11. Die Serielle Schnittstelle beim 80C535

11.1. Allgemeines

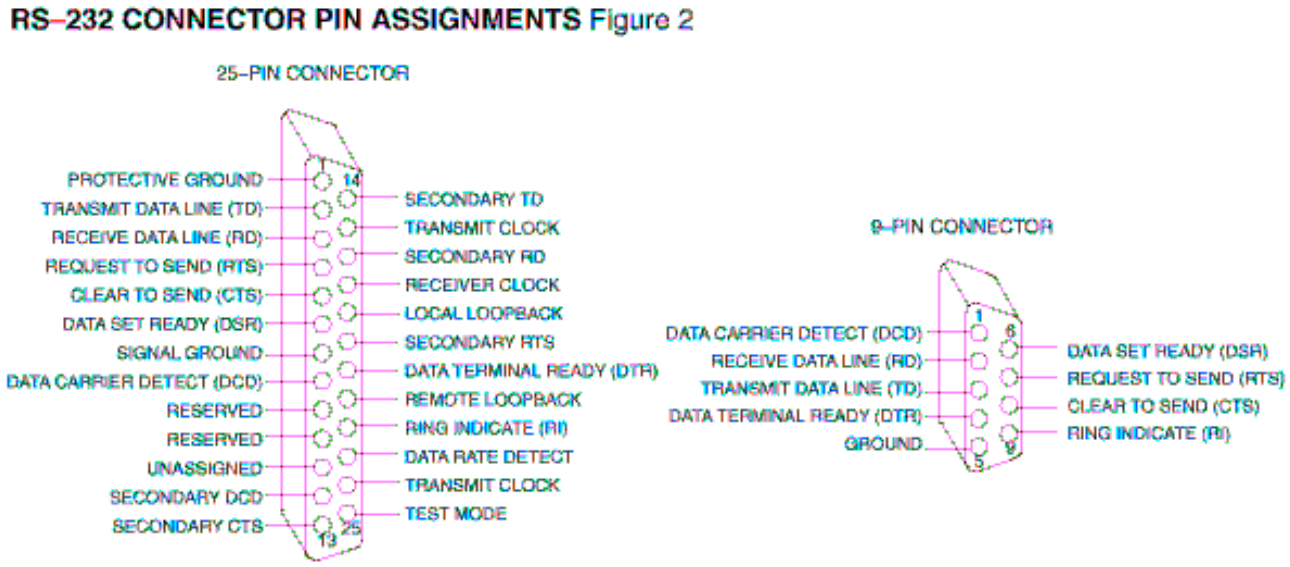
Die Asynchrone Serielle Schnittstelle wird durch die beiden Portpins **P3.0 (= RxD = Receive Data** = Datenempfang) und **P3.1 (= TxD = Transmit data** = Datensendung) realisiert. Dabei ist sogar „FULL DUPLEX OPERATION“ (= gleichzeitiges Senden und Empfangen möglich.
Hier nochmals zur Erläuterung das Datenformat bei einer solchen Übertragung:

Achtung: Die **“0ffizielle” serielle Schnittstelle RS232** schreibt die beiden Spannungen **“+12V” für log. Null** und **“-12V” für log. Eins** vor. Zum Glück sind **die meisten PCs** mit +5V bzw. Null Volt zufrieden **und arbeiten korrekt über die drei Pins TxD, RxD sowie Common (= Masse = Signal Ground).**
Ist das **nicht zulässig,** dann muss man besondere Schnittstellenbausteine (z. B. MAX232) zur Pegelanhebung einsetzen bzw. noch irgendwelche Portpins zur Darstellung der zusätzlichen RS232-Signale heranziehen.

Stiftbelegung für die beim Umgang mit dem PC wichtigen SUB D 9 - und SUB D 25 - Buchsen der seriellen Schnittstelle und die Bedeutung der Abkürzungen

Rxd.....Receive Data
COM.....Common (Signal Ground)
DSR.....Data Set Ready
CTS.....Clear To Send

TxD.....Transmit Data
DTR.....Data Terminal Ready
RTS.....Request To Send
DCD.....Data Carrier Detect



Zur grundsätzlichen Einarbeitung in diese Begriffe siehe die **Application Note Nr. 83 der Firma DALLAS** (Fundamentals of RS 232 Serial Communication) auf der Mikrocontroller - CD

11.2. Betriebsarten und beteiligte Register beim Mikrocontroller 80C535

Es gibt vier Betriebsarten für unsere Serielle Schnittstelle beim 80C535:

a) **„MODE 0“** bedeutet „**Synchroner Schieberegisterbetrieb**“. Dabei wird über den Pin „RxD“ ein Datenbyte ausgegeben oder eingelesen. Der Pin „TxD“ liefert (oder erhält!) den zugehörigen Schiebetakt:

