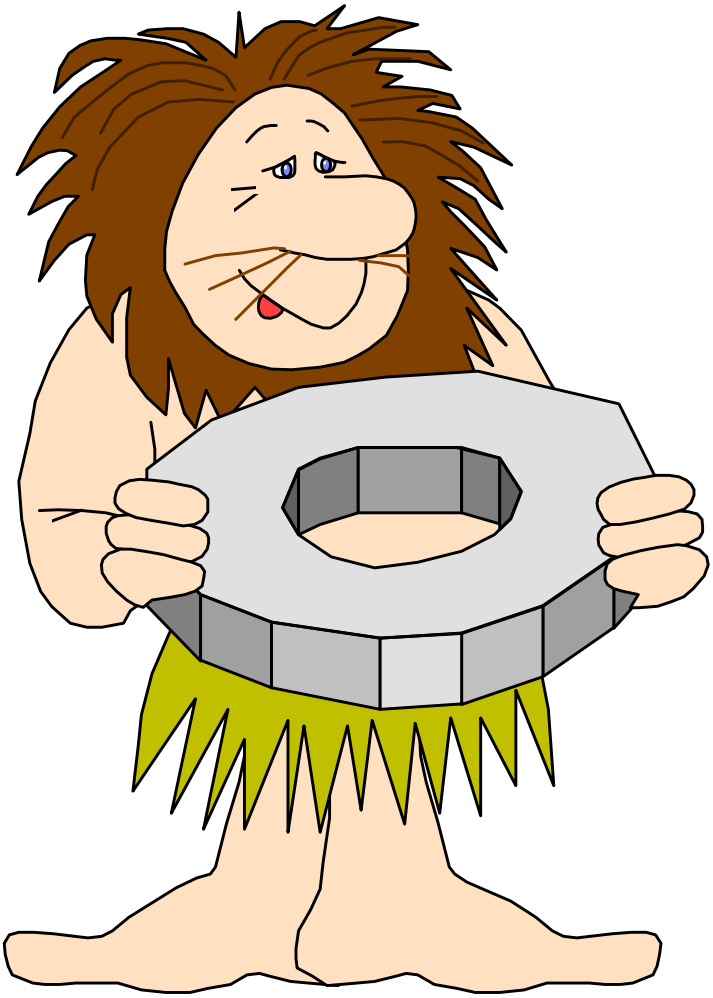


MODUL 3
Bedienung der seriellen Schnittstelle und der LCD-Anzeige

V 1.1 J. Humer 1997

INHALTSVERZEICHNIS
MODUL 3
Bedienung der seriellen Schnittstelle und der LCD Anzeige

1 Aus- und Eingabe über die serielle Schnittstelle

Die Programmiersprache C stellt einige Funktionen zur Ein- und Ausgabe von Daten zur Verfügung. C-Compiler wie Turbo C 2.0, mit welchen Programme für den PC erstellt werden können, geben die Daten standardmäßig auf dem Bildschirm aus und lesen standardmäßig Daten von der Tastatur ein.
Für Mikrocontroller wäre dies jedoch nicht sinnvoll. Da alle Derivate der Familie 8051 zumindest mit einer seriellen Schnittstelle ausgestattet sind, wird bei Ein- und Ausgabefunktionen standardmäßig auf die serielle Schnittstelle zugegriffen.

Sämtliche Ein- und Ausgabefunktionen für die serielle Schnittstelle sind in der Standard Bibliothek **STDIO.H** (= Run Time Library STDIO.H) enthalten. Um die Bibliothek in das Programm mit einzubinden, muß daher bei der Angabe der Include Dateien die Zeile

#include <stdio.h>

eingegeben werden.

Alle Funktionen dieser Bibliothek bauen auf zwei grundlegende Funktionen auf :

- Alle Funktionen, welche über die serielle Schnittstelle Daten einlesen, bauen auf die Funktion char _GETKEY ( ) auf.
- Alle Funktionen, welche über die serielle Schnittstelle Daten ausgeben, bauen auf die Funktion _PUTCHAR (char) auf.

1.1 Die Funktionen GETKEY und PUTCHAR

Diese beiden Funktionen dienen als Grundlage für alle Ein- und Ausgabefunktionen.

1.1.1 PUTCHAR (char)

Standardmäßig dient die Funktion PUTCHAR (char) dazu, das übergebene Zeichen über die serielle Schnittstelle 0 auszugeben.
Für diese Funktion wird der Source Code mit dem Compiler mitgeliefert. Das heißt, man hat die Möglichkeit, durch die Änderung des Source Codes dieser Funktion die Ausgabe aller in der Bibliothek enthaltenen Funktionen wie printf und puts auf eine andere Ausgabequelle umzuleiten. Eine andere Ausgabequelle kann z.B. die zweite serielle Schnittstelle oder die LCD Anzeige sein.

Beispiel: Der Standard Source Code der Funktion PUTCHAR (char) lautet folgendermaßen:

```
/***********************************************************************/
/*  This file is part of the C-51 Compiler package                     */
/*  Copyright KEIL ELEKTRONIK GmbH 1990                                */
/***********************************************************************/
/* PUTCHAR.C:  This routine is the general character output of C-51.*/
/***********************************************************************/
#include <reg51.h>

#define XON  0x11
#define XOFF 0x13

char putchar (char c)  {
  if (c == '\n')  {
    if (RI)  {
      if (SBUF == XOFF)  {
        do  {
          RI = 0;
          while (!RI);
        }
        while (SBUF != XON);
        RI = 0;
      }
    }
    while (!TI);
    TI = 0;
    SBUF = 0x0d;                          /* output CR  */
  }
  if (RI)  {
    if (SBUF == XOFF)  {
      do  {
        RI = 0;
        while (!RI);
      }
      while (SBUF != XON);
      RI = 0;
    }
  }
  while (!TI);
  TI = 0;
  return (SBUF = c);
}
```

Das Monitorprogramm des 80C517 arbeitet kann für die serielle Schnittstelle 0 oder 1 erstellt werden. Beide Varianten haben ihre Berechtigung. Sollte die serielle Schnittstelle 1 verwendet werden, muß wie nachstehend beschrieben verfahren werden.

```
/*********************************************************************/
/*  AENDERUNG: SERIELLE AUSGABE AUF SCHNITTSTELLE 1 UMGELEITET      */
/*********************************************************************/
#include <reg517.h>

#define XON  0x11
#define XOFF 0x13

char putchar (char c)  {
  if (c == '\n')  {
    if ((S1CON & 0x01) == 1)  {
      if (S1BUF == XOFF)  {
        do  {
          S1CON = S1CON & 0xFE;
          while ((S1CON & 0x01) == 0);
        }
        while (S1BUF != XON);
        S1CON = S1CON & 0xFE;
      }
    }
    while ((S1CON & 0x02) == 0);
    S1CON = S1CON & 0xFD;
    S1BUF = 0x0d;                          /* output CR  */
  }
  if ((S1CON & 0x01) == 1)  {
    if (S1BUF == XOFF)  {
      do  {
          S1CON = S1CON & 0xFE;
        while ((S1CON & 0x01) == 0);
      }
      while (S1BUF != XON);
      S1CON = S1CON & 0xFE;
    }
  }
  while ((S1CON & 0x02) == 0);
  S1CON = S1CON & 0xFD;
  return (S1BUF = c);
}
```

1.1.2 char GETKEY ( )

Die Funktion GETKEY ( ) liest einZeichen von der seriellen Schnittstelle des 8051 ein. Ist kein Zeichen vorhanden, so wird das Einlaufen eines Zeichens abgewartet.
Auch für diese Funktion wird der Source Code mitgeliefert und kann geändert werden, um die Eingabe an die jeweilige Eingabequelle anzupassen.

