int low,high;
                        // unsigned, damit nicht arithmetisch geschobe wird
      ADCON1=kanal;     // ADC Kanal festlegen 0-11
      ADDATL = 0;       // Schreibzugriff auf ADDTAL startet Umsetzung
while(BSY)  {;}       // Warte bis BSY Flag Null wird ( Bit 5  in ADCON0 )
low  = ADDATL >> 6;
                  // ADDATL.7,ADDATL.6 (D1,D0) an Position 1 und 0 schieben
high = ADDATH << 2;
            // ADDATH.7 bis ADDATH.0 (D9-D2) an Position 9 bis 2 schieben
result = (int)(high+low);
                                        // Ergebnis zusammensetzen
return(result);
}
```

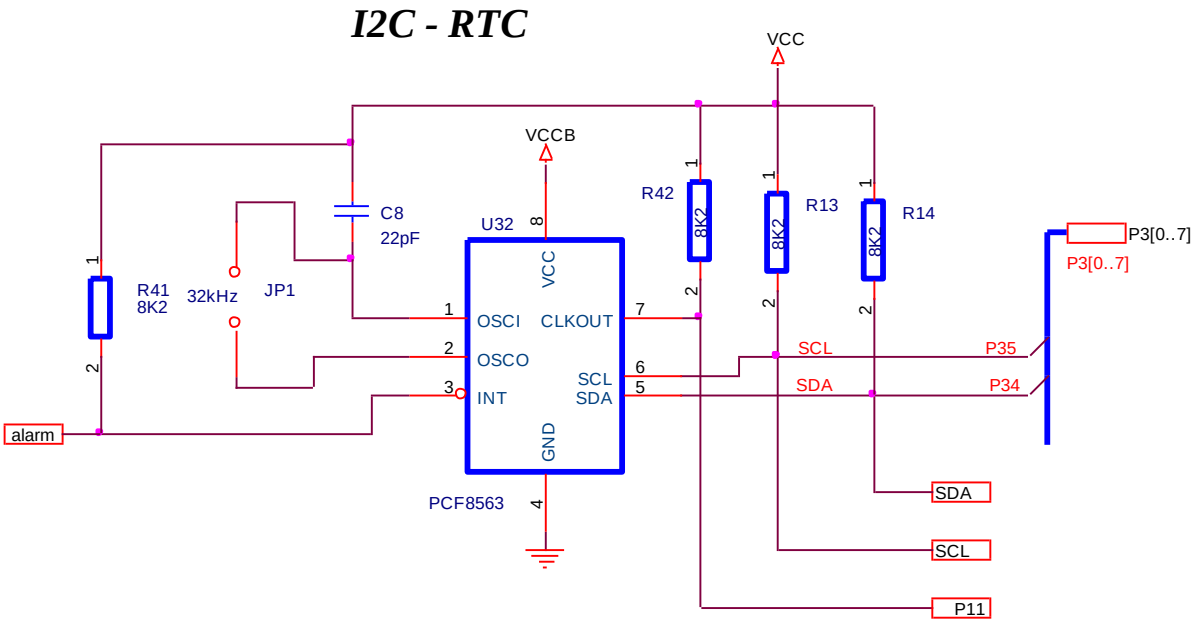
10 I2C-System

10.1 Allgemeines

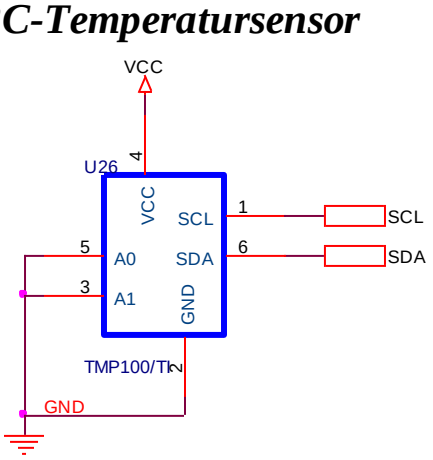
Der I2C-Bus ist in der Literatur allgemein sehr gut beschrieben, hier werde ich nur auf die Programmierung eingehen. In der bereitgestellten Bibliothek sind in der Datei i2c.h folgende Funktionen definiert:

```
                                        // Headerfile zu I2C-Funktionen
void i2c_init(void);
void i2c_start(void);
void i2c_stop(void);
bit i2c_write(unsigned char);
unsigned char i2c_read(char send_ack);
void delay_time(unsigned char);
void I2C_Schreiben(unsigned char, unsigned char, unsigned char);
unsigned char  I2C_Lesen(unsigned char, unsigned char );
```

10.1.1 Schaltplan RTC



10.1.2 Schaltplan Temperatursensor



10.1.3 Blockschaltbild RTC PCF8563

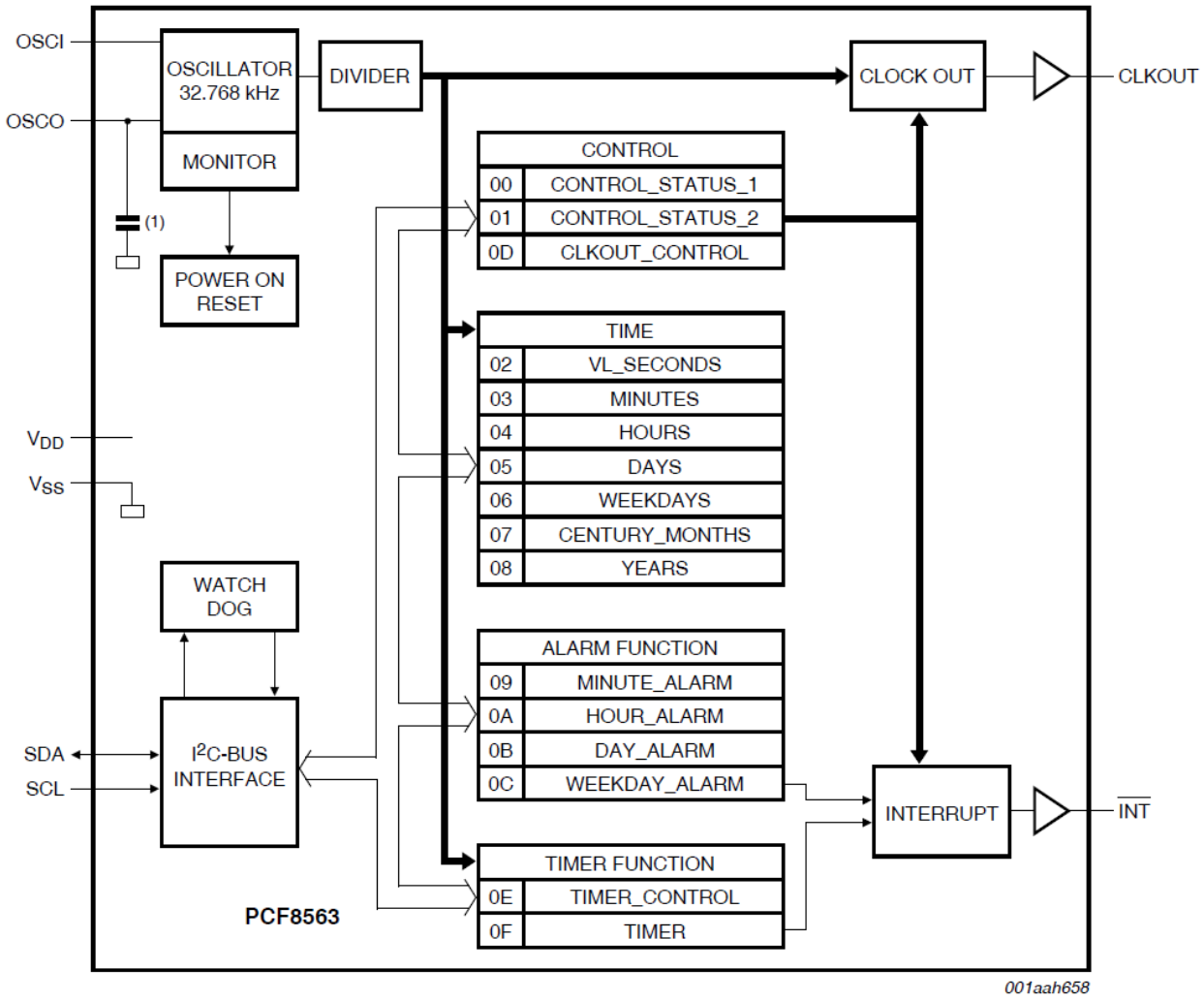


Bild Blockschaltbild des Bausteins PCF8563 (I2C-RTC)

10.2 Beispiele

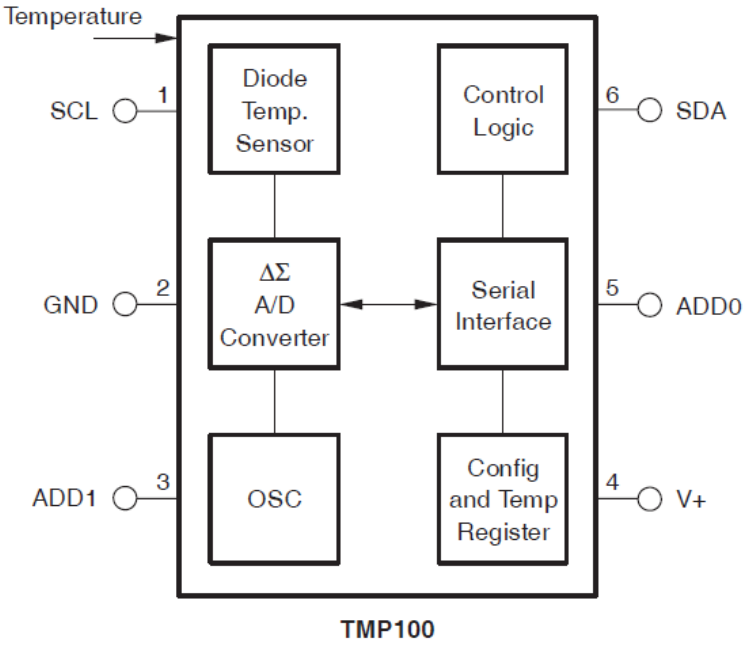
10.2.1 RTC