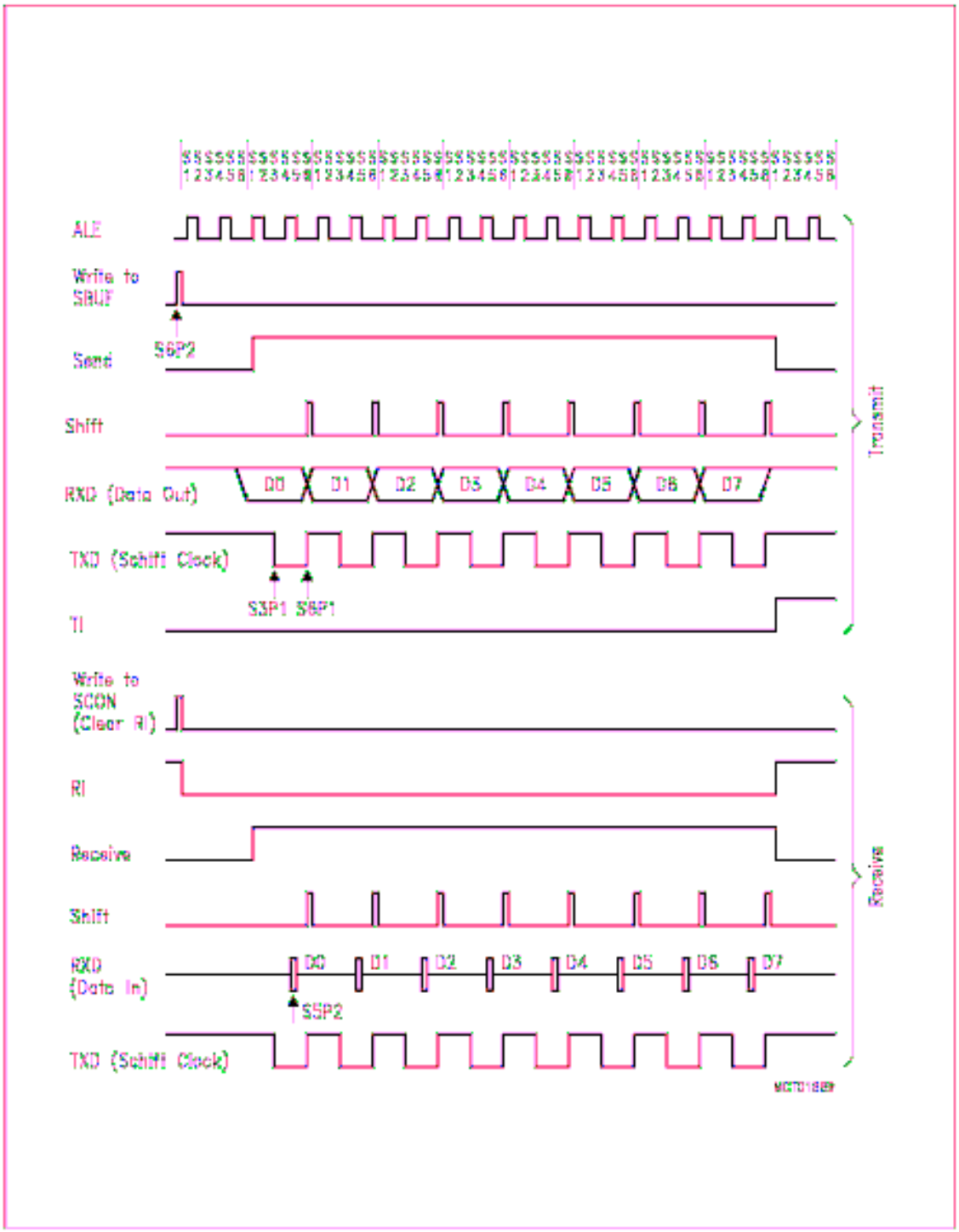


Figure 7-16 b)
Timing Diagram - Serial Interface, Mode 0

b) **Mode 1** ergibt eine **8 Bit - Übertragung mit Start- und Stoppbit**. Allerdings wird auf das 9. Bit (z. B. Parity) verzichtet. (Das bedeutet einen 8 Bit - UART). Zusammen mit dem Start- und Stoppbit werden also 10 Bits eingesetzt. Die Baudrate ist variabel.

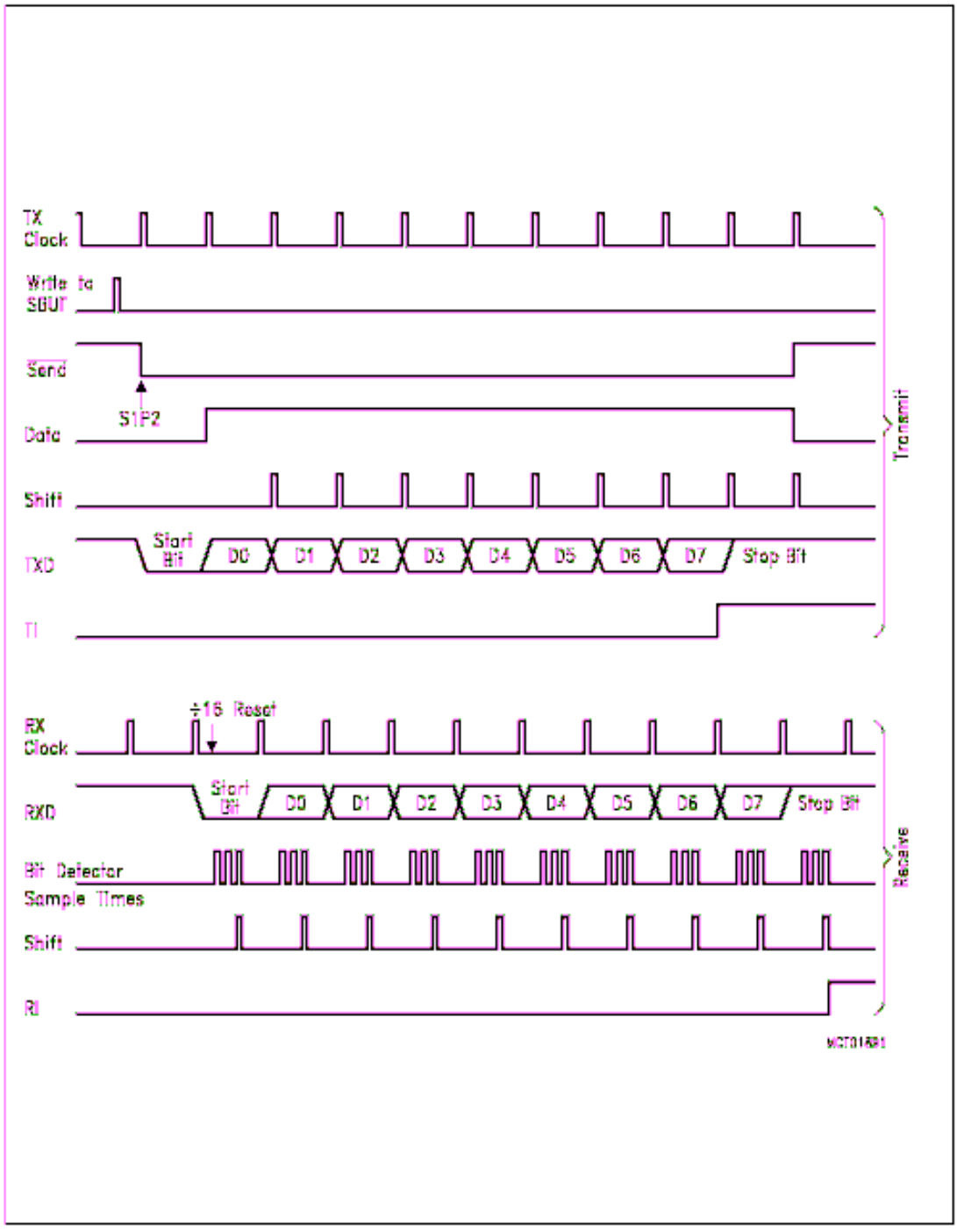


Figure 7-17 b)
Timing Diagram - Serial Interface, Mode 1

c) **MODE 2** verwendet nun **auch das 9. Datenbit** (z. B. als Parity - Bit oder zum Aufbau eines kleinen Mikrocontroller-Netzes,
bei dem auf diese Weise zwischen Adressbyte und Datenbyte unterschieden werden kann). Insgesamt werden also 11 Bit gesendet und wir haben einen 9 Bit - UART vor uns.
Vorsicht: Die **Baudrate liegt hier fest** und wird vom Oszillatortakt abgeleitet (wahlweise 1/32 oder 1/64 der Oszillatorfrequenz).

d) **MODE 3** stimmt vollkommen mit MODE 2 überein (= **9 Bit - UART**). Der einzige Unterschied ist die nun **wieder frei wählbare Baudrate** (wie bei MODE1, z. B. durch Verwendung von Timer 1 oder durch Einsatz des integrierten Baudraten-Generators).