Beispiel: Änderung, um über die serielle Schnittstelle 1 des 80C517 einzulesen.
Die Standard-Funktion für getkey() lautet:

```
/*--------------------------------------------------------------------*/
/*  This file is part of the C-51 Compiler package                    */
/*  Copyright KEIL ELEKTRONIK GmbH 1990                               */
/*--------------------------------------------------------------------*/
/*  GETKEY.C:  This routine is the general character input of C-51.   */
/*--------------------------------------------------------------------*/

#include <reg51.h>

char _getkey ()  {
  char c;

  while (!RI);
  c = SBUF;
  RI = 0;
  return (c);
}
```

Die geänderte Funktion:

```
/*--------------------------------------------------------------------*/
/* AENDERUNG: SERIELLE AUSGABE AUF SCHNITTSTELLE 1 UMGELEITET         */
/*--------------------------------------------------------------------*/

#include <reg517.h>

char _getkey ()  {
  char c;

  while ((S1CON & 0x01) == 0);
  c = S1BUF;
  S1CON = S1CON & 0xFE;
  return (c);
}
```

Um die geänderten Funktionen in das Anwenderprogramm einbinden zu können, müssen sie mit dem Compiler compiliert werden. Die daraus erzeugten Object files müssen beim Linken des Anwenderprogramms zusätzlich angegeben werden.
Die Link Anweisung erweitert sich folgendermaßen:
L51 <eigene object file Liste>, GETKEY.OBJ, PUTCHAR.OBJ <controls>

1.2 Die seriellen Schnittstellen des 80C517

1.2.1 Serielle Schnittstelle 0

Die serielle Schnittstelle 0 arbeitet in vier Betriebsarten. Das dazugehörige Steuerregister ist S0CON.

S0CON (98H), bitadressierbar

MSB LSB

SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
9FH	9EH	9DH	9CH	9BH	9AH	99H	98H

Steuerregister der seriellen Schnittstelle 0 (Serial Interface 0 Control). Es enthält Steuerbits für die serielle Schnittstelle 0.
Reset-Wert: 00H

Bitsymbol	Funktion
SM0 SM1	Serieller Modus Bit 0/1
0 0	Modus 0: Schieberegister-Modus
0 1	Modus 1: 8-Bit-UART, flexible Baudrate
1 0	Modus 2: 9-Bit-UART, feste Baudrate
1 1	Modus 3: 9-Bit-UART, flexible Baudrate
SM20	Aktiviert den Multiprozessor-Kommunikationsbetrieb in Modus 2 und 3. Mit SM20 = 1 in Modus 2 und 3 wird RI0 nicht gesetzt, wenn das 9. empfangene Datenbit 0 ist. Mit SM20 = 1 in Modus 1 wird RI0 nicht gesetzt, wenn kein gültiges Stoppbit empfangen wurde. In Modus 0 muß SM20 immer 0 sein.
REN0	Empfängeraktivierung (Receive Enable). wenn REN0 auf 1 gesetzt ist, ist serieller Empfang möglich, mit REN0 = 0 ist serieller Empfang abgeschaltet. Muß durch Software gesetzt/rückgesetzt werden.
TB80	Sendebit 8 (Transmitter Bit 8). Dies ist das 9. Datenbit, das in Modus 2 und 3 ausgesendet wird. Muß durch Software gesetzt/rückgesetzt werden.
RB80	Empfangsbit 8 (Receiver Bit 8). Dies ist das 9. Datenbit, das in Modus 2 und 3 empfangen wird. In Modus 1 (mit SM20 = 0) ist RB80 das empfangene Stoppbit. In Modus 0 wird RB80 nicht benutzt.
TI0	Sende-Interrupt (Transmitter Interrupt). Dies ist das Interrupt-Request-Flag für den Sender. TI0 wird von derH ardware in Modus 0 am Ende des 8. Datenbits gesetzt, in den anderen Modi bei Beginn des Stoppbits. TI0 muß durch Software rückgesetzt werden.
RI0	Empfänger-Interrupt (Receiver Interrupt). Dies ist das Interrupt-Request-Flag für den Empfänger. RI0 wird in Modus 0 von der Hardware am Ende des 8. Datenbits gesetzt, in den anderen Modi während des Stoppbits. RI0 muß durch Software rückgesetzt werden.

Bild 8-2: Special-Function-Register S0CON (98H)

PCON (87H), nicht bitadressierbar

MSB							LSB
SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE

Steuerregister für die Stromaufnahme (Power Control Register). Es enthält Steuerbits für die Stromaufnahme und die serielle Schnittstelle 0.
Reset-Wert: 00H

Bitsymbol	Funktion
SMOD	Bei gesetztem Bit SMOD verdoppelt sich die Baudrate der seriellen Schnittstelle 0 in den Modi 1,2 und 3

Bild 8-3: Special-Function-Register PCON (87H)

ADCON0 (D8H), bitadressierbar

MSB							LSB
BD	CLH	ADEX	BSY	ADM	MX2	MX1	MX0
DFH	DEH	DDH	DCH	DBH	DAH	D9H	D8H

A/D-Wandler-Steuerregister 0 (A/D Converter Control Register 0). Es enthält Steuerbits für den A/D-Wandler und die serielle Schnittstelle 0.
Reset-Wert: 00H.

Bitsymbol	Funktion
BD	Freigabebit für den Baudratengenerator der seriellen Schnittstelle 0. Wenn es gesetzt ist, wird die Baudrate in den Modi 1 und 3 vom Baudratengenerator der seriellen Schnittstelle 0 abgeleitet. Damit können die Standardbaudraten 4,8 KBaud bzw. 9,6 KBaud bei 12 MHz Oszillatorfrequenz erzeugt werden.

Bild 8-4: Special-Function-Register ADCON0 (D8H)

In Bild 8-7 sind einige repräsentative Betriebsfälle bei Baudratenerzeugung mit Timer 1 herausgegriffen.