```
// ************************************************************
// Programmbeispiel: Echtzeituhr PCF8563 ueber I2C-Bus
// Datum: 21.9.2005: mpa (Dr. Manfred Pauritsch)

#include<reg517a.h>
#include<stdio.h>
#include<lcd.h>
#include<i2c.h>

#define RTC 0xA2              // PCF8563 I2C - Adresse 10100010 (schreiben)
#define TASTENPORT P7               // Tastaturdekoder ist an Port 7
sbit  TASTENDRUCK = P3^2;
                              // Tastedruck Puls an P3^2 (INT0) aktiv LOW
                              // Globale variablen (beginnen mit g...)
char gTXT[25];                // Allgemeiner Textbuffer
char gTaste;                  // Nummer der gedrueckten Taste
char gNeueTaste;              // Signalisiert neuen Tastendruck

// Deklaration der Funktionen
main()
{
unsigned char Daten;                          // Allgemeine Variable
unsigned char Taste;                          // Taste
unsigned char bcd;                            // BCD-Zahl
unsigned char hz,he,mz,me;    // Stunden Zehner,Einer Minuten Zehner,einer
unsigned char h,m,s;                          // Stunden, Minuten,
Sekunden

i2c_init();                                   // I2C-Bus initialisieren
init_lcd();                                   // LCD initialisieren
blank_lcd();                                  // LCD loeschen


sprintf(gTXT,"Uhrzeit (hh:mm)");              // String beschreiben
print_lcd(1,1,gTXT);              // String auf LCD ausgeben. Zeile 1
sprintf(gTXT,"          __:__   ");           // String beschreiben
print_lcd(2,1,gTXT);              // String auf LCD ausgeben. Zeile 2


while(TASTENDRUCK)      {;}         // Warte bis Taste gedrueckt (LOW) wird
hz = TASTENPORT & 0x0F;
                  // Einlesen der Tasten und Maskieren der Portbits 0-3
sprintf(gTXT,"          %1bx_:__   ",hz);         // Stunden Zehner schreiben
print_lcd(2,1,gTXT);          // String auf LCD ausgeben. Zeile 2
bcd = hz << 4;
            // 4 Bit nach links schieben um BCD Zahl zu erzeugen (Zehner)
while(TASTENDRUCK==0)   {;}
            // Warte bis Taste wieder losgelassen wird (HIGH)

while(TASTENDRUCK)      {;}   // Warte bis Taste gedrueckt (LOW) wird
he = TASTENPORT & 0x0F;
                  // Einlesen der Tasten und Maskieren der Portbits 0-3
sprintf(gTXT,"          %1bx%1bx:__   ",hz,he);
                  // Stunden Zehner, Stunden Einer schreiben
```

```
print_lcd(2,1,gTXT);
                  // String auf LCD ausgeben. Zeile 2
bcd = bcd | he;
                                    // Einer zu BCD Zahl hinzufuegen
I2C_Schreiben(RTC, 0x04, bcd);  // Register 04:Stunden schreiben in BCD
while(TASTENDRUCK==0)   {;}
                        // Warte bis Taste wieder losgelassen wird (HIGH)

while(TASTENDRUCK)      {;}
                        // Warte bis Taste gedrueckt (LOW) wird
mz = TASTENPORT & 0x0F;
                  // Einlesen der Tasten und Maskieren der Portbits 0-3
sprintf(gTXT,"          %1bx%1bx:%1bx_   ",hz,he,mz);
                  // Minuten Zehner schreiben
print_lcd(2,1,gTXT);
                  // String auf LCD ausgeben. Zeile 2
bcd = mz << 4;
            // 4 Bit nach links schieben um BCD Zahl zu erzeugen (Zehner)
while(TASTENDRUCK==0)   {;}
                        // Warte bis Taste wieder losgelassen wird (HIGH)

while(TASTENDRUCK)      {;}
                        // Warte bis Taste gedrueckt (LOW) wird
me = TASTENPORT & 0x0F;
                  // Einlesen der Tasten und Maskieren der Portbits 0-3
sprintf(gTXT,"          %1bx%1bx:%1bx%1bx  ",hz,he,mz,me);
                              // Minuten Zehner, Minuten Einer schreiben
print_lcd(2,1,gTXT);
                              // String auf LCD ausgeben. Zeile 2
bcd = bcd | me;
                              // Einer zu BCD Zahl hinzufuegen
I2C_Schreiben(RTC, 0x03, bcd);
                              // Register 03:   Minuten schreiben in BCD
while(TASTENDRUCK==0)   {;}
                  // Warte bis Taste wieder losgelassen wird (HIGH)


I2C_Schreiben(RTC, 0x02, 0x00);
                  // Register 02:   Sekunden auf Null setzen


sprintf(gTXT,"                          ");            // Zeile loeschen
print_lcd(1,1,gTXT);                  // String auf LCD ausgeben. Zeile 1
print_lcd(2,1,gTXT);                  // String auf LCD ausgeben. Zeile 2

while(1)                              // Endlosschleife
      {
      h = I2C_Lesen(RTC,0x04);        // Stunden auslesen
      h = h & 0x3F;                   // Stunden masieren
      m = I2C_Lesen(RTC,0x03);        // Minuten auslesen
      m = m & 0x7F;                   // Minuten masieren
      s = I2C_Lesen(RTC,0x02);        // Minuten auslesen
      s = s & 0x7F;                   // Minuten masieren
                                 // Zeit ausgeben, 2xBCD kann dierekt als HEX
      sprintf(gTXT,"Zeit:    %2bx:%02bx:%b02x",h,m,s);
                                 // Fuehrende Nullen bei Min. und Sek.
      print_lcd(1,1,gTXT);
                                 // String auf LCD ausgeben. Zeile 1
      }
}
```

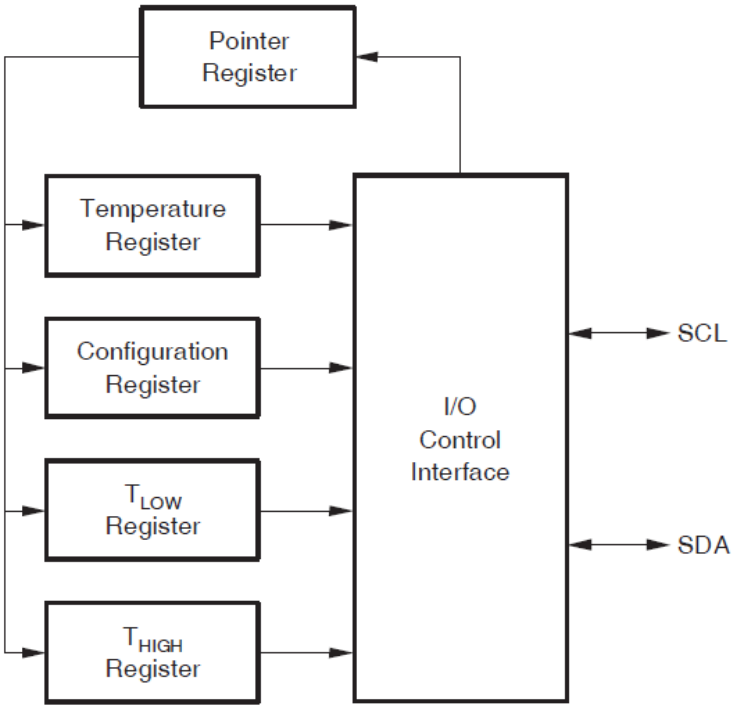
10.2.2 Temperatursensor TMP100

10.2.3 Blockschaltbild Temperatursensor TMP100 (TI)



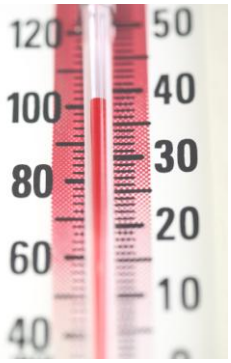
TMP100

10.2.4 Interne Register des Sensors



10.2.5 Programmbeispiel TMP100

```
// ************************************************************
// PROGRAMMBEISPIEL:       TEMPERATURMESSUNG MIT TMP100 UEBER I2C-BUS
//                                  AUFLOESUNG 12BIT
// DATUM: 12.10.2005: MPA