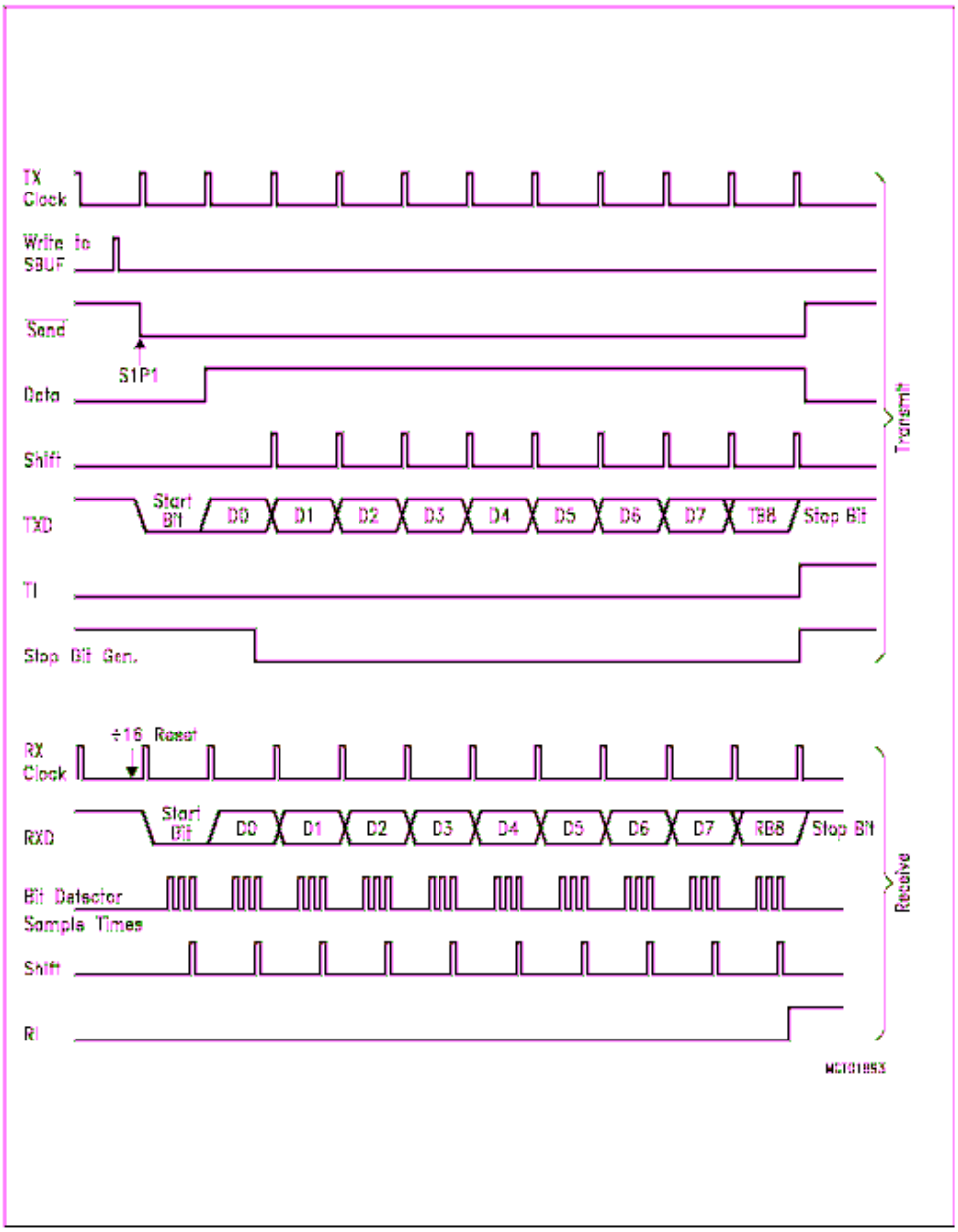


Figure 7-18 b)
Timing Diagram - Serial Interface, Modes 2 and 3

Für die praktische Arbeit mit der Schnittstelle bei unserem Microcontroller sind **zwei Register und zusätzlich zwei weitere Bits** wichtig:

a) **Kontrollregister SCON** (Adresse: 098H) für die **Programmierung der Schnittstelle**.

Bit SCON.7	BIT SCON.6	Bit SCON.5	Bit SCON.4	Bit SCON.3	Bit SCON.2	Bit SCON.1	Bit SCON.0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Erklärung: **SM0 und SM1** legen den **MODUS** (0...3) fest.

SM2 gibt die „Multiprozessor-Kommunikation“ frei (Siehe entsprechendes Kapitel im Handbuch)

REN heißt „Receive Enable“ = Empfängerfreigabe.

TB8 stellt das auszusendende **9. Datenbit** in Mode 2 und Mode 3 dar und wird per Software gesetzt oder gelöscht.

RB8: hier wird in **Mode 2 und Mode 3 das empfangene 9. Datenbit abgelegt**. (Bei Mode 1 stellt es das empfangene Stoppbit dar, falls SM2 = 0).

TI (= **Transmit-Interrupt-Flag**) zeigt das **Ende des Sendevorganges** (= Beginn des Stoppbits) bei einem ausgesendeten Byte an. Wird durch die Hardware gesetzt, muss jedoch per Software gelöscht werden.

RI (= **Receive Interrupt Flag**) meldet, wenn ein **Byte vollständig empfangen und im Empfangspuffer eingelesen** ist. Es wird folglich während des Stoppbits gesetzt. Muss per Software wieder gelöscht werden.

---

b) **Serial - Buffer - Register SBUF** (Adresse 099H). Dorthin werden die zu sendenden Bytes geschrieben oder von dort wird das empfangene Byte abgeholt.

ACHTUNG:
Sobald in dieses Register etwas hineingeschrieben wird, löst man damit automatisch den Sendevorgang aus!