Baudrate in Modus 1 u. 3	f_{osz} [MHz]	SMOD	Timer 1		
			C/T#	Modus	Reload
62,5 KBaud	12,000	1	0	2	FFH
19,2 KBaud	11,059	1	0	2	FDH
9,6 KBaud	11,059	0	0	2	FDH
4,8 KBaud	11,059	0	0	2	FAH
2,4 KBaud	11,059	0	0	2	F4H
1,2 KBaud	11,059	0	0	2	E8H
110 Baud	6,000	0	0	2	72H
110 Baud	12,000	0	0	1	FEEBH

Bild 8-7: Baudratenerzeugung mit Timer 1

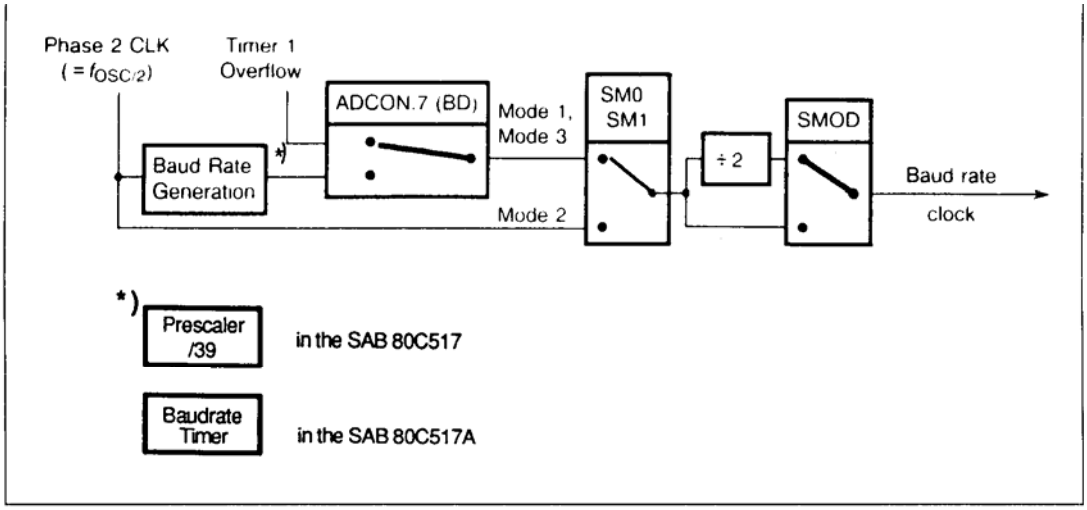


Bild 8-8: Baudratenerzeugung für die serielle Schnittstelle 0

Baudratentakt abgeleitet von	Modus	Baudrate
Timer 1 in Modus 1 (siehe Bild 7-7)	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{1}{16} *$ (Timer 1-Überlaufrate)
Timer 1 in Modus 2	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{1}{16} * \frac{f_{osz}}{12 * (256 - (TH1))}$
Oszillator	2	$\frac{2^{SMOD}}{2} * \frac{1}{16} * \frac{f_{osz}}{2}$
Baudratengenerator im 80C517	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{f_{osz}}{1250}$
Baudratengenerator im 80C517A	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{1}{32} * \frac{f_{osz}}{2^{10} - S0REL}$ (mit S0REL = S0RELH.1-0, S0RELL.7-0)

Bild 8-9: Formeln für Baudraten für die serielle Schnittstelle 0

1.2.2 Serielle Schnittstelle 1:

Die serielle Schnittstelle 1 hat lediglich asynchrone Betriebsarten und kann ähnlich wie die serielle Schnittstelle 0 in den Modi 8- oder 9-bit-Datenübertragung arbeiten. Das dazugehörige SFR ist S1CON.

S1CON (9BH), nicht bitadressierbar

MSB							LSB
SM	-	**SM21**	**REN1**	**TB81**	**RB80**	**TI1**	**RI1**

Steuerregister der seriellen Schnittstelle 1 (Serial Interface 1 Control). Es enthält Steuerbits für die serielle Schnittstelle 1.
Reset-Wert: 0x00 0000B

Bitsymbol	Funktion
SM	SM = 0: serielle Betriebsart A; 9-Bit UART SM = 1: serielle Betriebsart B; 8-Bit UART
SM21	Aktiviert den Multiprozessor-Kommunikation-Betrieb in Modus A. Mit SM21 = 1 wird RI1 nicht gesetzt, wenn das 9. empfangene Datenbit 0 ist. Mit SM21 = 1 in Modus B wird RI1 nicht gesetzt, wenn kein gültiges Stoppbit empfangen wurde.
REN1	Receive Enable; Enable für den Empfänger; wenn auf 1 gesetzt, ist serieller Empfang möglich, mit REN1 = 0 ist serieller Empfang abgeschaltet. Es muß durch Software gesetzt/rückgesetzt werden.
TB81	Transmitter Bit 8; Dies ist das 9. Datenbit, das in Modus A ausgesendet wird. Es muß durch Software gesetzt/rückgesetzt werden.
RB81	Receiver Bit 8; Dies ist das 9. Datenbit, das in Modus A empfangen wird. In Modus B (mit SM21 = 0) ist RB81 das empfangene Stoppbit.
TI1	Transmitter Interrupt; Dies ist das Interrupt-Request-Flag für den Sender. TI1 wird von der Hardware bei Beginn des Stoppbits gesetzt. TI1 muß durch Software rückgesetzt werden.
RI1	Receiver Interrupt; Dies ist das Interrupt-Request-Flag für den Empfänger. TI1 wird während des Stoppbits gesetzt. RI1 muß durch Software rückgesetzt werden.

Bild 8-10: Special-Function-Register S1CON (9BH)

S1BUF (9CH), nicht bitadressierbar

MSB							LSB
S1BUF.7	**S1BUF.6**	**S1BUF.5**	**S1BUF.4**	**S1BUF.3**	**S1BUF.2**	**S1BUF.1**	**S1BUF.0**

Empfangs- und Sendepuffer der seriellen Schnittstelle 1 (Serial Interface 1 Buffer). Ein Schreibbefehl auf S1BUF lädt den Sendepuffer und startet eine Übertragung. Ein Lesebefehl auf S1BUF greift auf das physikalisch getrennte Empfangsregister zu.
Reset-Wert: XXH

Bild 8-11: Special-Function-Register S1BUF (9CH)

Baudraten:

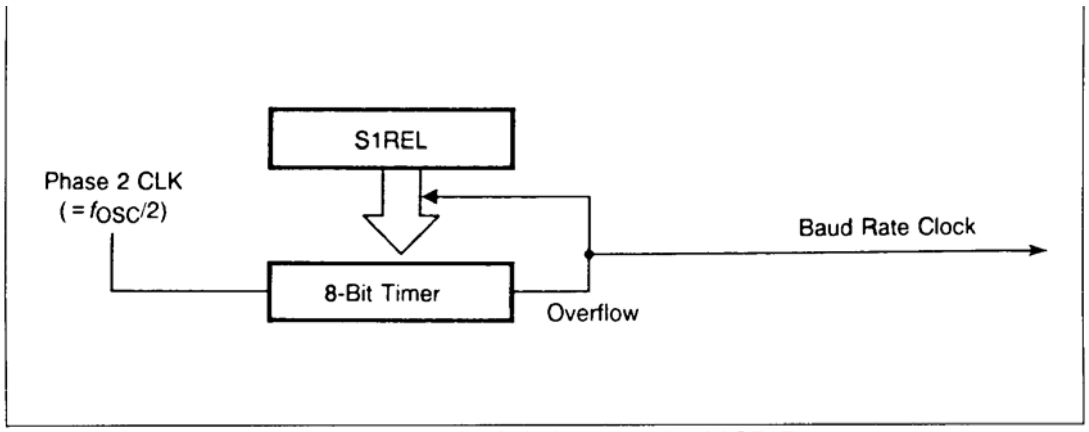


Bild 8-12: Baudratengenerator für serielle Schnittstelle 1 im 80C517

S1REL (9DH), nicht bitadressierbar

MSB							LSB
S1REL.7	**S1REL.6**	**S1REL.5**	**S1REL.4**	**S1REL.3**	**S1REL.2**	**S1REL.1**	**S1REL.0**

Reload-Register (8 bit) für den Baudratengenerator der seriellen Schnittstelle 1 im 80C517.
Nach Reset ist die Baudrate 1,5 KBaud bei 12 MHz Oszillatorfrequenz.
Reset-Wert: 00H

$$\text{Baudrate} = \frac{f_{osz}}{32 * \left(256 - \left(\text{S1REL}\right)\right)}$$