#include<reg517a.h>
#include<stdio.h>
#include<lcd.h>
#include<i2c.h>

#define TMP100_Adr 0x90     // TMP100 I2C - Adresse 10010000 (schreiben)


// Globale variablen (beginnen mit g...)
char gTXT[25];              // Allgemeiner Textbuffer


// Deklaration der Funktionen
main()
{
int  Temp ;                         // Temperaturwerte als Integer (16bit)
unsigned char TempH,TempL;          // Temperaturwerte als Byte (8bit)
float Temperatur;                   // Skalierter Temperaturwert als float

i2c_init();                         // I2C-Bus initialisieren
init_lcd();                         // LCD initialisieren
blank_lcd();                        // LCD loeschen


sprintf(gTXT,"TMP100 Sensor:");     // String beschreiben
print_lcd(1,1,gTXT);                // String auf LCD ausgeben. Zeile 1


i2c_start();                        // Startbit senden
i2c_write(TMP100_Adr);              // Slave Adresse
i2c_write(0x01);                    // Configuration Register schreiben
i2c_write(0x60);                    // Setze 12bit Aufloesung (R1,R0 = 1,1)
i2c_stop();                         // Stopbit senden


                                    // weiter auf der nächsten Seite
```

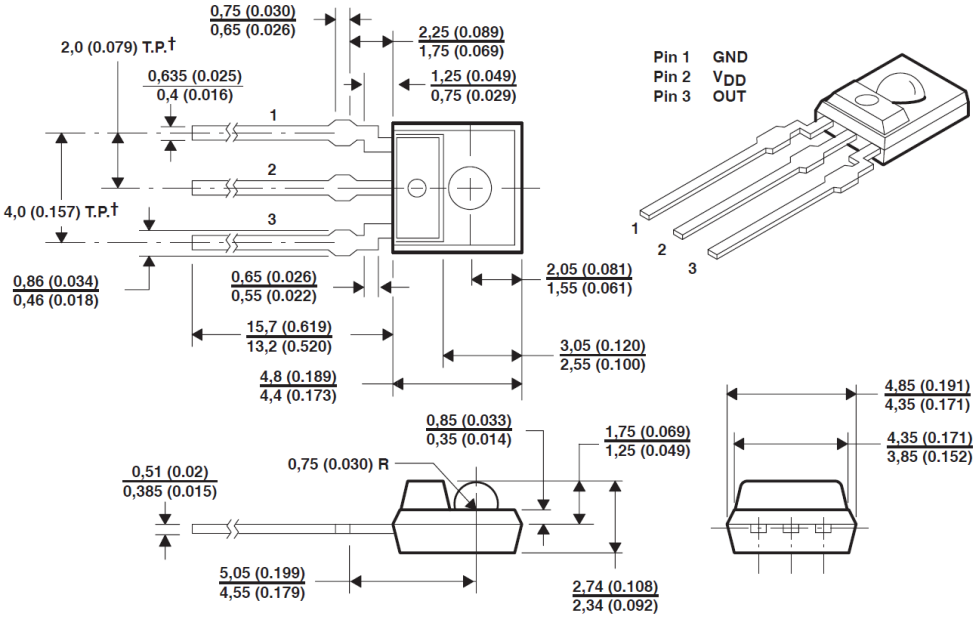


```
while(1)                                                    //
Endlosschleife
      {
      i2c_start();                  // Startbit senden
      i2c_write(TMP100_Adr);        // Slave Adresse, schreiben
      i2c_write(0x00);              // Temp Register schreiben
      i2c_stop();                   // Stopbit senden
      i2c_start();                  // Startbit senden
      i2c_write(TMP100_Adr  | 0x01);
// Slave Adresse mit gesetztem LSB zum lesen
      TempH = i2c_read(1);          // Daten lesen und   ACK schicken
      TempL = i2c_read(1);          // Daten lesen und   ACK schicken
      i2c_stop();                   // Stopbit senden

      Temp = TempH << 8;      // High Byte nach links schieben (obere 8 bit)
      Temp = Temp | TempL;          // Low Byte dazugeben (untere 8 bit)
      Temp = Temp >> 4;
// 4 bit nach rechts schieben unter Beibehaltung des Vorzeichens

      Temperatur = Temp*0.0625;     // Skalieren des Temperaturwertes


      sprintf(gTXT,"%6.4f Grad",Temperatur);    // Temp ausgeben
      print_lcd(2,1,gTXT);
// String auf LCD ausgeben. Zeile 1
      }
}
```

11 Licht Frequenz Umsetzer (LFU)

11.1 Allgemeines

11.1.1 Sensor TSL235

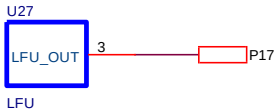


† True position when unit is installed.

ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

11.1.2 Schaltplan

Licht-Frequenzumsetzer



Der Sensor ist direkt mit dem Portpin T2 verbunden.

11.1.3 Kennlinien des Sensors

TYPICAL CHARACTERISTICS

OUTPUT FREQUENCY vs IRRADIANCE

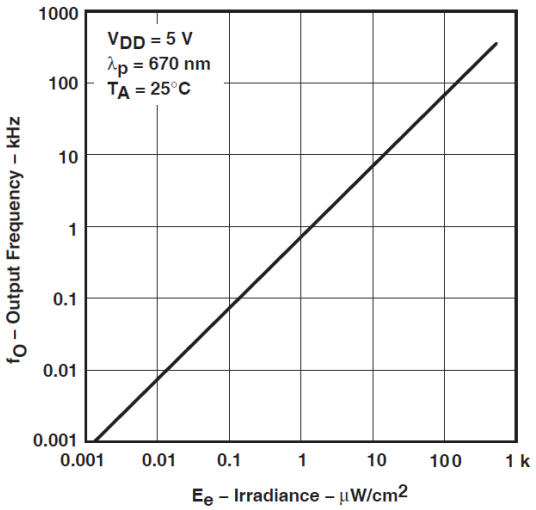


Figure 1

PHOTODIODE SPECTRAL RESPONSIVITY

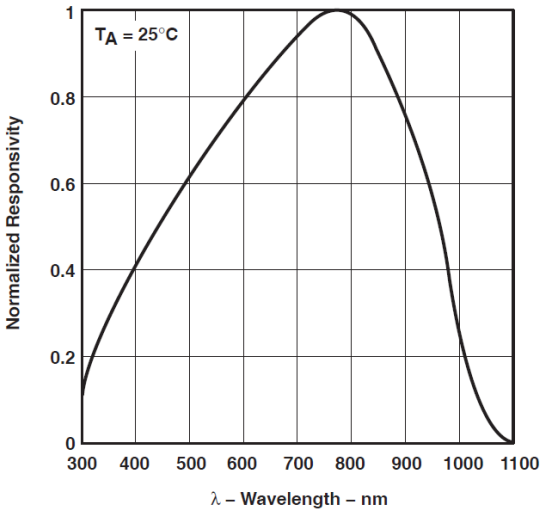
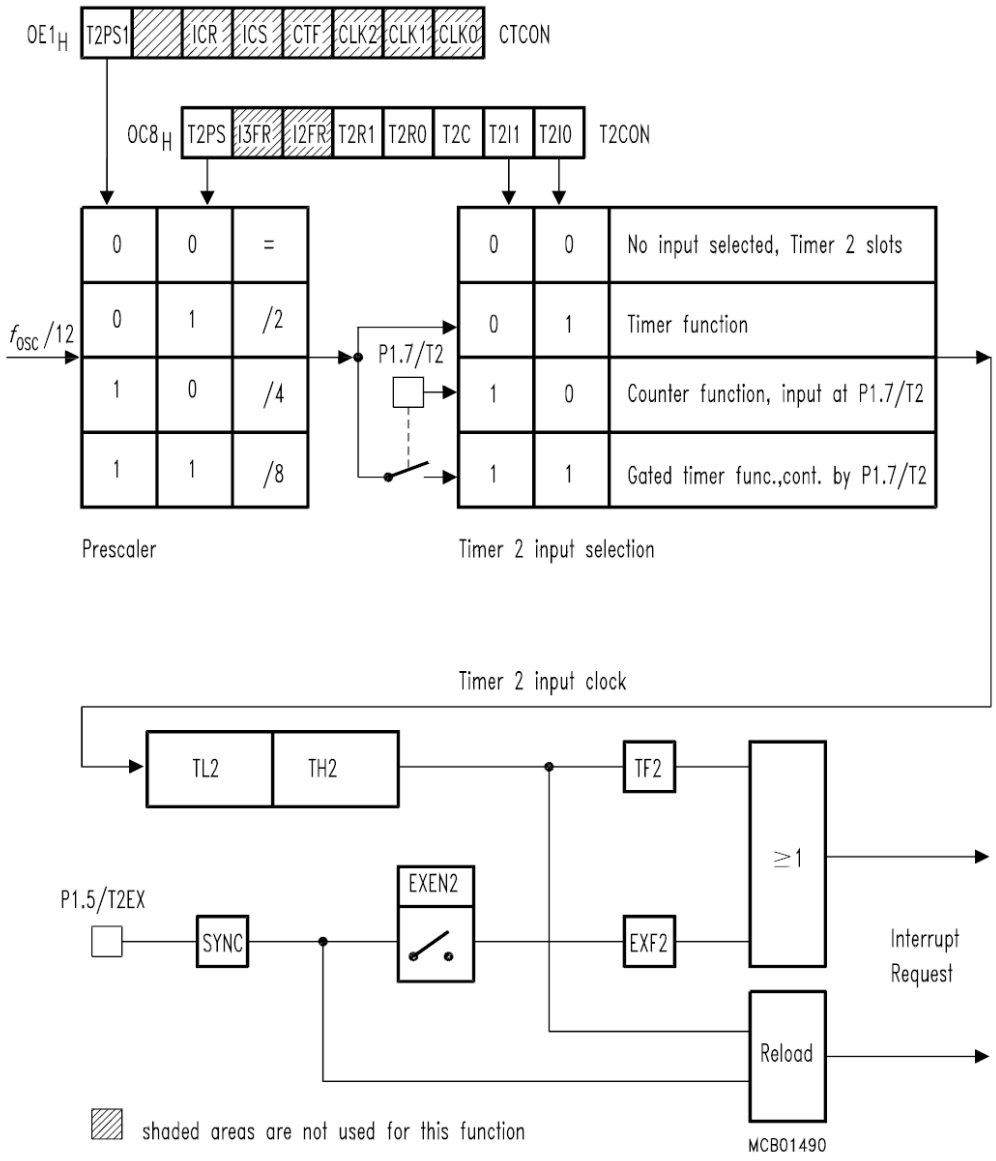


Figure 2

11.1.4 Blockschaltbild Timer 2



11.2 Beispiel

11.2.1 Lichtmessung