Obwohl für Sendung und Empfang dieselbe Adresse gilt (= SBUF - Register 099h), gibt es hier keine Kollisionen. Der Chip-Konstrukteur hat nämlich dafür zwei getrennte Speicherplätze unter derselben Adresse reserviert, die gegeneinander verriegelt sind. So kann nie ein plötzlich empfangenes Byte das gerade gesendete Byte überschreiben.

Beim **Empfang** muss man jedoch trotzdem aufpassen und ein vollständig eingelaufenes Byte **sofort** (mit einem MOV...-Befehl) aus dem Empfangspuffer SBUF entfernen, denn das nächste einlaufende Byte überschreibt gnadenlos alles, was dort drinsteht....

---

c) **Bit “ADCON.7”** (= “BD”). Wird es **gesetzt**, so ist der extra vorhandene, **interne Baudratengenerator auf dem Chip eingeschaltet.**

Wird es **gesetzt**, dann ist der interne Baudratengenerator auf dem Chip eingeschaltet.
Ist es **gelöscht**, dann wird **Timer 1** als Baudratengenerator verwendet und man muss ihn entsprechend programmieren.

---

d) **Bit “PCON.7”** (= “SMOD”). Wird es **gesetzt**, dann **verdoppelt sich die Baudrate**.

ACHTUNG: das PCON - Register ist NICHT Bit-adressierbar!

Deshalb entweder den kompletten Registerinhalt neu laden oder ihn über eine ODER – bzw. UND - Verknüpfung ändern!
Beispiel: wird der interne Baudratengenerator benützt, dann erhält man bei gesetztem SMOD - Bit eine Rate von 9600 Baud, bei gelöschtem Bit dagegen 4800 Baud.

11.3. Betrieb der "Standard-Schnittstelle" im Mode 1 (8 Bit -UART)

Aufgabe: Erst beim Druck auf eine entsprechende „Starttaste“ an Portpin P1.0 wird alle 5 Millisekunden ein bestimmtes ASCII-Zeichen (= Ziffer oder Buchstabe nach Ihrer Wahl) im MODE 1 an Pin „TxD“ mit 9600 Baud ausgegeben.

a) Arbeiten Sie sich anhand der Anleitung auf den folgenden Seiten in die Kommunikation zwischen PC und Controllerboard über das WINDOWS-Tool „Hyperterminal“ ein und senden Sie dann dieses Zeichen zum PC!

b) Trennen Sie die Verbindung zwischen PC und Controllerboard auf und schließen Sie stattdessen die Buchsenplatine an Port 3 an. Messen Sie das Ausgangssignal mit dem Oszilloskop. Bestimmen Sie jedes einzelne Bit im Oszillogramm. Analysieren Sie den Zusammenhang zwischen den einzelnen messbaren Pulslängen und der Baudrate.

Lösungsweg für das Programm.:

a) SCON programmieren: Wir wählen MODE 1, löschen SM2, löschen REN, TB8, RB8, TI und RI.
Damit muss SCON mit „01000000b“ = **0x40** geladen werden.

b) Internen Baudratengenerator verwenden und auf 9600 Baud schalten.
Dazu müssen die Bits „BD“ und „SMOD“ gesetzt sein. Vorsicht: **Register PCON ist NICHT bit-adressierbar....**

Wir schreiben in das SBUF - Register das gewünschte ASCII-Zeichen (Siehe LCD-Display-Kapitel) und starten damit die Übertragung. Nach jedem Sendevorgang wird das TI-Flag automatisch gesetzt. Das werten wir aus, rufen die Zeitverzögerung auf, löschen das Flag wieder und wiederholen den Vorgang erneut.

Und das darf nur ablaufen, solange der an Pin P1.0 angeschlossene Startknopf gedrückt ist....

Zusatzaufgaben:

a) Halbierung der Baudrate.

b) Aussendung eines anderen Zeichens.

c) Wahl einer anderen Wartezeit in der Zeitschleife.

d) Einsatz der Interrupt-Steuerung

e) Schaffen Sie Auswahlmöglichkeiten zwischen verschiedenen Zeichen über mehrere Starttasten.

f) Senden Sie ein komplettes Wort oder einen größeren Text über die Schnittstelle.

Anleitung zur direkten Kommunikation von Mikrocontroller und PC über die RS232 und das Programm „Hyperterminal“

Systemvoraussetzungen:

PC mit Betriebssystem Microsoft Windows 95/98/NT/ME/XP
Keil-Software „µvision1“
Microcontrollerboard „Compuboard 80C535“

Vorgehensweise:

Schnittstellen-Programm zum Empfang oder zur Ausgabe eines Zeichens über die Schnittstelle (z. B. in C) erstellen, kompilieren, linken und ein Hexfile erzeugen.

Wichtig:

Im Controller-Programm unbedingt eine Drucktastensteuerung vorsehen, damit später die Verbindung zwischen KEIL-Debugger und Controllerboard problemlos unterbrochen werden kann und trotzdem ein Programmstart möglich ist!!

Das Programm in bekannter Weise in das RAM des Controllers überspielen und durch Eingabe von „**g main**“ starten. Das eigentliche Schnittstellenprogramm ist damit noch nicht aktiv, sondern nur die Tastenabfrage.

Nun wird der Debugger „dscope“ der KEIL-Software durch Druck auf „STOP“ im DEBUG-Window deaktiviert und anschließend vollständig geschlossen, da sonst noch die Schnittstelle (z. B. COM2) belegt ist !

Jetzt in Microsoft Windows wechseln und nacheinander folgende Schritte ausführen:

Start -> Zubehör-> Kommunikation -> **Hyper Terminal** öffnen.

Dann einen **Namen für die neue Verbindung** eingeben (z.B. RS232 und Symbol rotes Telefon)

OK

Im untersten Pulldown-Menü einstellen:

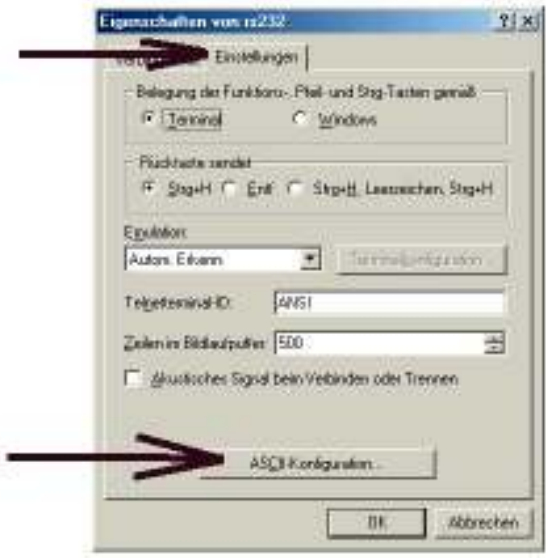
Verbinden über COM2

OK

Richtige Anschlusseinstellungen wählen**:**

9600 Baud / 8 Datenbit / no Parity / 1 Stopp-Bit / Kein Protokoll

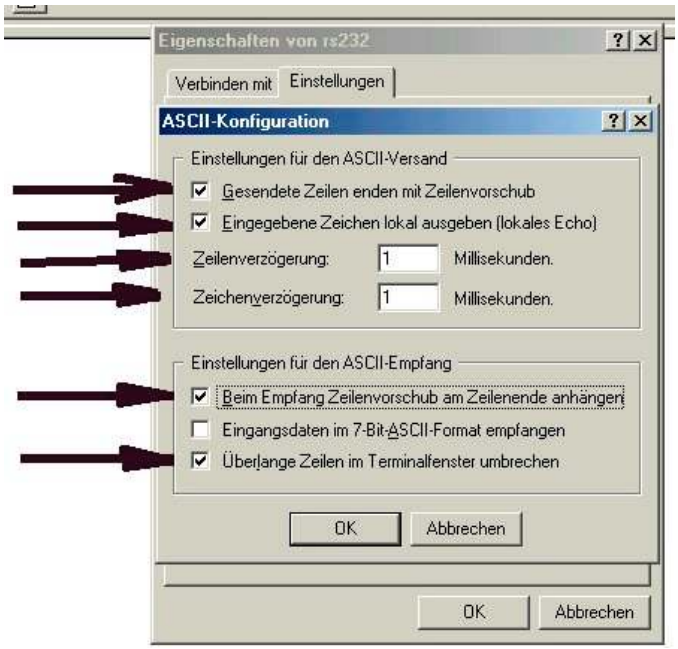
OK



Nun im Menü „**Datei**“ auf „**Eigenschaften**“ klicken.

Dann „**Einstellungen**“ und „**ASCII- Konfiguration**“ aufrufen.

(Der Rest der Einstellungen bleibt unverändert und sollte dem nebenstehenden Bild entsprechen).



Nun vier Häkchen an den bezeichneten Stellen setzen und zweimal eine Zeitverzögerung von 1 Millisekunde vorgeben.

OK

Jetzt kann man die Starttaste für das Sende- oder Empfangsprogramm beim Controllerboard drücken und anschließend entweder den PC-Bildschirm beobachten oder über **die Tastatur des PC ein oder mehrere Zeichen zum Mikrocontroller senden.**

Diese Zeichen können dann vom Controllerboard z. B. in Form der LED- Kette an Port P1 oder direkt auf dem LCD- Display angezeigt werden usw.

Parallel dazu kann man immer alle gesendeten Zeichen als Echo auf dem PC-Schirm kontrollieren.

Lösung der ersten Aufgabe in C: Baudrate = 9600 Bit pro Sekunde

```
/*---------------------------------------------------------------
Programmbeschreibung
---------------------------------------------------------------
Name:           rs_232_1.c
Funktion:       Über den Pin Txd (= Portpin P3.1) der
                Seriellen Schnittstelle des Controllers
                wird „auf Knopfdruck“ dauernd dasselbe Zeichen mit 9600 Baud
                ausgegeben, wobei zwischen jeden Sendevorgang
                 eine Pause von 5 Millisekunden gelegt wird.
                Es wird der interne Baudratengenerator
                benützt. Das Ausgangssignal kann nun mit
                dem Oszilloskop gemessen und analysiert
                werden.


Datum:          24. 01. 2005
Autor:          G. Kraus

---------------------------------------------------------------
Deklarationen und Konstanten
---------------------------------------------------------------*/

#include <reg515.h>         //Registersatz des 80C535 verwenden
#include <stdio.h>

sbit start=0x90;            //Portpin P1.0 als Starttaste
/*---------------------------------------------------------------
Hauptprogramm
---------------------------------------------------------------*/
void zeit(void);            //Prototypen-Anmeldung der Verzögerung

void main(void)
{   SCON=0x40;              //Mode 1 (Startbit, 8 Datenbits, no parity)
    BD=1;                   //Internen Baudratengenerator einschalten
    PCON=PCON |128;         //Baudrate auf 9600 einstellen
        P1=0xFF;            //Port P1 auf Lesen schalten
    TI=0;                   //Sendeflag löschen

  while(1)
  {  while(start==0)

     {  SBUF=0x35;          //Sende ASCII-Wert für 5
        while(TI==0);       //Senden beendet?
        TI=0;               //Wenn ja: Sendeflag löschen
        zeit();             //Delay von 5000 Mikrosekunden
     }
  }
}
/*---------------------------------------------------------------
Zusatzfunktionen
---------------------------------------------------------------*/
void zeit(void)
{
  unsigned int x;
  for(x=0;x<=1000;x++);     //Int-Wert 1000 ergibt etwa 5 Millisekunden
}

//---------------------------------------------------------------
```

Lösung der zweiten Aufgabe in C: Baudrate = 4800 Bit pro Sekunde

```
/*-------------------------------------------------------------------
Programmbeschreibung
-------------------------------------------------------------------
Name:           rs_232_1.c
Funktion:       Über den Pin Txd (= Portpin P3.1) der
                Seriellen Schnittstelle des Controllers
                wird „auf Knofdruck“ dauernd dasselbe Zeichen mit 4800 Baud
                ausgegeben, wobei zwischen jeden Sendevorgang
                eine Pause von ca. 3,7 Millisekunden gelegt wird.
                Es wird der interne Baudratengenerator
                benützt. Das Ausgangssignal kann nun mit
                dem Oszilloskop gemessen und analysiert
                werden. Das Board sendet nur, solange die
                Starttaste an P1.0 (LOW-aktiv!) gedrückt ist.


Datum:          24. 01. 2005
Autor:          G. Kraus

-------------------------------------------------------------------
Deklarationen und Konstanten
-------------------------------------------------------------------*/

#include <reg515.h>             //Registersatz des 80C535 verwenden
#include <stdio.h>

sbit start=0x90;                //Portpin P1.0 als Starttaste
/*-------------------------------------------------------------------
Hauptprogramm
-------------------------------------------------------------------*/
void zeit(void);                //Prototypen-Anmeldung der Verzögerung

void main(void)
{     SCON=0x40;                //Mode 1 (Startbit, 8 Datenbits, no parity)
      BD=1;                     //Internen Baudratengenerator einschalten
      PCON=PCON & 0x7F;         //Baudrate auf 4800 einstellen
      P1=0xFF;                  //Port P1 auf Lesen schalten
      TI=0;                     //Sendeflag löschen

    while(1)
    {     while(start==0)
          {   SBUF=0x31;        //Sende ASCII-Wert für 3
              while(TI==0);     //Senden beendet?
              TI=0;             //Wenn ja: Sendeflag löschen
              zeit();           //Delay von ca. 5000 Mikrosekunden
          }
    }
}
/*-------------------------------------------------------------------
Zusatzfunktionen
-------------------------------------------------------------------*/
void zeit(void)
{
    unsigned int x;
    for(x=0;x<=1000;x++);       //Int-Wert 1000 ergibt etwa 5 Millisekunden
}
```

