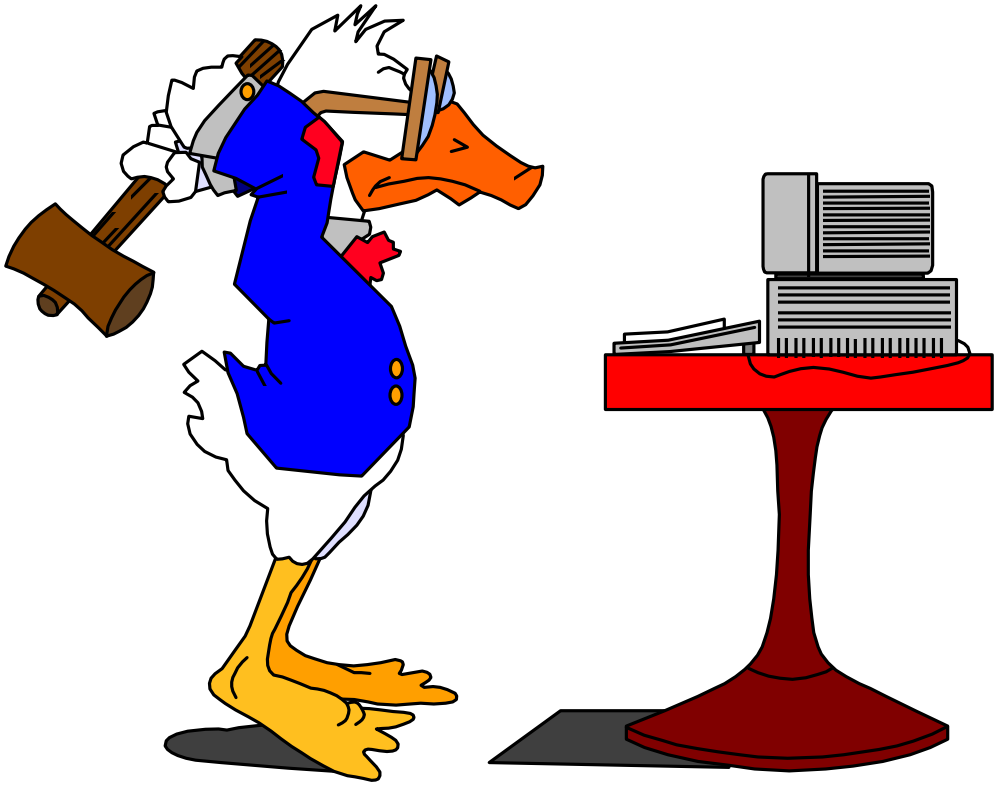

MODUL 7

I²C-Bussystem

Version 1.0 Maierhofer 1998

Inhaltsverzeichnis
Modul 7
I²C-Bussystem

Inhalt **Seite**

1 Allgemeines über den I²C-Bus:

I²C ist die Abkürzung für IIC (Intern IC-Bus). Es ist ein serieller Bus, jedoch kein Feldbus und er wird vor allem zur Verbindung von Mikroprozessoren und anderen IC´s auf Platinen verwendet.
Der Vorteil dieses Buses ist, daß er nur 2 Leitungen benötigt (SDA - Serial DAta, SCL - Serial CLock). Parallel Buse brauchen sehr viel mehr Leitungen und deshalb müssen Geräte mit parallelen Busen größer gebaut werden, da die vielen Leitungen mehr Platinenfläche benötigen und auch die Bauform der anderen IC´s größer wird. Der I²C-Bus ist ein relativ langsamer Bus. Aber da die Datenmengen bei Konsumgeräten meist gering sind, reicht die Übertragungsrate des I²C-Buses aus (nur 10kByte/sec). Sollen mehr Daten übertragen werden, muß ein anderer (paralleler) Bus verwendet werden.