```
// ************************************************************
// Programmbeispiel:       Lichtmessung mit TSL235
// Messung beruht auf Frequenzmessung mit automatischer Bereichswahl
// Automatische Umschaltung erfolgt bei 65536 Hz bzw. 60000 Hz
// Das erforderliche Tor-Signal wird von Timer0 (10ms) abgeleitet
//                                    Gezählt wird mit Timer2.
//                                    Datum: 6.11.2005: mpa
//
//Pulse|||||||||||||||||||||||||||||||||||| in Timer2 gezaehlt
//                                  ____________________
//  Tor zur Messung     __________|                    |________ von Timer1
// abgeleitet

#include<reg517a.h>
#include<stdio.h>
#include<lcd.h>

// Globale variablen (beginnen mit g...)
char gTXT[25];                  // Allgemeiner Textbuffer
int g10ms;                      // Zaehlt die 10ms Ueberlaufe

// Deklaration der Funktionen

main()
{
unsigned long timer2val;        // 32 bit Ergebniss
int maxcount;
                    // Maximale Anzahl der 10ms Durchlaufe fuer autoscale

init_lcd();
blank_lcd();
maxcount = 100;   // Beginne mit 100*10ms = 1Sekunde Zaehldauer

TMOD  =0x01;      // Timer0 in moduls 01: 16 zaehler
g10ms       =0;   // 10ms Zaehler  bei Null beginnen
TL0   =0xF6;      // 65536 - 9994 = 55542 vorsetzen = 0xD8F6
TH0   =0xD8;      // Naehere Beschreibung siehe Timer0 interrupt
TH2   =0;         // Timer 2 High byte loeschen
TL2   =0;         // Timer 2 Low  byte loeschen
TR0         =1;   // Timer0 starten
T2CON =0x02;      // Timer2 starten im Counter mode (von P1.7 getaktet )

TF2         =0;   // Timer2 Ueberlaufbit loeschen
ET0         =1;   // Timer0 int freigeben
EAL         =1;   // Globale Interrupt Freigabe
```

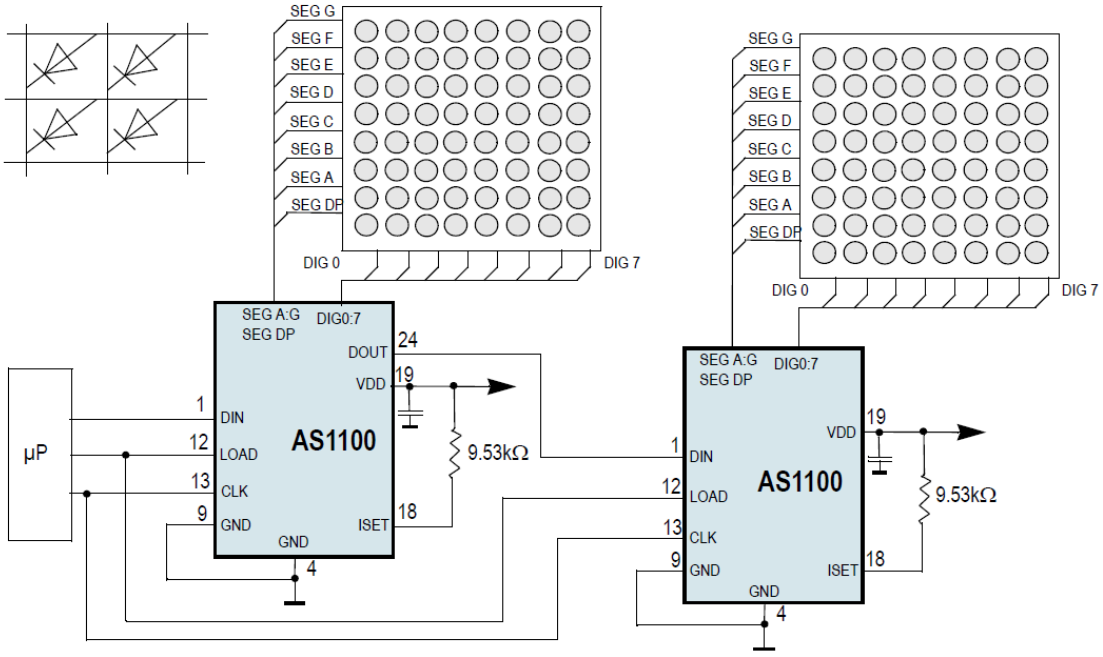
```
while(1)          // Endlosschleife
      {
      while( g10ms < maxcount )     { ; }
                              // Warte, geforderte Zaehldauer verstrichen
      T2CON = 0x00;                                   // Timer 2 stoppen
      TR0     = 0;            // Timer 0 stoppen
      timer2val = 256*TH2 + TL2;
                              // 16 bit Wert aus T2L und T2H zusammensetzen
      if(maxcount == 10)      // Wenn Zaehldauer 100ms war, Ergebniss * 10
            {
            timer2val = timer2val * 10;
            }
      if(TF2 == 1 )
      // Falls waehrend Zaehlung Ueberlauf auftrat Zaehldauer verkleinern
            {
            maxcount = 10;
            // Zaehldauer auf 10*10ms = 100ms verringern und neustarten
            TF2 = 0;          // Ueberlaufflag wieder loeschen
            }
      else if ( timer2val < 60000   )
                  // Falls Zaehlerstand zu klein war Zaehldauer erhoehen
            {
            sprintf(gTXT,"Messzeit=   %4dms",maxcount*10);
                                                // Ausgabe der Messzeit
            print_lcd(1,1,gTXT);                // Auf LCD ausgeben
            sprintf(gTXT,"Frequenz= %6luHz",timer2val);
                                                // In Textbuffer schreiben
            print_lcd(2,1,gTXT);                // Auf LCD ausgeben
            maxcount = 100;
                              // Zaehldauer auf 100*10ms = 100ms erhoehen
            }
      else
            {
            sprintf(gTXT,"Messzeit=   %4dms",maxcount*10);
                              // Ausgabe der Messzeit
            print_lcd(1,1,gTXT);
                                                // Auf LCD ausgeben
            sprintf(gTXT,"Frequenz= %6luHz",timer2val);
                                                // In Textbuffer schreiben
            print_lcd(2,1,gTXT);                // Auf LCD ausgeben
            }

      g10ms =0;                     // 10ms Zaehler  bei Null beginnen
      TL0   =0xF6;            // 65536 - 9994 = 55542 vorsetzen = 0xD8F6
      TH0   =0xD8;      // Naehere Beschreibung siehe Timer0 interrupt
      TH2   =0;         // Timer 2 High byte loeschen
      TL2   =0;         // Timer 2 Low  byte loeschen
      TR0   =1;         // Timer0 starten
      T2CON =0x02;
            // Timer2 starten im Counter mode ( von pin P1.7 getaktet )
      }
}
```