Lösung der dritten Aufgabe in C: Baudrate = 9600 Bit pro Sekunde und abwechselnde Ausgabe von zwei verschiedenen Zeichen

```
/*---------------------------------------------------------------------
Programmbeschreibung
---------------------------------------------------------------------
Name:           rs_232_3.c
Funktion:       Über den Pin Txd (= Portpin P3.1) der
                Seriellen Schnittstelle des Controllers
                werden abwechselnd zwei Zeichen mit 9600 Baud
                ausgegeben, wobei zwischen jeden Sendevorgang
                eine deutliche Pause gelegt wird.
                Es wird der interne Baudratengenerator
                benützt.
                Nach dem Programmstart muss zuerst die Verbindung
                Zwischen PC und Controllerboard unterbrochen
                werden, da sonst die KEIL-Software Probleme
                macht.
                Verbindet man nun die Pins TxD und RxD mit einer
                Kabelbrücke, so können die auf den Tastendruck
                hin gesendeten Zeichen gleich wieder empfangen und
                an Port P1 mit der LED-Kette sichtbar gemacht werden.


Datum:          24. 01. 2005
Autor:          G. Kraus

---------------------------------------------------------------------
Deklarationen und Konstanten
---------------------------------------------------------------------*/

#include <reg515.h>         //Registersatz des 80C535 verwenden
#include <stdio.h>

bit warten=1;
sbit starten=0x90;          //Portpin P1.0 als Starttaste
/*---------------------------------------------------------------------
Hauptprogramm
---------------------------------------------------------------------*/
void ISR_RS(void);          //Prototypen-Anmeldung
void zeit(void);

void main(void)
{       SCON=0x40;              //Mode 1 (Startbit, 8 Datenbits, no parity)
        BD=1;                   //Internen Baudratengenerator einschalten
        REN=1;                  //Empfänger einschalten
        EAL=1;                  //Alle Interrupts freigeben
        ES=1;                   //Interrupt der RS232 freigeben
        PCON=PCON | 0x80;       //Baudrate auf 9600 einstellen
        P1=0xFF;                //Port P1 auf Lesen schalten
        TI=0;                   //Sendeflag löschen
        RI=0;                   //Empfangsflag löschen

        while(1)
        {   while(starten==0)
            {       SBUF=0x31;          //Sende ersten ASCII-Wert
                    while(warten==1);   //Senden beendet?
                    warten=1;            //Setze wieder das Warte-flag
                    SBUF=0x35;          //Sende zweiten ASCII-Wert
                    while(warten==1);   //Senden beendet?
                    warten=1;           //Setze wieder das Warte-Flag
            }
        }
}
```

```
/*---------------------------------------------------------------------------
Zusatzfunktionen
---------------------------------------------------------------------------*/
void zeit(void)
{           unsigned int x;
            for(x=0;x<=30000;x++);          //Wartezeit produzieren
}



void ISR_RS (void)  interrupt 4             //RS232 hat Int-Vektor 0x23
{           if(TI==1)                       //Sende-Flag auswerten
            {       TI=0;                   //TI-Flag löschen
                    zeit();                 //warten
            }

             if (RI==1)                     //Empfangs-Flag auswerten
            {       RI=0;                   //RI-Flag löschen
                    P1=SBUF;                //Empfangenes Byte an P1 übergeben
                    zeit();                 //warten
            }
             warten=0;                      //Warte-Flag löschen
}
```

Lösung der vierten Aufgabe in C: Ausgabe unterschiedlicher Texte beim Drücken bestimmter Tasten (9600 Baud)

```
/*---------------------------------------------------------------
Programmbeschreibung
---------------------------------------------------------------
Name:           rs_232_4.c
Funktion:       Sobald man eine bestimmte, an Port P4 angeschlossenene
                Taste drückt, werden über den Pin Txd (= Portpin P3.1)
                der Seriellen Schnittstelle des Controllers
                die einzelnen Zeichen einer bestimmten Zeichenkette
                (= 16 Zeichen) mit 9600 Baud ausgegeben.
                Als Abschluß des Sendevorganges soll "\0" übergeben
                werden.
                Zwischen jedes gesendete Byte soll eine Pause gelegt
                werden. Gleichzeitig gibt das Programm die gesendeten
                Zeichen an Port P1 aus und macht sie durch die LED-Kette
                sichtbar.
                Es wird der interne Baudratengenerator
                benützt.


Datum:          24. 01. 2005
Autor:          G. Kraus

---------------------------------------------------------------
Deklarationen und Konstanten
---------------------------------------------------------------*/

#include <reg515.h>             //Registersatz des 80C535 verwenden
#include <stdio.h>

sbit Taste0=0xE8;               //Taste0 hängt an Portpin P4.0 und gibt Text0 aus
sbit Taste1=0xE9;               //Taste1 hängt an Portpin P4.1 und gibt Text1 aus
sbit Taste2=0xEA;               //Taste2 hängt an Portpin P4.2 und gibt Text2 aus
/*---------------------------------------------------------------
Hauptprogramm
---------------------------------------------------------------*/


void zeit(void);
void send_rs(unsigned char *ptr);

void main(void)
{       unsigned char text0[17]={"RS2-232-Test   !"};
        unsigned char text1[17]={"Dasselbe nochmal"};
        unsigned char text2[17]={"Alles klar     !"};

        SCON=0x40;                      //Mode 1 (Startbit, 8 Datenbits, no parity)
        BD=1;                           //Internen Baudratengenerator einschalten
        P4=0xFF;                        //Port P4 auf Lesen schalten
        PCON=PCON | 0x80;               //Baudrate auf 9600 einstellen

        while(1)                        //Endlosschleife
        {       if(Taste0==0)           //Taste 0 gedrückt?
                {       send_rs(text0);  //Wenn ja, dann sende den Text 0
                        SBUF='\0';      //Zeichenkette mit \0 abschließen
                         P1='\0';       ///\0 an LEDs anzeigen
                }

                if(Taste1==0)           //Taste 1 gedrückt?
                {       send_rs(text1); //Wenn ja, dann sende den Text 1
                        SBUF='\0';       //Zeichenkette mit \0 abschließen
                        P1='\0';         ///\0 an LEDs anzeigen
                }

                if(Taste2==0)           //Taste 2 gedrückt?
```

```
                {       send_rs(text2);     //Wenn ja, dann sende den Text 2
                        SBUF='\0';          //Zeichenkette mit \0 abschließen
                        P1='\0';            //\0 an LEDs anzeigen
                }
        }
}
/*-------------------------------------------------------------------------
Zusatzfunktionen
-------------------------------------------------------------------------*/
void send_rs(unsigned char *ptr)
{       while(*ptr)                         //solange senden, bis "\0" erreicht ist
        {       SBUF=*ptr;                  //Byte senden
                P1=*ptr;                    //Byte an LEDs ausgeben
                ptr++;                      //Nächstes Byte adressieren
                zeit();                     //Wartezeit aufrufen

        }
}

//-------------------------------------------------------------------------

void zeit(void)
{       unsigned int x;
        for(x=0;x<=30000;x++);              //Wartezeit produzieren
}
```