Jeder Teilnehmer des Buses besitzt eine eigene Adresse. Es ist dabei egal ob es sich um ein LCD-Display oder um einen Speicherbaustein handelt. Jeder Teilnehmer kann nun Sende- und Empfangsfunktionen ausüben. Es aber denkbar, daß z.B. ein Tastaturinterface nur Daten senden wird. Ein LCD-Display wird wahrscheinlich nur Daten empfangen. Wichtig ist aber die Unterscheidung zwischen Master und Slave. Als Master bezeichnet man einen Baustein der den Datentransfer, dessen Richtung und auch die Impulse auf der Clock-Leitung (SCL) generiert. Ein Gerät, daß vom Master angesprochen wird und an einem Transfer teilnimmt bezeichnet man als Slave. Die meisten I²C kompatiblen Bausteine beginnen nicht mit einem Transfer sonder nehmen den Slave-Modus ein. Natürlich darf es am Bus mehrere Master geben, aber nur einer darf aktiv sein. Allgemein müssen Geräte, egal ob Master oder Slave, Open-Drain- oder Open-Collector Ausgänge besitzen.

2 Start und Stopbedingungen:

Um vom Master aus einen Slave anzusprechen und einen Transfer zu initiieren, muß eine Startbedingung erzeugt werden. Dies kann aber nur geschehen wenn der Bus frei ist. Erkennen kann das ein Master daran, daß die Leitungen SCL und SDA HIGH sind. Zur Beendigung eines Transfers muß dann wieder eine Stopbedingung erzeugt werden. Ein Vorteil beim I²C Bus liegt darin, daß er für diese Bedingungen auch die Leitungen SDA und SCL benutzt. Durch spezielle Kombinationen von Pegel und Flanken können diese Funktionen realisiert werden (siehe Diagramm 1). Es ist also nicht notwendig eigene Leitung für diese Funktionen zu legen.

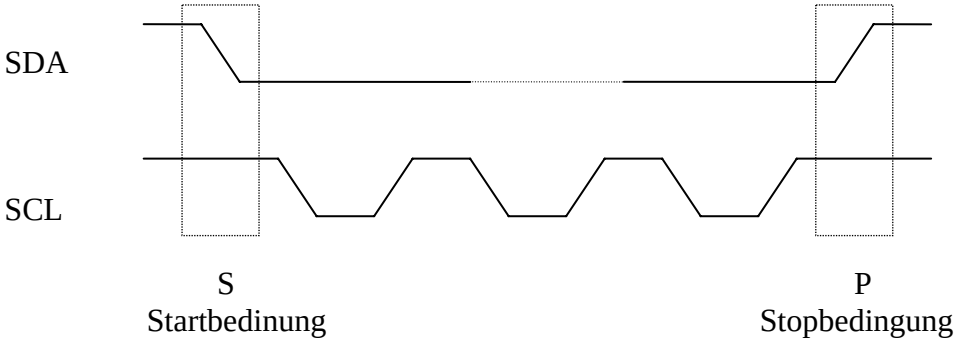

Diagramm 1

Wie im Diagramm oben beschrieben kann eine Startbedingung dadurch erzeugt werden, daß bei dem Zustand SCL=HIGH die Leitung SDA von HIGH auf LOW geht. Bei der Stopbedingung geschieht dies genau umgekehrt, d.h. bei SCL=HIGH geht SDA von LOW auf HIGH.

Eine Besonderheit ist, daß zur Beendigung eines Transfers nicht unbedingt eine Stopbedingung verwendet werden muß, sondern auch eine sogenannte „Repeated start condition“ eingesetzt werden kann, d.h. der Master erzeugt einfach eine neue Startbedingung. Der Vorteil liegt darin, daß bei einem etwaigen nachfolgenden Transfer durch wegfallen der Stopbedingung Zeit gespart werden kann.

3 Acknowledge-Bit:

Beim Acknowledge-Bit wird während der HIGH-Zeit des 9. Clockimpulses, d.h. erster Impuls nach den Daten, der Zustand auf der SDA-Leitung überprüft. Ist der Zustand LOW so ist das eine Bestätigung dafür, daß der Slave die Daten erhalten hat. Ist der Zustand jedoch HIGH wird das als Not Acknowledge interpretiert. Ist dies der Fall heißt das, daß zum Beispiel ein angesprochener Slave an der Übertragung nicht mehr teilnimmt.