```
// Interrupt timer 0 wird alle 10ms ausgeloest:
// Timertakt = fclk/12 = 12MHz/12 = 1MHz (1us)
// Einsprung und Setzten des Nachladewertes dauert 6 cycles
// (LCALL,LJMP,POP ACC)
// d.h. Zaehler 10000 - 6 = 9994 Takte zaehlen lassen
// d.h. Zahler auf 65536 - 9994 = 55542 vorsetzen = 0xD8F6


void INTTIM0(void) interrupt 1
{
TL0 = 0xF6;
TH0 = 0xD8;                    // Zaehler jetzt nachgeladen
g10ms++;                       // 10ms Variable erhöhen
}
```

12 LED Graphikanzeige

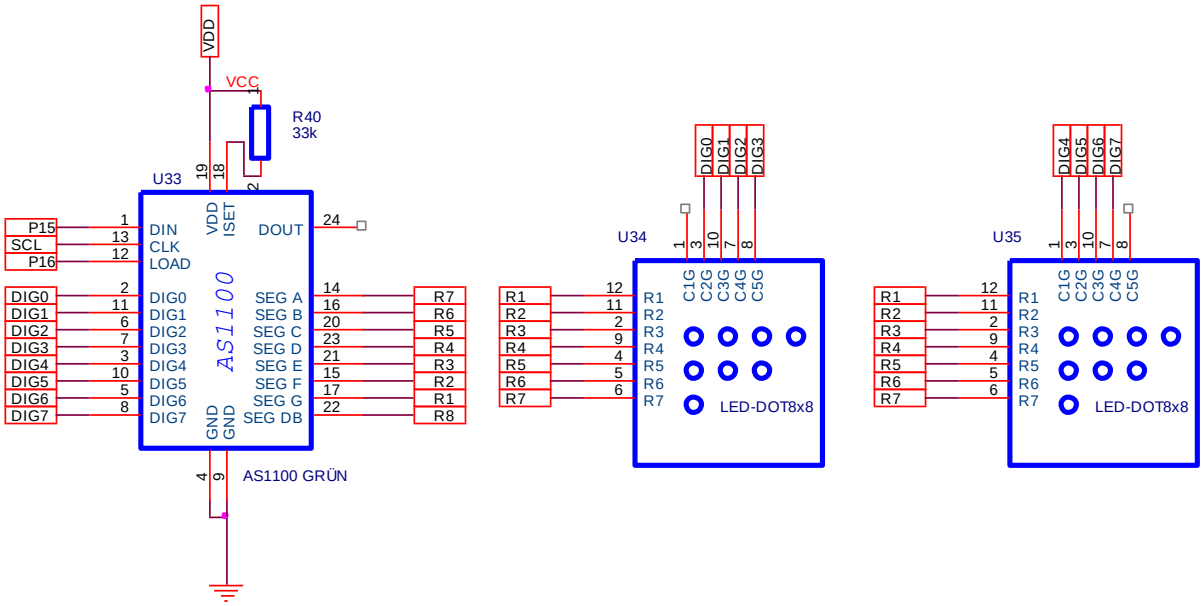
12.1 Blockschaltbild AS1100

(austriamicrosystems.com)



12.2 Schaltplan LED Graphikschaltung

LED 7x8 Graphikdisplay



12.3 Beispiel

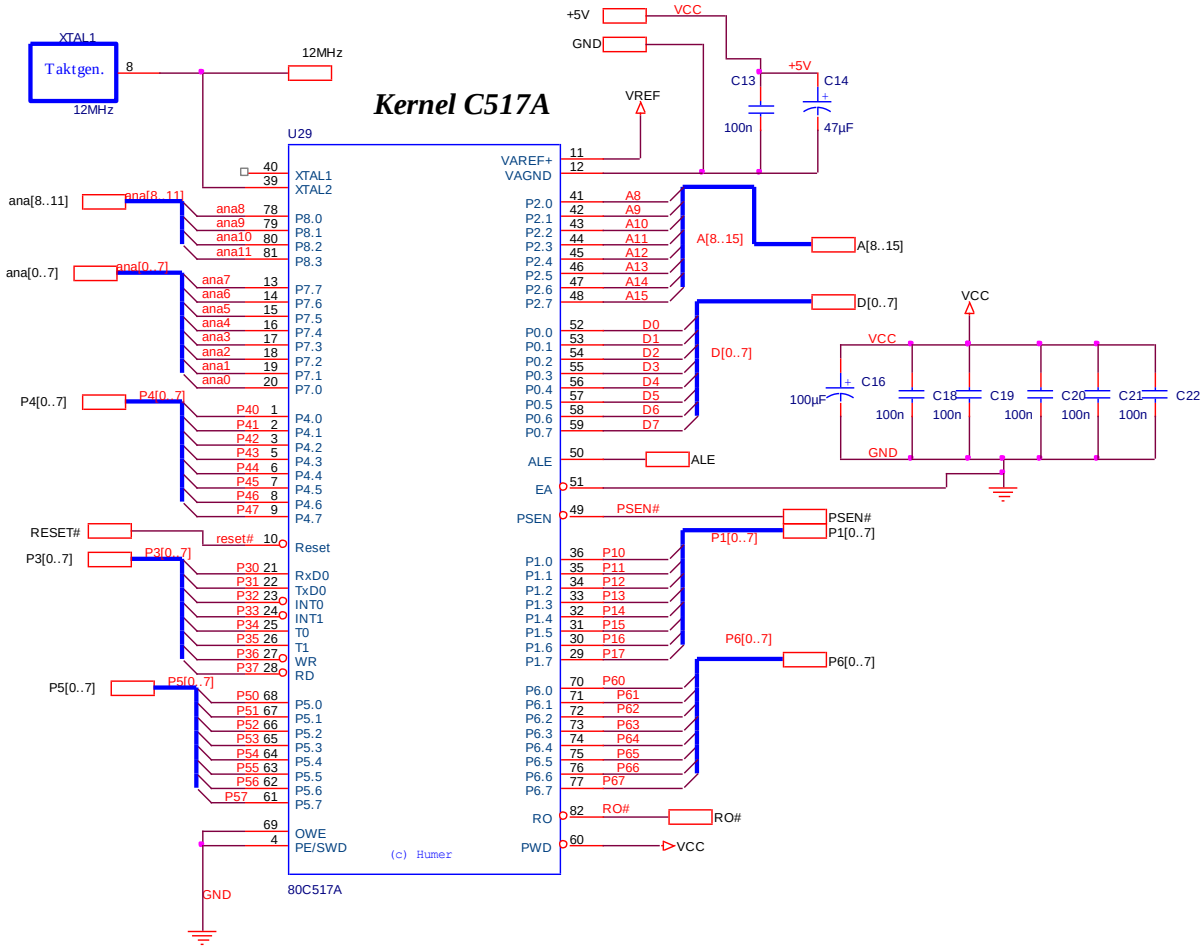
```
// ************************************************************
// PROGRAMMBEISPIEL: LED PUNKTMATRIX
// DATUM: 21.9.2005: MPA
//

#include<reg517a.h>
#include<stdio.h>
#include<lcd.h>

// Globale variablen (beginnen mit g...)
sbit gdin   =  P1^5;          // AS1100 Daten-Eingang
sbit gload  =  P1^6;          // AS1100 Load-Eingang
sbit gclk   =  P3^5;          // AS1100 Takt-Eingang

// LED-Matrix Grafik
//    LSB         1 1 1 1 1 1 1 1
//                1 0 0 0 0 0 0 1
//                1 0 0 0 0 0 0 1
//                1 0 0 0 0 0 0 1
//                1 0 0 0 0 0 0 1
//                1 0 0 0 0 0 0 1
//    MSB         1 1 1 1 1 1 1 1
// HEX:       7F 41 41 41 41 41 41 7F
char ggrafik1[8]={0x7F,0x41,0x41,0x41,0x41,0x41,0x41,0x7F};

// LED-Matrix Grafik
//    LSB         0 0 0 0 0 0 0 0
//                0 1 1 1 1 1 1 0
//                0 1 0 0 0 0 1 0
//                0 1 0 0 0 0 1 0
//                0 1 0 0 0 0 1 0
//                0 1 1 1 1 1 1 0
//    MSB         0 0 0 0 0 0 0 0
// HEX:       00 3e 22 22 22 22 3e 00
char ggrafik2[8]={0x00,0x3e,0x22,0x22,0x22,0x22,0x3e,0x00};

// LED-Matrix Grafik
//    LSB         0 0 0 0 0 0 0 0
//                0 0 0 0 0 0 0 0
//                0 0 1 1 1 1 0 0
//                0 0 1 0 0 1 0 0
//                0 0 1 1 1 1 0 0
//                0 0 0 0 0 0 0 0
//    MSB         0 0 0 0 0 0 0 0
// HEX:       00 00 1c 14 14 1c 00 00
char ggrafik3[8]={0x00,0x00,0x1c,0x14,0x14,0x1c,0x00,0x00};

// LED-Matrix Grafik
//    LSB         0 0 0 0 0 0 0 0
//                0 0 0 0 0 0 0 0
//                0 0 0 0 0 0 0 0
//                0 0 0 1 1 0 0 0
//                0 0 0 0 0 0 0 0
//                0 0 0 0 0 0 0 0
//    MSB         0 0 0 0 0 0 0 0
// HEX:       00 00 00 08 08 00 00 00
char ggrafik4[8]={0x00,0x00,0x00,0x08,0x08,0x00,0x00,0x00};
```