1.3 Die Funktionen printf und scanf

Die Funktion ***int printf (const char *,...)*** formatiert eine Reihe von Zeichen (Strings) und Zahlenwerten und gibt diese über die Funktion ***Putchar*** aus. Der zurückgegebene Wert ist die Anzahl der ausgegebenen Zeichen.
Eine Formatangabe beginnt immer mit einem Prozent Zeichen. Jedes Feld einer Formatangabe besteht aus einem oder mehreren Zeichen oder einer Zahl. Eine Auflistung der einzelnen Formattypen finden Sie in den begelegten Blättern der Run Time Library.

Die Funktion int scanf (const char *, ...) zerlegt die Eingabe in Argumente. Der Eingabewert wird über die Funktion getchar eingelesen. Der Rückgabewert ist die Anzahl der eingelesenen Argumente.
Eine genaue Beschreibung finden Sie in den beigelegten Blättern.

Übung 8: Schreiben Sie ein Programm, welches Character, Integerzahlen, Floatzahlen und Strings auf der seriellen Schnittstelle ausgibt. Lesen Sie mit scanf von der seriellen Schnittstelle ein und geben Sie den String wieder aus.

Verwenden Sie dazu das auf der Festplatte gespeicherte File UEBUNG8.C
Eine mögliche Lösung ist auf der nächsten Seite dargestellt!

```
/*******************************************************/
/*                      UEBUNG8.C                      */
/*     Ein- und Ausgeben ueber ser. Schnittstelle      */
/*******************************************************/

/****   Steueranweisungen an den Compiler       ****/
/****   Angabe der Include Dateien              ****/
#include <reg517a.h>
#include <stdio.h>

/****   Prototypen der Funktionen               ****/
void ini_ser1(void);

/****   Funktionen                              ****/
void ini_ser1(void)
{
        S0CON = 0x5E;   /* Serieller Mode 1, 8bit, 1 Stopbit */
        BD = 1;   /* Baudrategenerator aktiv */
        PCON = 0x80;    /* SMOD = 1; 9600baud,12MHz */
}

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

main()
{
char puffer[30];
char c;
int a,d;
float b;

c = 'X';
a = -1023;
b = 0.123456789;

ini_ser1();

printf ("Ausgabe ueber ser. Schnittstelle\n\n");
printf ("Character: %c\n", c);
printf ("Integer: %d\n", a);
printf ("Unsigned Integer: %u\n",a);
printf ("Integer hexadezimal: %x\n", a);
printf ("Float: %1.1f\t%1.3f\n\n", b,b);

printf ("Geben Sie Zeichen ein, nach Return oder Blank\n");
printf ("wird der String wieder ausgeben.\n\n");

scanf("%s",&puffer);
printf("%s\n",&puffer);
}
```

1.4 Programmbeispiel - RECHNER

Es soll ein kleiner Taschenrechner mit Grundrechnungsfunktionen programmiert werden, der seine Daten über die RS232E-Schnittstelle erhält, die Rechenaperation durchführt und das Ergebnis wieder über die Schnittstelle zurückgibt.

 Hinweis

Das PC-Programm mon51.exe sendet peridisch ein Ctrl+Q (=0x11H) um festzustellen, in welchem Zustand sich das Monitorprogramm auf dem Board befindet. Es ist nun das Original File **getkey.c** so zu ändern, daß dieses Zeichen ignoriert wird.

```
/*--------------------------------------------------------------------------*/
/*  This file is part of the C-51 Compiler package                          */
/*  Copyright KEIL ELEKTRONIK GmbH 1993                                     */
/*--------------------------------------------------------------------------*/
/*                                                                          */
/*  GETKEY.C:  This routine is the general character input of C-51.         */
/*                                                                          */
/*  To translate this file use C51 with the following invocation:           */
/*                                                                          */
/*     C51 GETKEY.C  <memory model>                                         */
/*                                                                          */
/*  To link the modified GETKEY.OBJ file to your application use the        */
/*  following L51 invocation:                                               */
/*                                                                          */
/*     L51 <your object file list>, GETKEY.OBJ <controls>                   */
/*                                                                          */
/*--------------------------------------------------------------------------*/

#include <reg51.h>

char _getkey ()  {
  char c;
do
{
  while (!RI);
  c = SBUF;
  RI = 0;
}
while(c==0x11);
  return (c);
}
```

Programmbeispiel rechner.c

```
/* ************************************************************ */
/* ****** Datei:    Rechner.C ********************************** */
/* ****** Datum:    10.05.1995     ***************************** */
/* ****** Compiler: Keil - C 51 Version 4.0 ******************** */
/* ************************************************************ */

#include <reg517.h>
#include <stdio.h>

/*Variablendefinitionen*/
float zahl1,zahl2,erg;
char op;

/* Hauptprogramm */
main()
{
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 */
```

2 Datenausgabe auf die LCD Anzeige

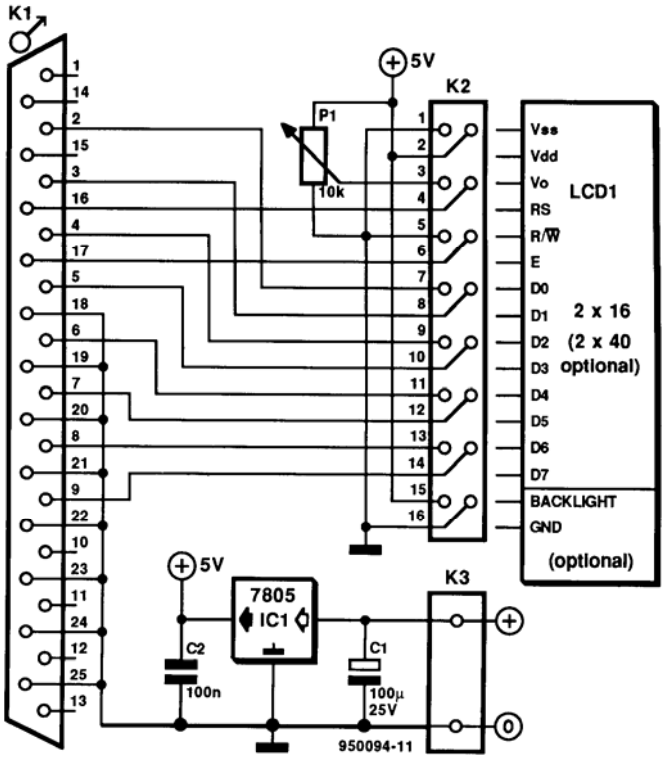
Eine sehr oft verwendete Anzeigeeinheit in Verbindung mit einem Mikrocontroller ist eine LCD Anzeige. Neben normalen Sieben Segment Displays ohne integrierte Ansteuerelektronik sind auch komplette Anzeigemodule erhältlich. Hier unterscheidet man Punkt Matrix LCD Module für Schriftdarstellung und Punkt Matrix LCD Module, mit welchen Grafik dargestellt werden kann.