11. 4. Ausgabe von empfangenen ASCII-Zeichen auf dem LCD-Display

Sie sollen nun über die PC-Tastatur verschiedene Zeichen oder komplette Worte eingeben und (mittels des HyperTerminals) zum Controllerboard senden. Übergeben Sie jedes empfangene Zeichen an das (an Port P5 angeschlossene) LCD- Display und lassen Sie dieses Zeichen dort anzeigen. Werden mehr als 16 Zeichen oder „\0“ empfangen, dann beginnt die Anzeige wieder am Anfang des Displays.
Verwenden Sie dazu den bereits vorhandenen (und z. B. beim Digitalvoltmeter eingesetzten) Modul „LCD_CTRL_new.c“ und verbinden Sie ihn mit einem selbst geschriebenen „main“-Programm.

Entwickeln Sie zuerst den Lösungsweg mit Hilfe eines Programm-Ablauf-Plans (PAP).

Lösung: Anzeige empfangener Zeichen auf dem LCD-Display

```
/*Achtung, Achtung, Achtung!
Die KEIL-Software reagiert oft griesgrämig und in nicht
vorhersehbarer Weise auf Experimente mit der RS232
des Boards (z. B. werden Zeichen verschluckt oder
Leerzeichen eingefügt oder sonstiger Unfug ausgeführt..)

Deshalb wird dieses Programm mit einer Starttaste ausgerüstet, die den
Portpin P1.1 beim Drücken an Masse legt und erst dadurch die
Schnittstelle einschaltet. So kann in gewohnter Weise erst das
Programm gestartet werden, ohne dass Probleme zu befürchten sind.
Zur Kommunikation mit dem Hyperterminal wird dann einfach im
dscope-Debugger der "Stop"-Button und anschließend "Abort driver"
gedrückt.
Nun kann die Verbindung zwischen Controllerboard und Hyperterminal
eingerichtet und das eigentliche Empfangsprogramm durch einen Knopfdruck
gestartet und getestet werden.

/*-------------------------------------------------------------------
Programmbeschreibung
-------------------------------------------------------------------
Name:           rs_232_6.c
Funktion:       Empfangs- und Anzeigeprogramm für die
                Serielle Schnittstelle
                Das Programm zeigt maximal 16 einzelne empfangene
                Zeichen der Seriellen Schnittstelle sofort an.
                Trifft "\0" ein oder werden mehr als 16 Zeichen empfangen,
                dann wird der Cursor wieder an den Anfang des Displays
                gestellt und der alte Text überschrieben.

ACHTUNG:        Dieses Programm muß vom Linker anschließend mit dem
                LCD-Anzeigeprogramm "LCD_CTRL_new.C" zusammengebunden
                werden. Deshalb ist ein entsprechender Eintrag in
                der Projektliste (mit "Edit Project" anzusehen...)
                und eine Prototypen-Deklaration aller im Anzeige-
                modul verwendeten Funktionen in diesem Programm
                erforderlich (Vergleiche Programm "LCD_DEMO.C").

Datum:          10. 01. 2005
Autor:          G. Kraus
-------------------------------------------------------------------
Deklarationen und Konstanten
-------------------------------------------------------------------*/

#include <reg515.h>             //Registersatz des 80C535 verwenden
#include <stdio.h>

void zeit_s(void);              //Prototypen des LCD-Display-Moduls
void zeit_l(void);
void lcd_com(void);
void lcd_ein(void);
void lcd_ini1(void);
void switch_z1(void);
void switch_z2(void);
void show_text(char *ptr);
void show_char(char *zeichen);

sbit receive=0x90;                      /*Empfang erst möglich,
                                        sobald Taste an Portpin
                                        P1.0 gedrückt und dadurch
                                        dieser Pin an Masse gelegt wird*/
/*-------------------------------------------------------------------
Hauptprogramm
-------------------------------------------------------------------*/
```

```
void main(void)
{       unsigned char message[17]={"Wartestellung   "};
        unsigned char x=1,y=0;

        SCON=0x40;                      //Mode 1 (Startbit, 8 Datenbits, no parity)
        BD=1;                           //Internen Baudratengenerator einschalten

        PCON=PCON | 0x80;               //Baudrate auf 9600 einstellen

        RI=0;                           //Empfangsflag löschen

        lcd_ein();                      //Einschaltroutine für Display
        lcd_ini1();                     //Einstellung der Display-Betriebswerte
        switch_z1();                    //Schalte auf den Anfang von Zeile 1
        show_text(message);             //Zeige den Text "Wartestellung" an
        switch_z1();                    //Schalte erneut auf den Anfang von Zeile 1
        REN=1;                          //Empfänger einschalten

        while(receive==1);              //Bei Tastendruck Empfang freigeben

        while(1)                        //Auf Empfang gehen
        {       while(RI==0);           //Warten auf ein Byte
                x=SBUF;                 //Eingelaufenes Byte kopieren
                show_char(&x);          //und anzeigen
                y++;                    //Angezeigte Zeichen zählen
                if((x=='\0') || (y>15))  //Ende der Anzeige erkennen
                {       x=1;            //Anzeigepuffer darf nicht leer sein
                        y=0;            //Zeichenzähler auf Null stellen
                        switch_z1();    //Auf Zeile 1 umschalten
                }
                RI=0;                   //Empfangsflag wieder löschen
        }

}
```