4 Arten der Datenübertragung am I²C-Bus:

Um am I²C-Bus Daten zu übertragen, muß dafür gesorgt werden, daß während der HIGH-Zeit der SCL Leitung sich der Zustand auf der SDA Leitung nicht ändert (siehe Diagramm 2). Ansonst könnte daß Signal, wie oben besprochen, als eine Start- oder Stopbedingung verstanden werden.

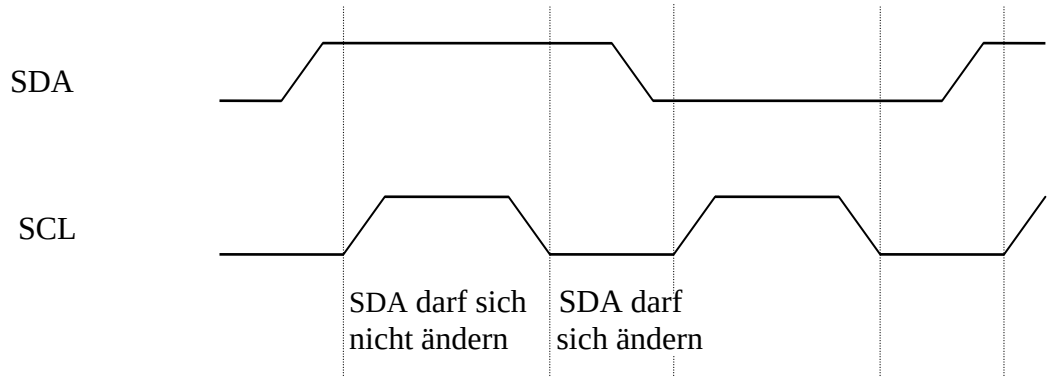

Diagramm 2

4.1 Datenübertragung bei 7-bit Adressierung:

Wie schon weiter vorne erwähnt muß bei der Datenübertragung die Adresse des Slaves gesendet werden. Dies geschieht direkt nach der Startbedingung, d.h. es wird als erstes Byte das Adressbyte gesendet, welches an den ersten sieben Stellen (ersten 7 Bit => 7-bit Adressierung) die Adresse und an der letzten Stelle (letztes Bit; LSB-Last significant Bit) die Richtung (lesen oder schreiben; R/W) angibt. Die Bits müssen einzeln

hintereinander gesendet werden, d.h. es müssen 8 Clock-Impulse auf der SCL Leitung erzeugt werden (dies gilt auch für normale Daten). Dadurch wird mit jedem Impuls ein Bit übertragen. Die Bausteine die durch die Startbedingung zur Überwachung des Buses aufgerufen wurden, vergleichen die gesendete Adresse mit ihrer eigenen. Stimmet die eigene Adresse mit der Gesendeten nicht überein so verabschiedet sich der Baustein vom Bus.

Bsp.: Adressbyte 1 0 1 0 0 0 0 1

Adresse „1“ für lesen; „0“ für schreiben (R/W)

Danach erzeugt der angesprochene Slave ein Acknowledge-Bit. Wird ein Not Acknowledge empfangen ist entweder der angesprochene Baustein beschädigt, oder er ist am Bus gar nicht vorhanden. Danach können solange beliebig viele Bytes übertragen werden, bis entweder eine Stopbedingung oder eine Repeated Startcondition auftritt.

4.1.1 Master sendet Daten an den Slave:

Der Transfer erfolgt wie unter 4.1 beschrieben. Es ist nur wichtig, daß das 8-Bit der Adresse „0“ , also auf schreiben gesetzt ist. Nach jedem Byte erzeugt der Slave ein Acknowledge (siehe Diagramm 3).