```
// Deklaration der Funktionen
void AS1100send(char , char);                    // AS1100 Daten senden
void AS1100init(void);                           // AS1100 inistialisieren
void AS1100display(char *);                      // AS1100 stellt Matrix dar
main()
{
gclk  = 0;                                  // Inititialisieren der globalen
Variablen
gdin  = 0;                                  // Inititialisieren der globalen
Variablen
gload = 0;                                  // Inititialisieren der globalen
Variablen
AS1100init();                               // AS1100 initialisieren
while(1)                                    // Endlosschleife
      {
      AS1100display(ggrafik4);        // Matrix darstellen
      warte(400);                           // Warten
      AS1100display(ggrafik3);
      warte(400);
      AS1100display(ggrafik2);
      warte(400);
      AS1100display(ggrafik1);
      warte(400);
      AS1100display(ggrafik2);
      warte(400);
      AS1100display(ggrafik3);
      warte(400);
      }
}
// Funktionen
// Daten an AS1100 senden
void AS1100send(char addr, char daten)
{
unsigned char mask;                         // Maske zum extrahieren eines
Bits
                                                 // Muss unsigned sein, sonst
wird kein logischer shift gemacht
mask = 0x80;                                // Maske beginnt mit 10000000
while(mask)                                 // Adressbyte schicken MSB zuerst
      {
      gdin = (addr & mask);           // Bit extrahieren und Ausgang setzten
      gclk = 1;                             // positive Taktflanke erzeugen
      gclk = 0;                             // negative Taktflanke erzeugen

      mask = mask >> 1;               // Maske: bit um 1 Stelle nach rechts
schieben
      }
mask = 0x80;                                // Maske beginnt mit 10000000
while(mask)                                 // Datenbyte schicken MSB zuerst
      {
      gdin = (daten & mask);          // Bit extrahieren und Ausgang setzten
      gclk = 1;                             // positive Taktflanke erzeugen
      gclk = 0;                             // negative Taktflanke erzeugen

      mask = mask >> 1;               // Maske: bit um 1 Stelle nach rechts
schieben
      }
gload = 1;                                  // Ladeimpuse erzeugen
gload = 0;
}
```

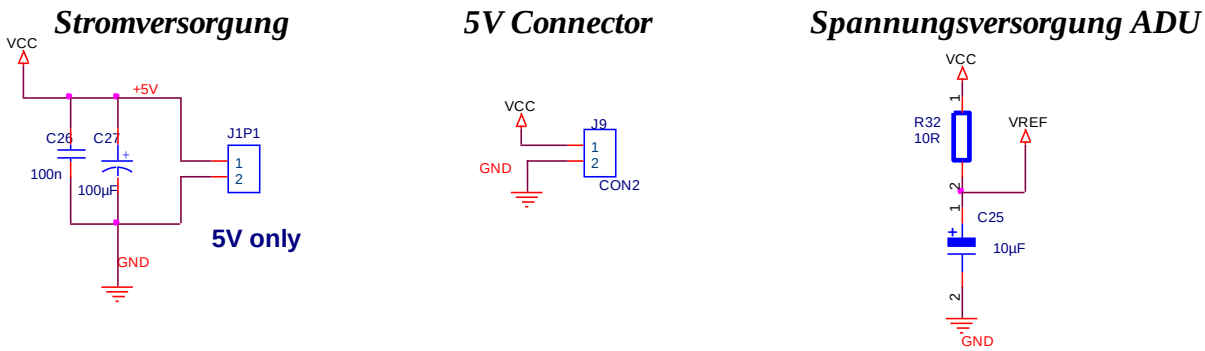
```
// AS1100 INITIALISIEREN
void AS1100init(void)
{
gload = 1;                                  // Ladeimpuse erzeugen um
Schieberegister auf Anfang zu setzen
gload = 0;


AS1100send(0x0E,0x02);              // Reset senden
AS1100send(0x0E,0x00);              // Reset verlassen
AS1100send(0x0C,0x01);              // Shutdown mode verlassen
AS1100send(0x0B,0x07);              // Scan limit setzen
AS1100send(0x09,0x00);              // Keine BCD-Dekodiereung
AS1100send(0x0A,0x0F);              // Intensity max
AS1100send(0x0F,0x00);              // Display test OFF
}