Kombiniertes Sende- und Empfangsprogramm

```
/*------------------------------------------------------------------
Programmbeschreibung
------------------------------------------------------------------
Name:           rs_232_7.c

Funktion:       Es handelt sich um ein interruptgesteuertes
                Sende- und Empfangsprogramm. Mit der Taste an
                P4.0 wird das Programm grundsätzlich aktiviert.
                Durch einen Druck auf die LOW-aktiven Tasten an den
                Portpins P4.1, P4.2 und P4.3 können dann
                drei verschiedene Text gesendet werden.
                Über den Pin Txd (= Portpin P3.1) der
                Seriellen Schnittstelle des Controllers
                wird der Text mit 9600 Baud
                ausgegeben, wobei zwischen jeden Sendevorgang
                eine deutliche Pause gelegt wird.
                Es wird der interne Baudratengenerator
                benützt. Verbindet man nun NACH dem Programmstart
                die Pins TxD und RxD mit einer Kabelbrücke, so
                können die gesendeten Zeichen gleich wieder empfangen
                und über Port P5 zum LCD-Display geschickt weerden.
                Während des Sendevorganges leuchten die Leds an
                Port P1.
                Bitte im Dauerbetrieb UNBEDINGT das RS232-
                Verbindungskabel zwischen Controllerboard und PC
                entfernen, sobald das Programm mit "g main"
                gestartet wird  --  und BEVOR die Starttaste an
                P4.0 gedrückt wird.

Datum:          10. 01. 2005
Autor:          G. Kraus

------------------------------------------------------------------
Deklarationen und Konstanten
------------------------------------------------------------------*/

#include <reg515.h>                     //Registersatz des 80C535 verwenden
#include <stdio.h>

void zeit_s(void);                      //Prototypen des LCD-Display-Moduls
void zeit_l(void);
void lcd_com(void);
void lcd_ein(void);
void lcd_ini1(void);
void switch_z1(void);
void switch_z2(void);
void show_text(char *ptr);
void show_char(char *zeichen);

void zeit(void);
void send_rs(char *ptr);

sbit starten=0xE8;                      //Starttaste an Portpin P4.0
sbit Taste0=0xE9;                       //Taste0 hängt an Portpin P4.1 und gibt Text0 aus
sbit Taste1=0xEA;                       //Taste1 hängt an Portpin P4.2 und gibt Text1 aus
sbit Taste2=0xEB;                       //Taste2 hängt an Portpin P4.3 und gibt Text2 aus

/*------------------------------------------------------------------
Hauptprogramm
------------------------------------------------------------------*/

void ISR_RS(void);                      //Prototypen-Anmeldung
void zeit(void);
```

```
void main(void)
{       unsigned char message[17]={"Wartestellung   "};
        unsigned char text0[17]=  {"RS232-Test     !"};
        unsigned char text1[17]=  {"Dasselbe nochmal"};
        unsigned char text2[17]=  {"Alles klar     !"};

        SCON=0x40;                              //Mode 1 (Startbit, 8 Datenbits, no parity)
        BD=1;                                   //Internen Baudratengenerator einschalten
        PCON=PCON | 0x80;                       //Baudrate auf 9600 einstellen
        lcd_ein();                               //Einschaltroutine für Display
        lcd_ini1();                             //Function set für Display
        switch_z1();                            //Cursor auf Zeilenanfang stellen
        show_text(message);                     //Text "Wartestellung" anzeigen
        switch_z1();                            //Cursor wieder auf Zeilenanfang stellen
        P4=0xFF;                                //Port P4 auf Lesen schalten

        while(starten==1);                      //Starttaste gedrückt?
        REN=1;                                  //Empfänger einschalten
        ES=1;                                   //Interrupt der RS232 freigeben
        EAL=1;                                  //Alle Interrupts freigeben
        TI=0;                                   //Sendeflag löschen
        RI=0;                                   //Empfangsflag löschen

        while(1)
        {       if(Taste0==0)                           //Taste 0 gedrückt?
                {       P1=0xFF;                        //LEDs an Port P1 ein
                        send_rs(text0);                 //Wenn ja, dann sende den Text 0
                        SBUF='\0';                      //Zeichenkette mit \0 abschließen
                        P1='\0';                        //Sende-Ende an LEDs anzeigen
                }

                if(Taste1==0)                           //Taste 1 gedrückt?
                {       P1=0xFF;                        //LEDs an Port P1 ein
                        send_rs(text1);                 //Wenn ja, dann sende den Text 1
                        SBUF='\0';                      //Zeichenkette mit \0 abschließen
                        P1='\0';                        //Sende-Ende an LEDs anzeigen
                }

                if(Taste2==0)                           //Taste 2 gedrückt?
                {       P1=0xFF;                        //LEDs an Port P1 ein
                        send_rs(text2);                 //Wenn ja, dann sende den Text 2
                        SBUF='\0';                      //Zeichenkette mit \0 abschließen
                        P1='\0';                        //Sende-Ende an LEDs anzeigen
                }
        }
}
/*---------------------------------------------------------------------
Zusatzfunktionen
---------------------------------------------------------------------*/
void zeit(void)
{
    unsigned int x;
    for(x=0;x<=30000;x++);                      //Wartezeit produzieren
}
//---------------------------------------------------------------------

void send_rs(unsigned char *ptr)
{       while(*ptr)                             //solange senden, bis "\0" erreicht ist
        {       SBUF=*ptr;                      //Byte senden
                P1=*ptr;                        //Byte an LEDs ausgeben
                ptr++;                          //Nächstes Byte adressieren
                zeit();                         //Wartezeit aufrufen
        }
}
//---------------------------------------------------------------------
```

```
void ISR_RS (void)   interrupt 4                    //RS232 hat Int-Vektor 0x23 = 35d
{           char x=1,y=0;

            if(TI==1)                               //Sende-Flag auswerten
            {           TI=0;                       //TI-Flag löschen
            }

            if (RI==1)                              //Empfangs-Flag auswerten
            {           RI=0;                       //RI-Flag löschen
                        x=SBUF;                     //Empfangenes Byte an x übergeben
                        show_char(&x);              //Empfangenes Byte anzeigen
                        y++;                        //Angezeigte Zeichen zählen

                        if((x=='\0') || (y>16))     //Ende der Anzeige erkennen
                        {           x=1;            //Anzeigepuffer darf nicht leer sein
                                    y=0;            //Zeichenzähler auf Null stellen
                                    switch_z1();    //Auf Zeile 1 umschalten
                        }

            }

}
```