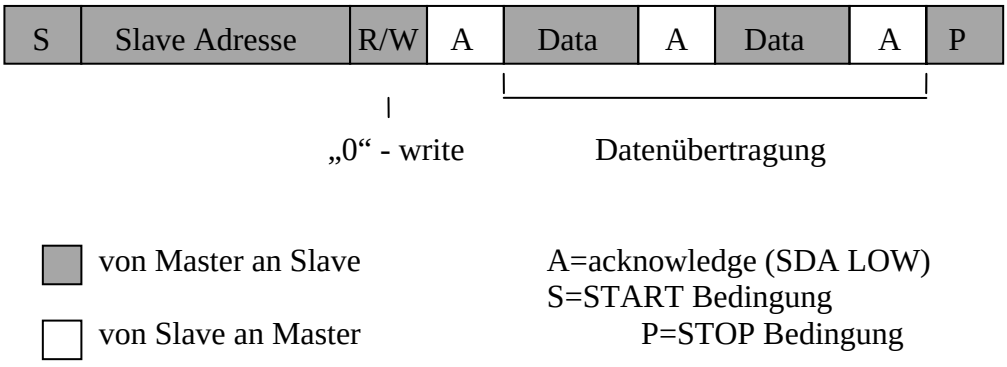

Diagramm 3

4.1.2 Master liest Daten vom Slave:

Hier wird wie unter 4.1 gezeigt das Adressbyte gesendet, aber nur mit „1“ als Richtungsbit (lesen). Danach wechselt der Slave in den Sendemodus und überträgt die Daten. Der Master hat nur mehr die Aufgabe, nach jedem empfangenem Byte mit einem Acknowledge zu bestätigen (siehe Diagramm 4).

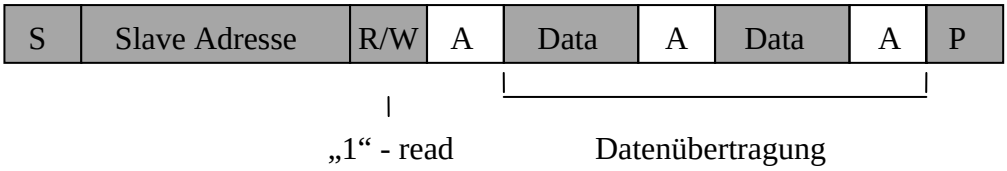

Diagramm 4

4.1.3 Kombiniertes Format:

Es wird wie unter 4.1 beschrieben zuerst die Adresse und dann die Daten gesendet. Das Richtungsbit ist je nach Wunsch „0“ oder „1“ (schreiben oder lesen). Um jetzt die Richtung des Transfers zu ändern, muß eine „Repeated startcondition“ erzeugt werden. Danach wird wieder die Adresse aber mit entgegengesetztem Richtungsbit gesendet (siehe Diagramm 5).

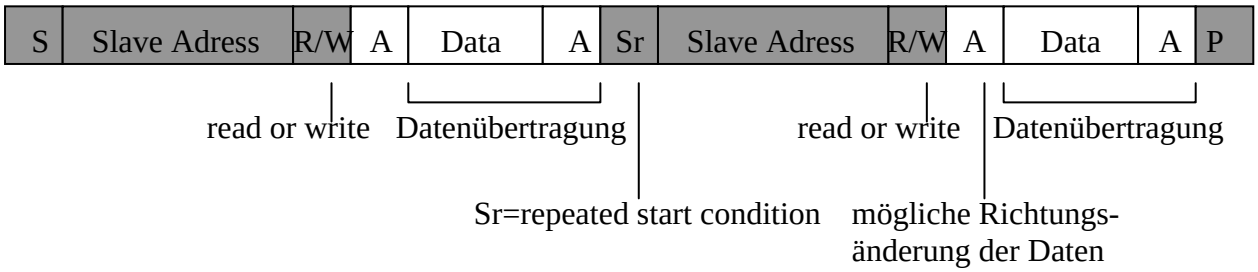

Diagramm 5

4.2 Datenübertragung bei 10-bit Adressierung:

Da die meisten Adressen bei der 7-bit Adressierung schon belegt sind, wurde von Philips ein 10-bit Adressmodus entwickelt. Der Grundgedanke dabei ist, zwei Adressbytes anstatt nur einem zu senden. Das Richtungsbit (lesen oder schreiben) ist bei dem ersten Adressbyte immer „0“. Der Slave quittiert nun mit einem Acknowledge und der Master sendet das zweite Adressbyte. Der Slave muß wieder mit einem Acknowledge bestätigen. Danach kann ein Datentransfer stattfinden.