// Grafikarray auf AS1100 ausgeben
void AS1100display(char *array)
{
char i;                                          // Laufvariable
for(i=0;i<8;i++)                    // Spalten 0-7
      {
      AS1100send(i+1,array[i]);     // Ausgabe: Spalte 1-8 , array[0-7]
      }
}
```

13 Schaltpläne

13.1 Kern

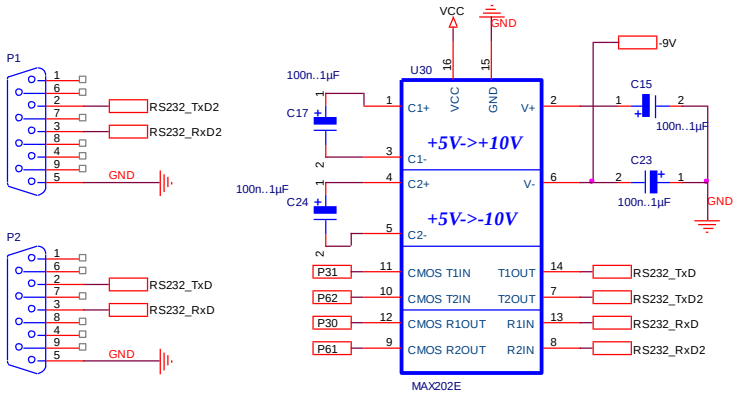


13.2 Stromversorgung



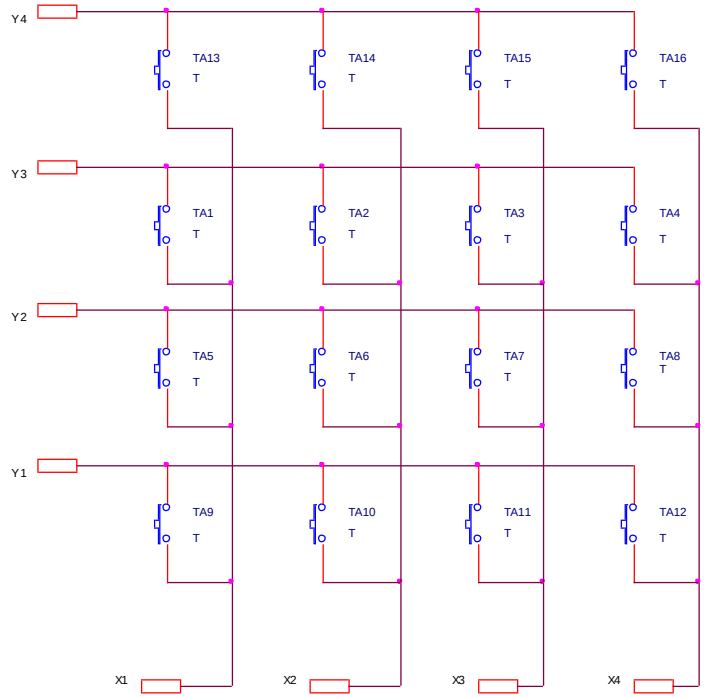
13.3 Serielle Schnittstelle

RS232 Einheit

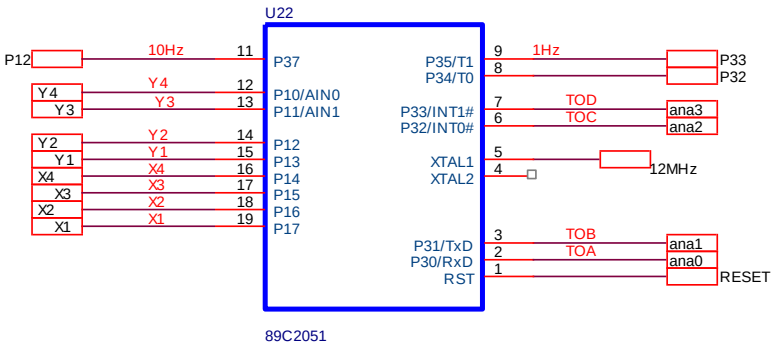


13.4 Tastaturbeschaltung

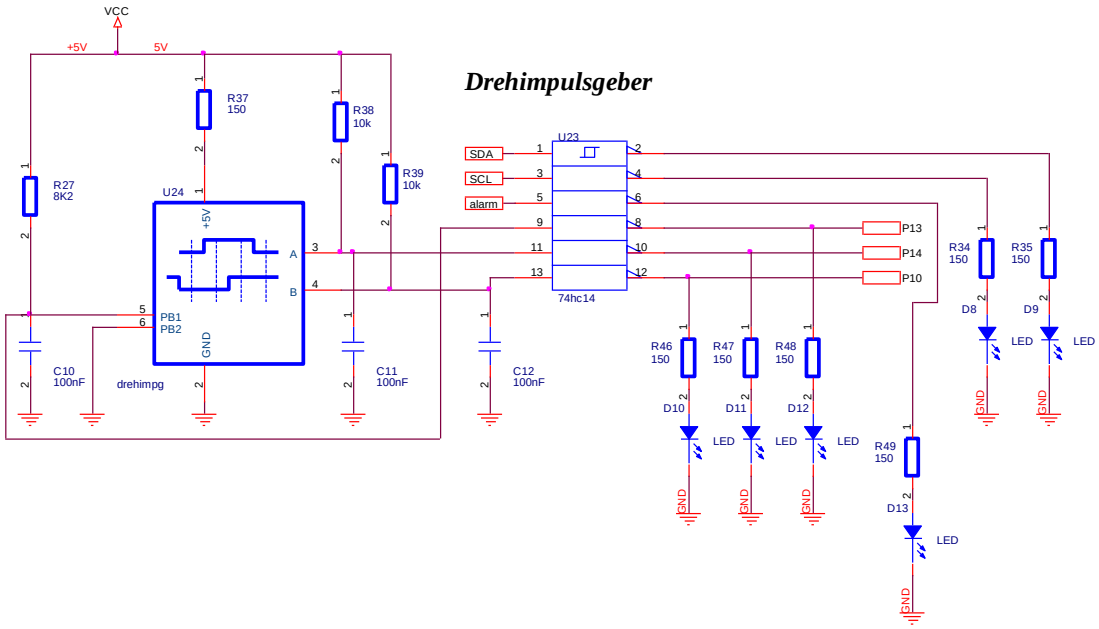
Tastaturmatrix



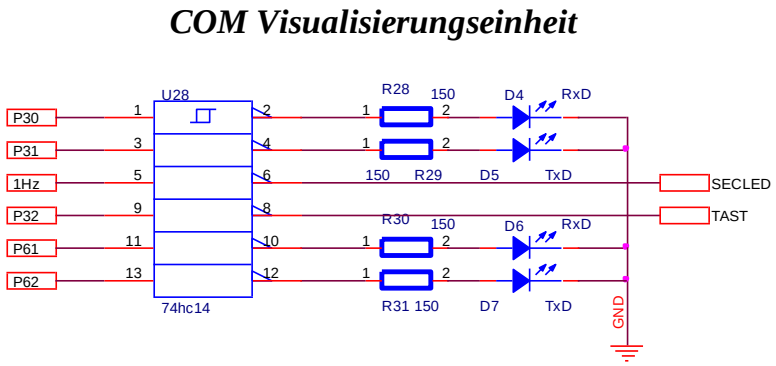
Tastaturdecoder



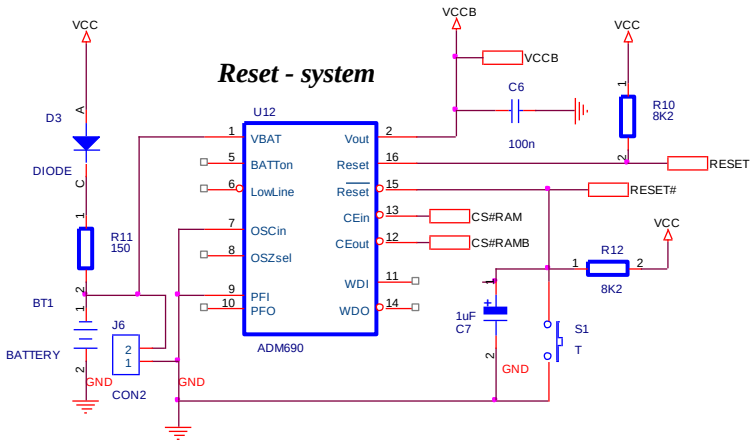
13.5 Drehimpulsgeber



13.6 RS232 Visualisierung

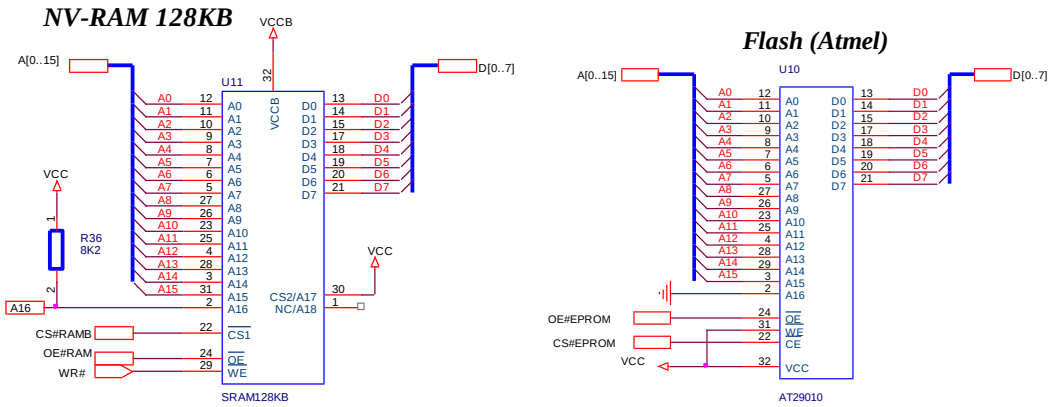


13.7 Reset-System

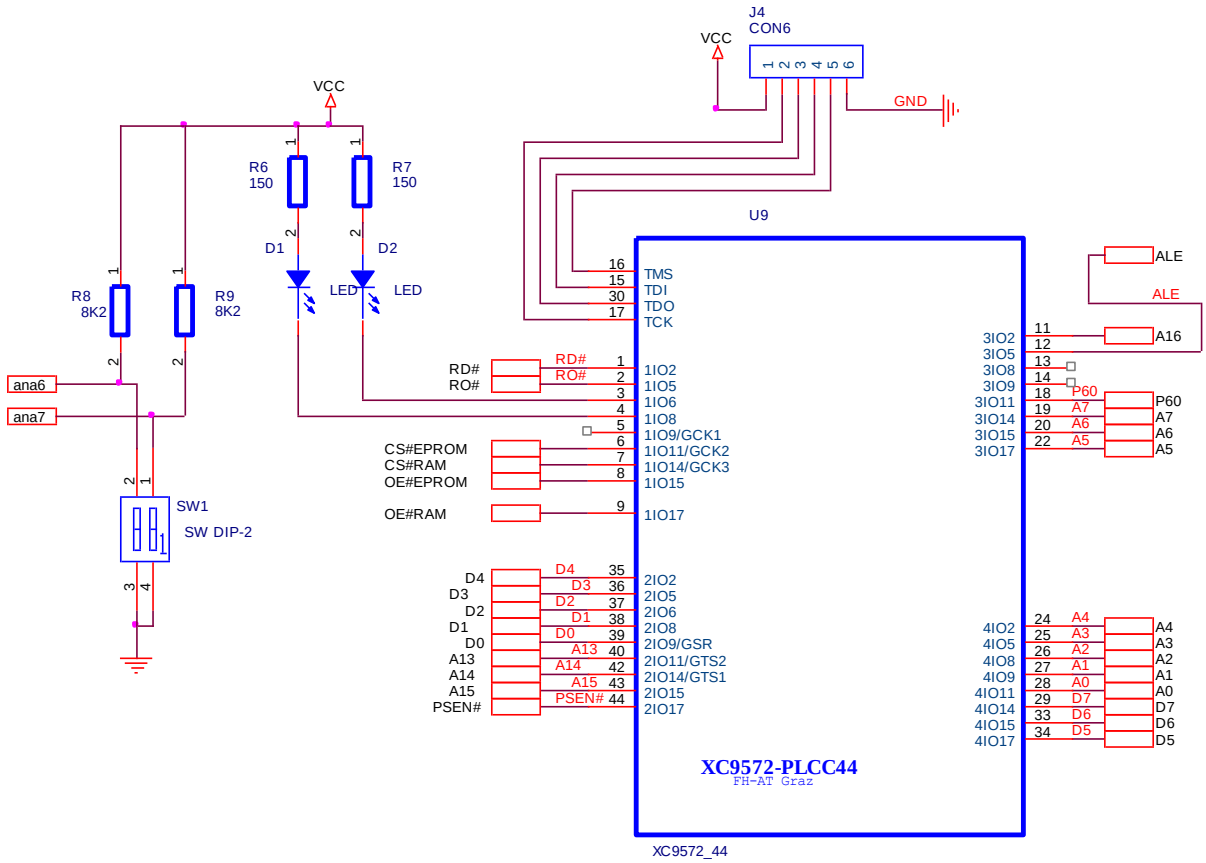


Lötjumper, on = no ACCU
D3 dann nicht bestückt

13.8 RAM und FLASH

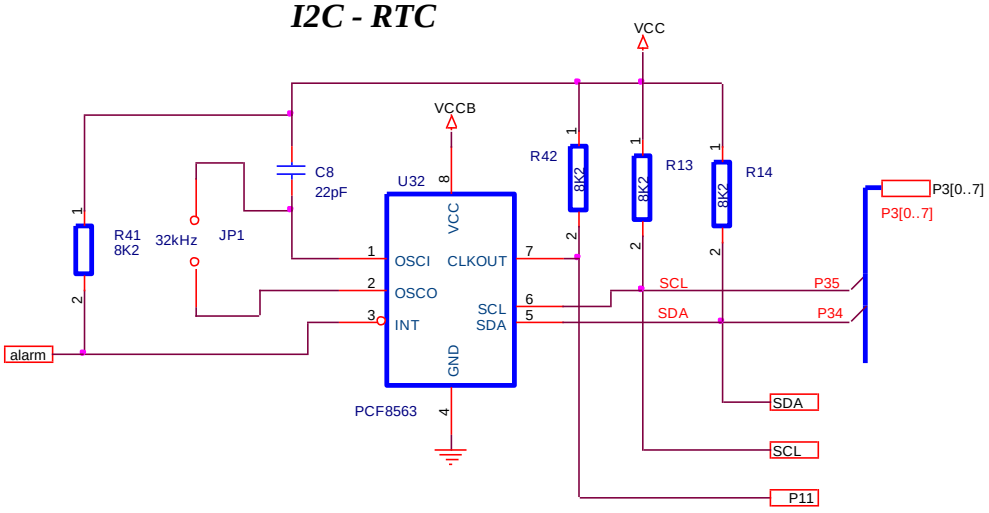


13.9 CPLD Baustein



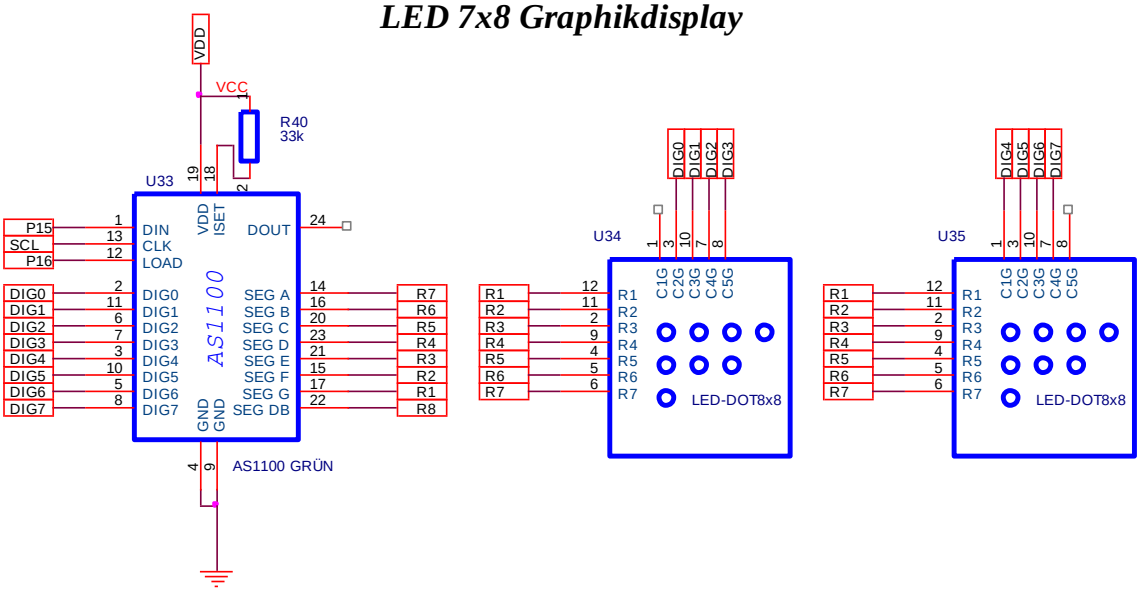