Punkt Matrix Module für Schriftdarstellung sind in verschiedenen Größen erhältlich. Sie verwenden alle eine Zeichenmatrix mit 5x7 Punkten zur Darstellung der Zeichen. Die auf dem Modul enthaltene Ansteuerelektronik ist meist mit dem Controller/Treiberbaustein HD44780 aufgebaut, welcher einen integrierten Zeichengenerator und ein Display RAM enthält.

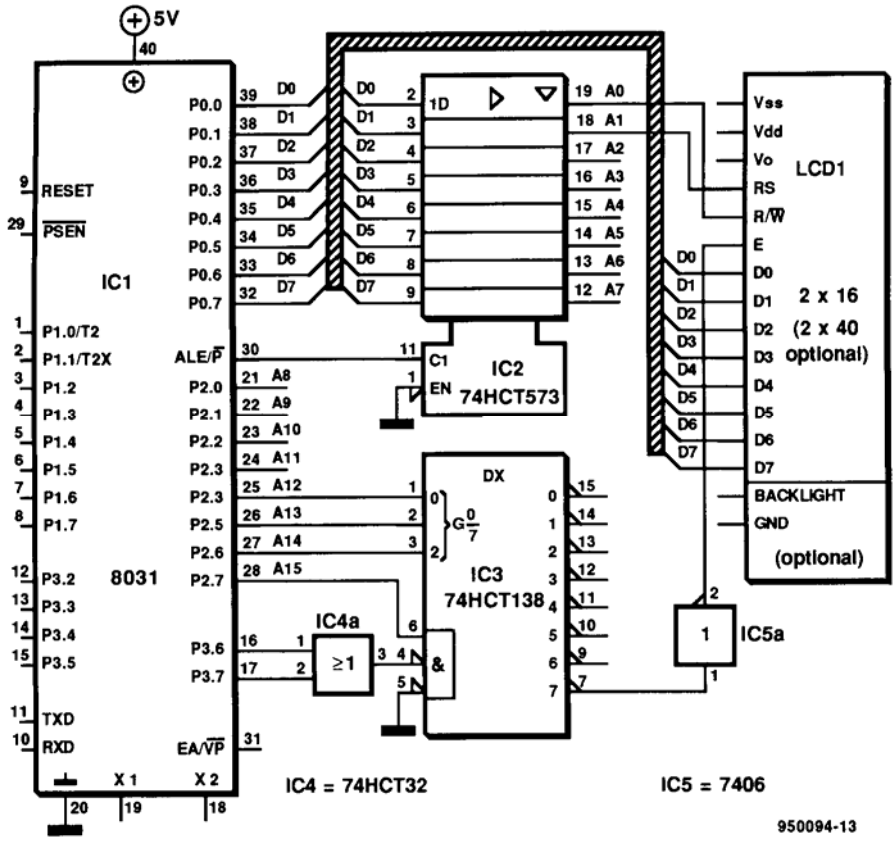
Auf der Übungsplatine ist ein Punkt Matrix Modul mit 2 Zeilen zu je 20 Zeichen aufgebaut. Dieses ist über den Port 6 an den Mikrocontroller 80C517 angeschlossen. Das Modul benötigt drei Steuerleitungen und vier Datenleitungen.

2.1 Allgemeines

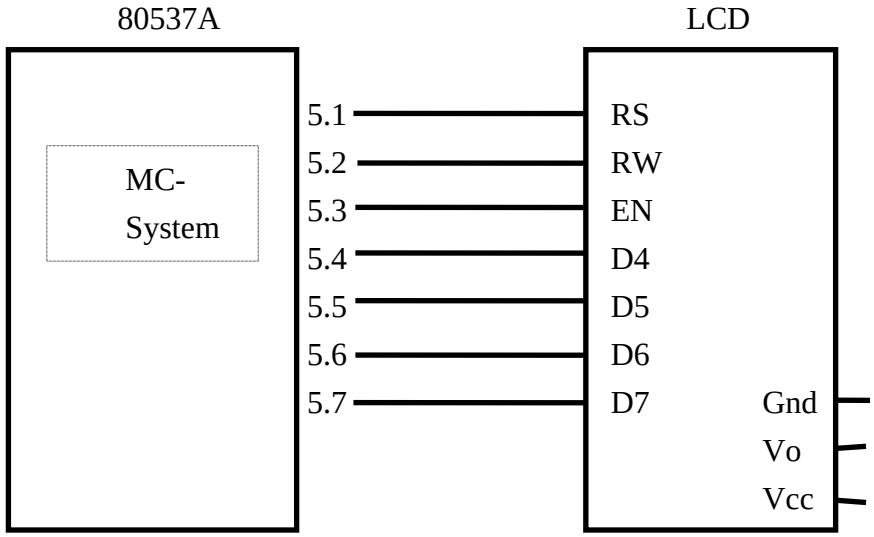
Beispiel für den Anschluß an die Druckerschnittstelle