4.2.1 Master sendet Daten an den Slave:

Der Transfer erfolgt wie unter 4.2 beschrieben, d.h. zuerst das erste Adressbyte mit dem Richtungsbit „0“, und dann daß zweite Adressbyte. Danach können beliebig viele Daten bis zu einer Stopbedingung übertragen werden (siehe Diagramm 6).

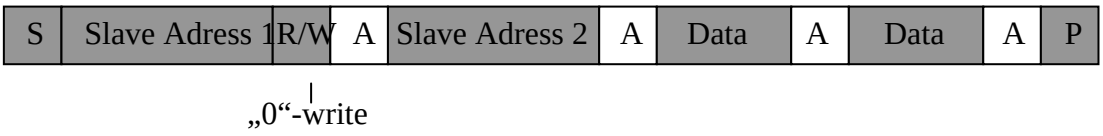

Diagramm 6

4.2.2 Master liest Daten vom Slave:

Es wird hier zuerst das erste Adressbyte mit dem Richtungsbit „0“ gesetzt. Danach wird das zweite Adressbyte wie unter 4.2.1 gesendet. Um jetzt den Slave auslesen zu können muß noch einmal das erste Adressbyte gesendet werden. Es ist aber zu beachten, daß das Richtungsbit jetzt „1“ (lesen) sein muß. Der Slave geht jetzt in den Sendemodus über und der Master generiert von nun an solange alle Acknowledge-bits bis der Transfer durch eine Stopbedingung oder eine Repeated startcondition beendet wird (siehe Diagramm 7).

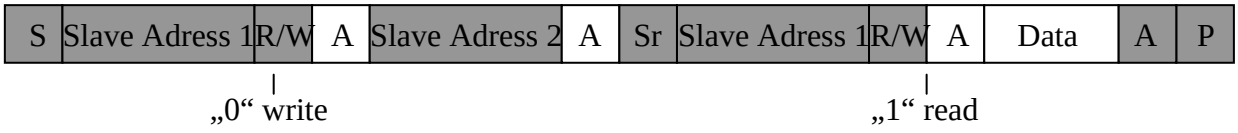

Diagramm 7

4.2.3 Kombiniertes Format: Wechsel von Schreiben auf Lesen:

Es wird zuerst das erste Adressbyte mit dem Richtungsbit „0“ gesendet und dann das zweite Adressbyte wie unter 4.1 übertragen. Jetzt können Daten übertragen werden. Um jetzt von Schreiben auf Lesen überzugehen, muß eine Repeated startcondition erzeugt werden. Danach wird wieder das erste Adressbyte mit dem Richtungsbit für lesen gesendet. Jetzt können beliebig viele Daten bis zu einer Stopbedingung übertragen werden.

4.2.4 Kombiniertes Format: Wechsel von Lesen auf Schreiben:

Wie unter 4.2.2 wird zuerst das erste Adressbyte mit dem Richtungsbit für lesen übertragen, und danach das zweite Adressbyte gesendet. Danach können beliebig viele Daten übertragen werden. Will man nun von lesen auf schreiben übergehen muß eine Repeated startcondition erzeugt werden. Danach müssen aber beide Adressbytes mit dem Richtungsbit „0“, wie unter 4.2.1, gesendet werden.

5 Verwendung des Bausteines PCF8583:

Der Baustein PCF8583 ist mit einer I²C-Echtzeituhr, einem Kalender und einem RAM (256*8) ausgestattet. Es ist ein 10-bit adressierbarer Baustein dessen Registeradressen von 00 bis 0F Uhr, Kalender und anderen Funktionen und ab 0F bis FF dem RAM zugeordnet sind (s. Diagramm 8).