13.10 I2C – RTC

I2C - RTC

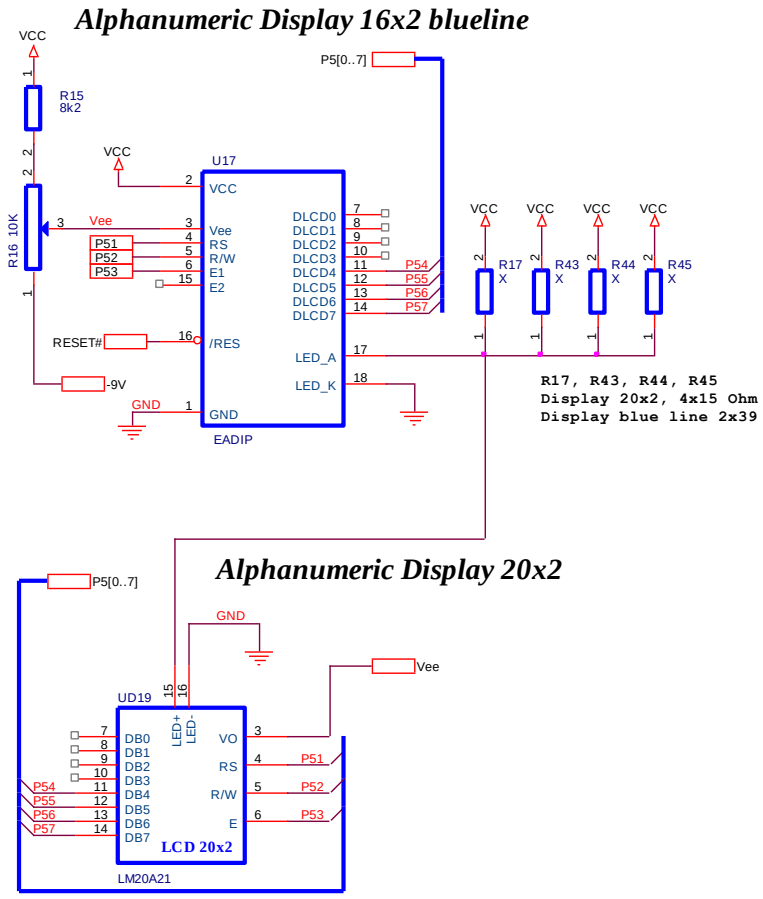


13.11 LED Grafikdisplay

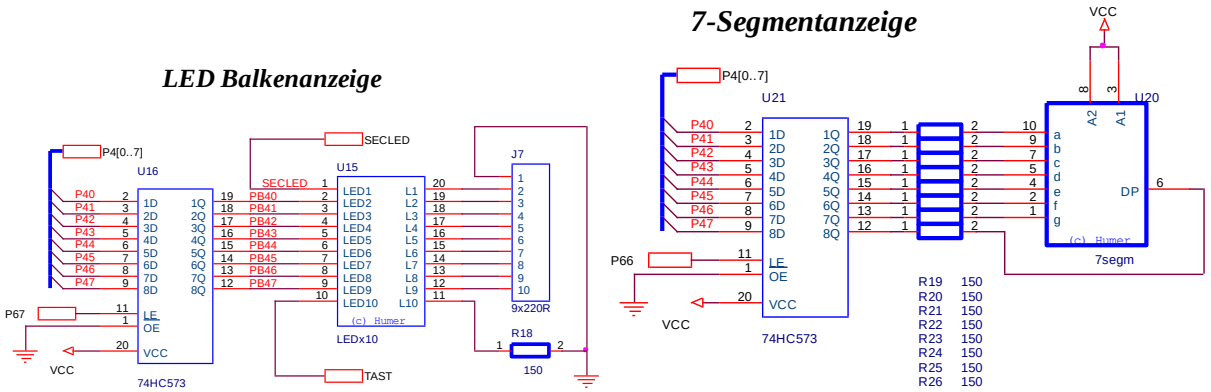
LED 7x8 Graphikdisplay



13.12 LC-Anzeige

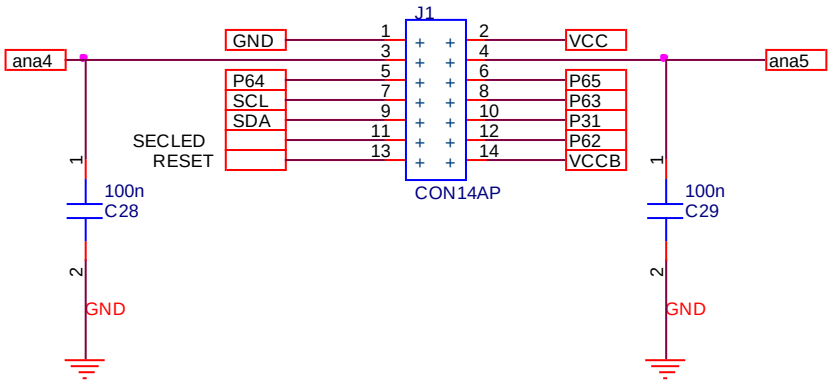


13.13 LED Balkenanzeige und 7Segmentanzeige



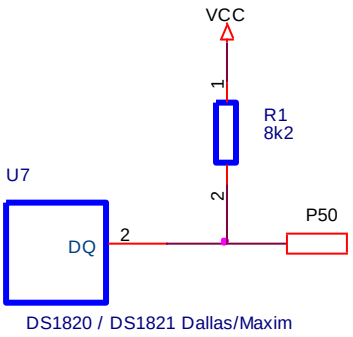
13.14 Erweiterungsstecker

Boarderweiterung



13.15 Temperatursensor 1-wire

Temperatursensor (1bit-Bus)



13.16 CPU-Belegung tabellarisch

P6.6 + P7.7 7-Segm und LED-B
P5.1 bis P5.7 LCD
P7.3 Tastaturdecoder - D
P7.2 Tastaturdecoder - C
P7.1 Tastaturdecoder - B
P7.0 Tastaturdecoder - A
P7.6 Anwenderprogramm
P7.0 System intern
P1.0 INT3 Drehimpg. B
P1.1 INT4 RTC
P1.2 INT5 10Hz
P1.3 INT6 (Enter) pos. Flanke
P1.4 INT2 Drehimpg. A
P1.5 DIN AS1100
P1.6 LOAD AS1100
P1.7 T2 Licht Frequenz Umsetzer
P3.0+1 Serielle Schnittstelle
P3.2 INT0 TASTATUR
P3.3 INT1 SEKUNDENTAKT
P3.4 T0 P34 SDA
P3.5 T1 P35 SCL
P5.0 1 bit - Bus Temperatursensor
POTI R4 Kanal 10
POTI R3 Kanal 9
NTC R8 Kanal 8
www.humerboard.at