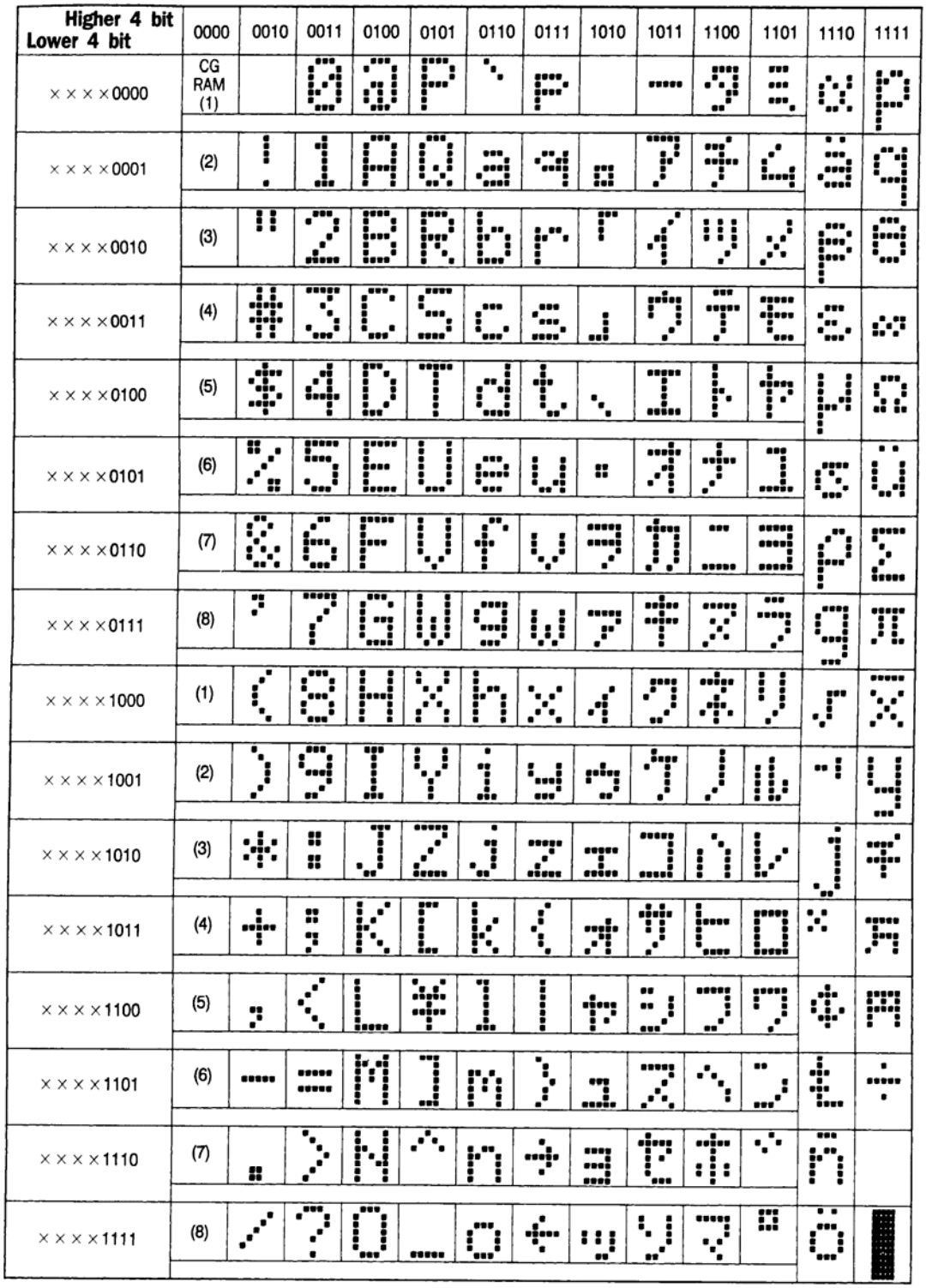
Beispiel für den Anschluß an einen 8031er Adresse F000H:



Anschluß direkt an einen Port:

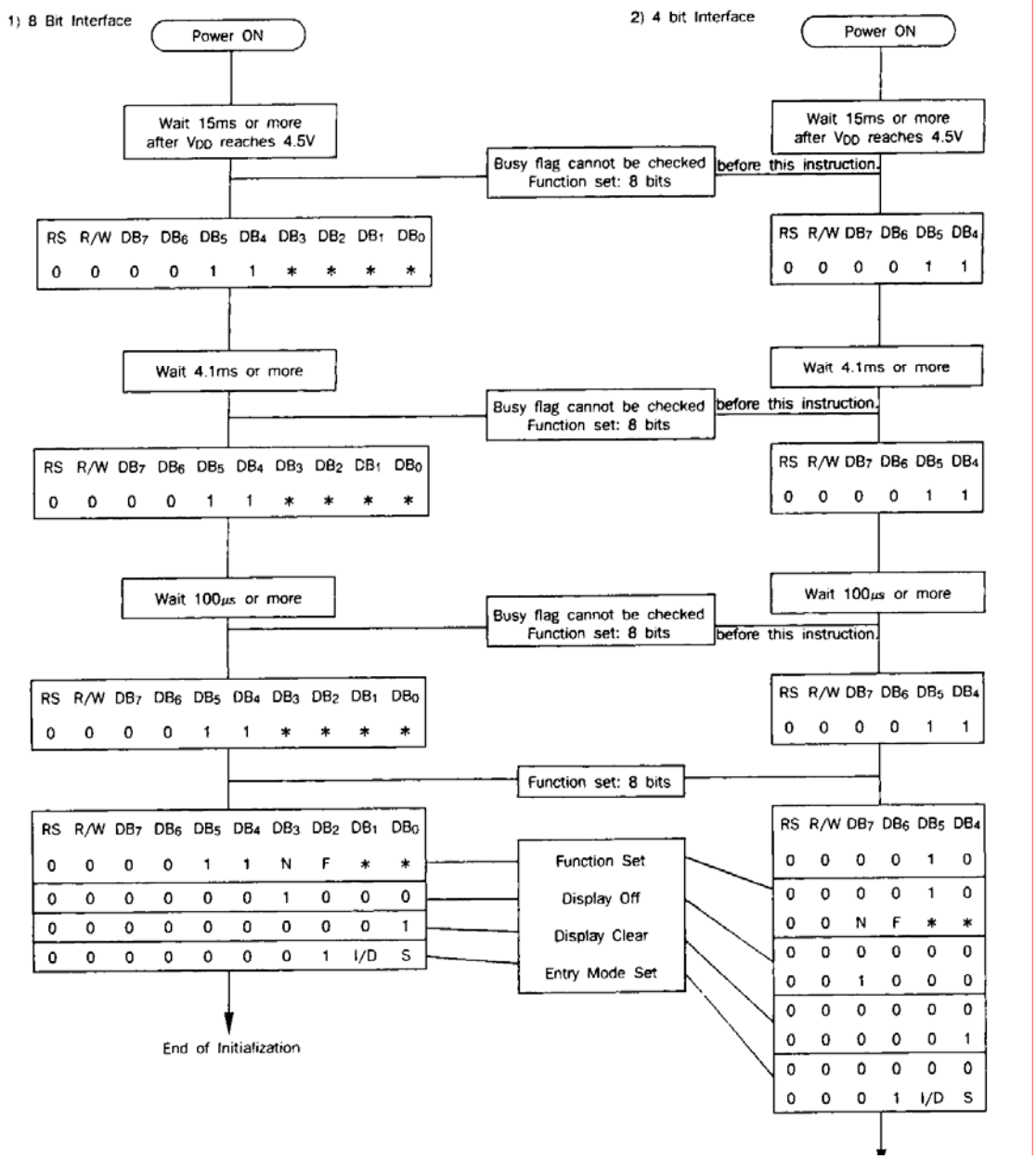


Der im ROM des Controllers verfügbare Zeichensatz. Es können auch noch maximal 8 Zeichen vom Anwender definiert werden.



Elektor 10/95 27

Initialisierungsroutine des Displays im Flußdiagramm. Dieser Prozeß muß nach jedem Einschalten durchlaufen werden, er dauert aber nur weige Millisekunden.



Der Befehlssatz des Hitachi-Displaycontrollers, der in den meisten alphanumerischen, ein- bis zweizeiligen LC-Anzeigen verwendet wird.