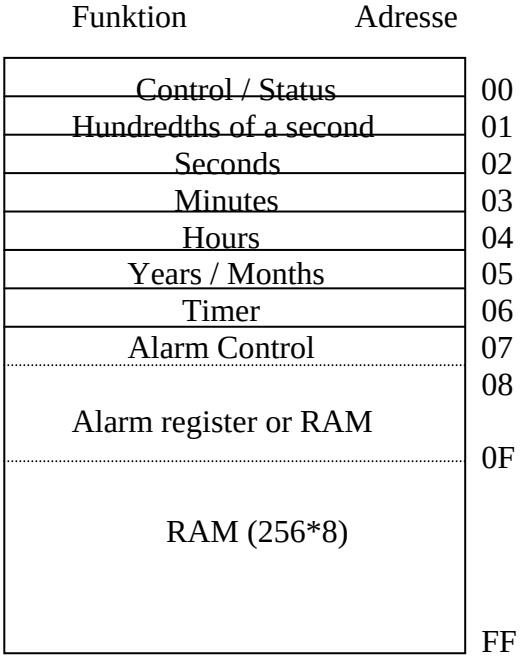

Diagramm 8

Um also den PCF8584 anzusprechen, muß eine 10-bit Adressierung (wie unter 4.2 beschrieben) verwendet werden. Das folgende Beispielprogramm liest die Uhrzeit des Bausteins PCF8584 aus.

Beispiel 1:

```
/*****************************************************************/
/*      Maierhofer Markus       5AIH    1997/98                  */
/*****************************************************************/

#pragma iv(0x8000)
#include <stdio.h>
#include <reg517a.h>
#include "lcd.h"

#define WT 2            /* Wait-Time (min 4,3 æsec) - see I²C
```

```
timeing protocol */

char serg,merg,herg,buffer[21];
sbit SCL = P3^5;
sbit SDA = P3^4;
Bit flag;
```

In diesem Teil werden die Headerdatein eingebunden und die Variablen definiert.In dem folgenden Unterprogramm wird der Interrupt 2 zur Generierung eines Sekundentaktes verwendet. Das daruaffolgende Unterprogramm dient nur zur Erzeugung einer Wartezeit.

```
void sectakt() interrupt 2
        {flag=1;
        }

void wait (int msec)
        {char i;
         for (msec; msec!=0; msec--)
                for (i=0; i<83; i++);     /* 12 µs/durchlauf */
                                          /* 12*83 = ca.1 ms */
        }
```

Die folgenden beiden Unterprogramme dienen zur Erzeugung einer Start- und Stopbedingung. Es wird dabei die Leitung SCL auf 1 gesetzt und danach die Leitung SDA auf 0. Es wird also während er Highzeit von SCL die SDA Leitung verändert. Die Stopbedingung funktioniert im Prinzip gleich, nur wird hier SDA von 0 auf 1 verändert.

```
void i2c_start()
        {SCL = 1;
         SDA = 0;    /* Serial Data Line I²C Bus */
         wait(WT);   /* Wait min. 4,7 æsec - see I²C timing   protocol
      PCF8583 */
         SCL = 0;    /* Serial Clock Line I²C Bus */
         wait(WT);
        }

void i2c_stop()
        {SDA = 0;    /* Serial Data Line I²C Bus */
         wait(WT);
         SCL = 1;    /* Serial Clock Line I²C Bus */
         wait(WT);
         SDA = 1;
         wait(WT);
        }
```

Um jetzt Daten schreiben oder lesen zu können werden die folgenden beiden Unterprogramme benötigt. Das erste hat die Funktion des Schreibens. Die Daten können nur seriell, d.h. jedes Bit einzeln übertragen werden. Es wird daher das Zeichen (z.B.: 14h => 00010010) mit einer

Bitmaske verglichen. Die Bitmaske beginnt mit 10000000 und wird mit jedem Durchlauf der FOR-Schleife nach rechts verschoben (10000000 => 01000000). Bei jedem Durchlauf wird die Maske mit dem Zeichen binär verglichen, d.h. das nur jenes Bit des Zeichens übertragen wird das mit der Bitmaske bei einer logischen UND-Verknüpfung eine 1 ergibt

(Zeichen: 00010010
Bitmaske: 10000000
Übertragen wird: 0 (weil 0 und 1 logisch 0 ergibt)

Nächster Durchlauf der FOR-Schleife:
Zeichen: 00010010
Bitmaske: 01000000
Übertragen wird: 0 (weil 0 und 1 logisch 0 ergibt)
u.s.w.)

```
void i2c_trans(unsigned char zeich)
        {unsigned char i;
         unsigned char mask = 0x80; /* Bitmaske */
         printf(" ->Transmit: ");
         for (i=0;i<8;i++)
               {SDA = ((zeich & mask) == mask) ? 1:0;
                     /*Zeichen u. Bitmaske werden binär verglichen.
                     Bei TRUE wird 1, bei FALSE wird 0 auf die
                     Leitung SDA gelegt.*/

                printf("%1d|",(int)SDA);
                mask >>= 1; /*Bitmaske wird nach rechts verschoben*/
                wait(WT);
                SCL = 1;
                wait(WT);  /*Daten können während SCL=1 gelesen
                           werden*/
                SCL = 0;
               }
         SDA = 1;
                     /*SDA wird „von Hand" auf 1 gesetzt. Bei einem
                     Acknowledge muß es von selbst sofort wieder auf
                     0 gehen (Abfrage darunter).*/
         SCL = 1;
         if (SDA==0) printf(" Acknowledge received\n");
         if (SDA==1) printf(" Acknowledge not received\n");
         wait(WT);
         SCL = 0;
         wait(WT);
        }
```

Das Lesen von Daten funktioniert im Prinzip gleich wie das Schreiben, nur müssen hier die 8 Datenbits die über die Leitung SDA seriell gesendet werden, wieder zu einem Wort zusammengebaut werden. Wenn SDA=1 ist, dann wird die Bitmaske mit dem bereits vorhanden Zeichen (standard=00000000) ODER verknüpft.
(d.h. Bitmaske: 1000000
Zeichen: 0000000

Endgült. Zeichen: 1000000

Bitmaske: 0100000
Zeichen: 1000000
Endgült. Zeichen: 1100000

u.s.w)

Ist SDA=0 wird das Zeichen einfach so gelassen wie es ist und nur die Bitmaske nach rechts verschoben.

```
unsigned char i2c_rec()
        {unsigned char i;
         unsigned char mask = 0x80;   /* Bitmaske */
         unsigned char zeich = 0;
         for (i=0;i<8;i++)
               {SCL = 1;
               wait(WT);
                zeich = (int)SDA ? (mask | zeich) : zeich;

                     /*Ist SDA=1 dann wird die Bitmaske mit dem
                     Zeichen binär ODER verknüpft. Bei SDA=0 bleibt
                     zeich so wie es vorher war.

               mask >>= 1;
                SCL = 0;
                wait(WT);
               }
         return(zeich);
        }
```

Im Hauptprogramm werden nur mehr die zuvor geschriebenen Unterprogramme aufgerufen. Es wird hier nur die Uhrzeit aus dem Baustein PCF8583 jede Sekunde (mit Hilfe des Interrupt 2 der jede Sekunde einmal ausgelöst wird) ausgelesen. Die Slaveadresse des Bausteins ist 0xA0, die Adressen für Stunde, Minute und Sekunde, die im zweiten Adressbyte gesendet werden, sind auf Seite 9 im Diagramm 8 ersichtlich.

```
/************************Hauptprogramm**********************/

void main()
      {init_lcd();                /*LCD-Display initialisieren u.
      löschen*/
       blank_lcd();
       EAL=1;                     /*Interrupt 2 freigeben*/
       IT1=1;
       EX1=1;
       while(1)
           {if (flag==1)
                {i2c_start();    /* Startbedingung */
                 i2c_trans(0xA0);  /* Sendet die Slaveadresse des
```

```
                                PCF8583 (R/W=0) */
                i2c_trans(0x02);/* Sendet das 2. Adressbyte mit der
                                Adresse für den Sekundenregister */
                i2c_start();    /* Repeated start condition*/
                i2c_trans(0xA1);/* Sendet wieder die Slaveadresse
                                aber mit R/W=1 =>    lesen */
                serg=i2c_rec(); /* Empfängt den Wert den der PCF8583
                                auf den I²C-Bus geschickt hat (vom
                                Sekundenregister) */
                i2c_stop();          /* Stopbedingung */
                i2c_start();    /* Startbedingung */
                i2c_trans(0xA0);/* Slaveadresse */
                i2c_trans(0x03);/* Adresse für den Minutenregister */
                i2c_start();
                i2c_trans(0xA1);/* Slaveadresse mit R/W=1*/
                merg=i2c_rec(); /* Empfängt den Wert des
                                Minutenregister*/
                i2c_stop();     /* Stopbedingung */
                i2c_start();
                i2c_trans(0xA0);
                i2c_trans(0x04);/* Adresse für den Stundenregister */
                i2c_start();
                i2c_trans(0xA1);/*Slaveadresse mit R/W=1 */
                herg=i2c_rec(); /* Empfängt den Wert des
                                Stundenregister*/
                i2c_stop();
                sprintf(buffer,
                "Uhrzeit:%2x:%2x:%2x",(int)herg,(int)merg,(int)serg);
                print_lcd(1,1,buffer);     /*Ausgabe am LCD-Display*/
                printf("****************************************\n");
                flag=0;
               }
          }
     }
```