Instruction	Code										Description	Execution Time (max) (when fcp or fosc is 250 KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM anddress 0 in address counter.	1.64 ms
Return Home	0	0	0	0	0	0	0	0	1	*	Sets DD RAM address 0 in address counter. Also returns display being shifted to original position. DD RAM contents remain unchanged.	1.64 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display. These operations are performed during data write and read.	40 µs
Display On / Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of entire display (D), Cursor ON/OFF (C), and blink of cursor position character (B).	40 µs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor & shifts display without changing DD RAM contents.	40 µs
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display lines (L) and character fonts (F).	40 µs
Set CG RAM Address	0	0	0	1	ACG						Sets CG RAM address. CG RAM data is sent and received after this setting.	40 µs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. CG RAM data is sent and received after this setting.	40 µs
Read Busy Flag and Address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM	40 µs
Read Data from CG or DD RAM	1	1	Read Data								Reads data into DD RAM or CG RAM	40 µs
	I/D = 1 : Increment I/D = 0 : Decrement S = 1 : Accompanies display shift S/C = 1 : Display shift S/C = 0 : Cursor move R/L = 1 : Shift to the right R/L = 0 : Shift to the left DL = 1 : 8 bits, DL = 0 : 4 bits N = 1 : 2 lines, N = 0 : 1 line F = 1 : 5 × 10 dots, F = 0 : 5 × 7 dots FB = 1 : Internally operating FB = 0 : Can accept instruction										DD RAM : Display data RAM CG RAM : Character generator RAM ACG: CG RAM address ADD: DD RAM Address : Corresponds to cursor address AC: Address counter used for both DD and CG RAM address.	Execution time changes when frequency changes Example: When fcp or fosc is 270 kHz: $40\ \mu s \times \frac{250}{270} = 37\ \mu s$	

*No effect

950094-1-16

2.2 Programmpaket zur Ansteuerung der LCD Anzeige

Auf den Festplatten der Rechner ist das File **LCD.C** gespeichert, welches ein fertiges Paket von C Routinen beinhaltet, die man benötigt, um die LCD Anzeige anzusprechen. Folgende Routinen sind darin enthalten:

- warte (msec) wartet die eingegebene Zahl von Millisekunden
- busy_lcd( ) wartet, während LCD Anzeige beschäftigt ist
- init_lcd( ) initialisiert die LCD Anzeige
- blank_lcd ( ) löscht die LCD Anzeige
- char_lcd(zeile, spalte, ascii)
  Schreibt den Character (das ASCII - Zeichen 'ascii') in die 'zeile' an die entsprechenden 'spalte' des LCD-Moduls
- print_lcd (zeile, spalte, string)
  Schreibt einen String in das Display nach 'zeile', beginnend in 'spalte'.
  Beispiele für einen Aufruf:
  print_lcd (1, 1, puffer); schreibt Zeichen, welche im Feld 'puffer' gespeichert sind, in Zeile 1, ab Spalte 1.
  print_lcd (2, 5, "Hallo"); schreibt 'Hallo' in Zeile 2, ab Spalte 5.
- graf_lcd (adresse, muster)
  Ermöglicht es, 8 Sonderzeichen zu definieren
- spezial_graf ( ) Definierte Sonderzeichen
- def_balken ( ) Definierte Sonderzeichen

2.3 Einbindung des Programmpakets

Um die im File **LCD.C** enthaltenen Funktionen im Anwenderprogramm nutzen zu können, sind folgende Schritte nötig :
Neben dem File **LCD.C** ist auch ein File **LCD.H** auf der Festplatte gespeichert. In diesem File sind die Prototypen aller Funktionen abgespeichert. Dieses File muß im Anwenderprogramm angegeben werden, damit der Compiler beim compilieren weiß,

das diese Funktionen in einem anderen File (**LCD.C**) abgespeichert sind und beim Linkvorgang dazugelinkt werden.
Die im Programm nötige Anweisung lautet: **#include "lcd.h"**
Im Unterschied zu Standard Bibliotheken wird bei *include* Anweisungen für Headerfiles, welche sich im gleichen Verzeichnis befinden wie das Hauptprogramm, der Name des Files unter " " gesetzt. (Standard Bibliothek: **#include <stdio.h>**)

Das File **LCD.C** muß mit dem C51 Compiler compiliert werden. Der Compiler erzeugt ein File **LCD.OBJ** .

Das Hauptprogramm wird ebenfalls compiliert und ein Object File erzeugt.

Mit dem Linker werden alle Files gemeinsam gelinkt. Der Aufruf des Linkers lautet daher:

L51 <>.OBJ,LCD.OBJ,START.OBJ CODE (8000H) XDATA (0C000H) RAMSIZE (256)

Der weitere Ablauf ist wie gehabt. Mit dem Programm OHS51 wird ein Intel Hex File erzeugt und mit dem Monitorprogramm MON51 geladen und getestet.