In dem vorigem Beispiel wurden nur Daten vom I²C-Bus gelesen. Im nächstem Beispiel wird zuerst einmal eine Uhrzeit in den Baustein PCF8583 geschrieben und danach wieder ausgelesen. Die Unterprogramme und das Hauptprogramm werden vom vorigen Programm übernommen, und es wird nur ein Unterprogramm UHRSET hinzugefügt und im Hauptprogramm einmal aufgerufen.

Beispiel 2:

```
void uhrset(char uhrzeitst,char uhrzeitmin,char uhrzeitsec)
                                /* Zeit:hh:mm:ss*/
     {i2c_start();              /* Startbedingung */
     i2c_trans(0xA0);           /* Sendet d. Slaveadresse mit
     R/W=>schreiben */
     i2c_trans(0x02);           /* Sendet das 2. Adressbyte für den
                                Sekundenregister */
     i2c_trans(uhrzeitsec);     /* Sendet den Wert für d.
     Sekundenregister(30)*/
     i2c_stop();                /* Stopbedingung */
     i2c_start();
     i2c_trans(0xA0);
     i2c_trans(0x03);           /* Sendet das 2. Adressbyte für den
                                Minutenregister */
```

```
     i2c_trans(uhrzeitmin);     / *Sendet den Wert für den
                                Minutenregister */
     i2c_stop();
     i2c_start();
     i2c_trans(0xA0);
     i2c_trans(0x04);           /* Sendet das 2. Adressbyte für den
                                Stundenregister */
     i2c_trans(uhrzeitst);      /* Sendet den Wert für den
                                Stundenregister */
     i2c_stop();
     }


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

void main()
     {init_lcd();                    /*LCD-Display initialisieren u.
                                     löschen*/
     blank_lcd();
     EAL=1;                               /*Interrupt 2 freigeben*/
     IT1=1;
     EX1=1;
     uhrset(0x10;0x20;0x30);              /* Zeit: 10:20:30 */
          while(1)
               {if (flag==1)
                    {i2c_start();         /* Startbedingung */
                     i2c_trans(0xA0);     /* Sendet die Slaveadresse
                                          des PCF8583 (R/W=0) */
                     i2c_trans(0x02);     /* Sendet das 2.Adressbyte
                                          mit der Adresse für den
                                          Minutenregister */
                          .
                          .
                          .
                          .
                          .
                     i2c_trans(0xA1);     /*Slaveadresse mit R/W=1*/
                     herg=i2c_rec();      /* Empfängt den Wert des
                                          Stundenregisters */
                     i2c_stop();
                     sprintf(buffer,"Uhrzeit:
                     %2x:%2x:%2x",(int)herg,(int)merg,(int)serg);
                     print_lcd(1,1,buffer);     /* Ausgabe am LCD-
                                                Display */
                     printf("**********************************\n");
                     flag=0;
               }
          }
     }
```