Source-Code LCD.C:

```
/*****************************************************************************/
/*      Verschiedene Routinen zur Ansteuerung eines Sharp LCD Displays       */
/*                vom Typ LM20A21 (2 Zeilen, 20 Spalten)                     */
/*****************************************************************************/
#include <reg517.h>
#include "lcd.h"
#define SPALTEN 20
#define ZEILEN 2
/* LCD-Anschluesse an Ports */
sbit RS = P5^0; sbit RW = P5^1; sbit EN = P5^2; sbit D4 = P5^3; sbit D5 = P5^4;
sbit D6 = P5^5; sbit D7 = P5^6;
/* Wartet eine entsprechende Anzahl von Millisekunden */
void warte(unsigned int msec)
{ data char i;
  for (msec; msec!=0; msec--) for (i=0; i<83; i++);
}
/* Wartet, bis LCD neue Daten aufnehmen kann */
void busy_lcd(void)
{ data bit busy;
  busy =1;
  while (busy == 1) {
                       D7=1; D6=1; D5=1; D4=1;      /*Ports auf Eingabe*/
                       RS=0; RW=1;
                       EN=1; busy=D7; EN=0;         /*lese 1.Halbbyte  */
                                                    /*busy-flag in D7  */
                       EN=1; EN=0;                  /*lese 2.Halbbyte  */
                     }
}
/*initialisiert den LCD-Controller auf 4 Bit Betrieb etc */
void init_lcd(void)
{
```

```
  D7=1; D6=1; D5=1; D4=1;
  RS=0; RW=0; EN=0;
  warte (20);
  D7=0; D6=0; D5=1; D4=1; EN=1; EN=0;   /*func set: interface is 8 bit long*/
  warte (5);
  D7=0; D6=0; D5=1; D4=1; EN=1; EN=0;   /*func set: interface is 8 bit long*/
  warte (1);
  D7=0; D6=0; D5=1; D4=1; EN=1; EN=0;   /*func set: interface is 8 bit long*/
  warte (1);
  D7=0; D6=0; D5=1; D4=0; EN=1; EN=0;   /*func set: interface is 4 bit long*/
  warte (1);
  D7=0; D6=0; D5=1; D4=0; EN=1; EN=0;   /*func set: interface is 4 bit long*/
  D7=1; D6=0; D5=0; D4=0; EN=1; EN=0;   /* number of display lines is 2    */
  busy_lcd ();
  RS=0; RW=0;
  D7=0; D6=0; D5=0; D4=0; EN=1; EN=0;    /* D6=1 display on; */
  D7=1; D6=1; D5=0; D4=0; EN=1; EN=0;
  busy_lcd ();
  RS=0; RW=0;
  D7=0; D6=0; D5=0; D4=0; EN=1; EN=0;    /* clear display */
  D7=0; D6=0; D5=0; D4=1; EN=1; EN=0;
  busy_lcd ();
  RS=0; RW=0;
  D7=0; D6=0; D5=0; D4=0; EN=1; EN=0;    /* entry mode: decrement, no shift */
  D7=0; D6=1; D5=0; D4=0; EN=1; EN=0;
}
void blank_lcd(void)            /*loescht die Anzeige und stellt den Cursor zur ck*/
{ busy_lcd ();
  RS=0; RW=0;
  D7=0; D6=0; D5=0; D4=0; EN=1; EN=0;                    /* clear display */
  D7=0; D6=0; D5=0; D4=1; EN=1; EN=0;
  busy_lcd ();
  RS=0; RW=0;
  D7=0; D6=0; D5=0; D4=0; EN=1; EN=0;                    /* return home */
  D7=0; D6=0; D5=1; D4=0; EN=1; EN=0;
}
void char_lcd(char zeile, char spalte, char ascii)
{ data char adresse;
  if (zeile>0 && zeile<(ZEILEN+1))
     if (spalte>0 && spalte<(SPALTEN+1))
               { adresse = spalte-1 + (zeile-1)*64;
                 busy_lcd(); RS=0; RW=0;
                 D7 = 1;
                 D6 = adresse & 64;
                 D5 = adresse & 32;
                 D4 = adresse & 16;
                 EN=1; EN=0;
                 D7 = adresse & 8;
                 D6 = adresse & 4;
                 D5 = adresse & 2;
                 D4 = adresse & 1;
                 EN=1; EN=0;
                 busy_lcd(); RS=1; RW=0;
                 D7 = ascii & 128;
                 D6 = ascii & 64;
                 D5 = ascii & 32;
                 D4 = ascii & 16;
                 EN=1; EN=0;
                 D7 = ascii & 8;
                 D6 = ascii & 4;
                 D5 = ascii & 2;
                 D4 = ascii & 1;
                 EN=1; EN=0;
                }
}
void print_lcd(char zeile, char spalte, char *str)
   {
      char i;
      for ( i = spalte; (i <= 20) &&  *str; i++)
         {
            warte(10);
            char_lcd(zeile,i,*str);
            str++;
         };
```

```
    }
void graf_lcd (char adresse, char muster)
{
        if (adresse <= 64) {              /* Gueltiger CG-Bereich ? */
                busy_lcd (); RS=0; RW=0;  /* CG-RAM-Adresse 0..64 */
              D7 = 0;
/* oberes Halbbyte adresse */
                D6 = 1;
                D5 = adresse & 32;
                D4 = adresse & 16;
                EN=1; EN=0;
                D7 = adresse & 8;
/* unteres Halbbyte adresse */
                D6 = adresse & 4;
                D5 = adresse & 2;
                D4 = adresse & 1;
                EN=1; EN=0;
                busy_lcd (); RS=1; RW=0;  /* muster an CG-RAM */
                D7 = muster & 128;                                          /*
oberes Halbbyte muster */
                D6 = muster & 64;
                D5 = muster & 32;
                D4 = muster & 16;
                EN=1; EN=0;
                D7 = muster & 8;                                            /*
unteres Halbbyte muster */
                D6 = muster & 4;
                D5 = muster & 2;
                D4 = muster & 1;
                EN=1; EN=0;
        }
}
/*********************************************
*  Define special grafic symbols              *
*  def_graf()                                 *
*********************************************/
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();
}

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();

}

/*************************************************************
* Ergänzung: Durch "Umleiten" der Ausgabe in der Funktion
* putchar() von der seriellen Schnittstelle auf die Funktion
* lcd_write() kann mit der Funktion printf() direkt auf die
* Dot-Matrix geschrieben werden. Dazu müssen noch die Variablen
* x,y und lcd_ser global definiert werden.
*************************************************************/

extern unsigned char x,y;
extern bit lcd_ser;

/*************************************************************
* pos(zeile,spalte) bestimmt die Anfangskoordinaten auf der
* Dot-Matrix, von denen aus die Zeichen über printf()
* geschrieben werden.
*************************************************************/


void pos(char zeile,char spalte)
{
x = spalte;
y = zeile;
```

```
}

/************************************************************
* Wenn das bit lcd_ser auf 1 gesetzt ist, so schreibt putchar()
* ein Zeichen auf die Dot-Matrix.
************************************************************/

void lcd_write(char c) {
char_lcd(y,x,c);
x++;
if(x>20) {
        x=1;
        y++;
        }
if (y>2) y=1;
}
```

2.3.1 Verwendung des Programmpakets

Um die LCD Anzeige richtig anzusprechen, sind im Programm folgende Aufrufe nötig: Am Beginn des Hauptprogramms muß die LCD Anzeige initialisiert werden. Dazu muß die Funktion ***init_lcd ( )*** aufgerufen werden. Im Anschluß sollte die LCD Anzeige mit der Funktion ***blank_lcd ( )*** gelöscht werden.

Strings können mit der funktion ***print_lcd (1, 1, "Hallo !! ");*** ausgegeben werden. Sollen mit dieser Funktion Variablen ausgegeben werden, so ist ein Pufferspeicher einzurichten, aus dem auf die LCD Anzeige geschrieben werden kann.(z.B. xdata char puffer[20];). Dies ist deshalb nötig, weil die LCD Anzeige nur mit Charactern beschrieben werden kann. Daher müssen Variablen mit der funktion ***sprintf (puffer, "%d", taste);*** zuerst in einen String aus Charactern umgewandelt werden. Dieser String wird dann von der Funktion ***print_lcd (1,1,puffer);*** auf die LCD Anzeige ausgegeben.

Übung 9: Schreiben Sie ein kurzes Programm, das eine kurze Meldung auf die erste Zeile der LCD Anzeige ausgibt.
Anschließend sollen Sie das Programm erweitern, indem Sie die Sonderzeichen mit der Funktion spezial_graf( ) in die LCD Anzeige laden und anschließend mit der Funktion char_lcd (...) in der zweiten Zeile nebeneinander darstellen.

Übung 10: Schreiben Sie ein Programm, mit dem Sie verschiedene Variablentypen ausgeben können. Verwenden Sie dazu die Funktion sprintf(...) in Verbindung mit print_lcd(....)

2.3.2 Musterlösung Übung 9

```
/*********************************************************/
/*                      UEBUNG9.C                        */
/*              Ausgeben an die LCD Anzeige              */
/*********************************************************/

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

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

/* globale Variablen */
unsigned char i;

main()
{
                        /* Initialisierung */
        init_lcd();
        blank_lcd();

        /* Ausgabe eines Strings auf der LCD*/
        print_lcd(1,1,"Hallo Du");
        print_lcd(1,10,"SoSo");

        /* Laden von Sonderzeichen */
        spezial_graf();

        /* Ausgabe der Sonderzeichen */
        for (i = 0; i <8; i++)char_lcd(2,i+1,i);

        while(1);

}
```

2.3.3 Musterlösung zu Übung 10

```
/*********************************************************/
/*                      UEBUNG9.C                        */
/*              Ausgeben an die LCD Anzeige              */
/*********************************************************/

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

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


/* globale Variable */

main()
{
int i;
char text[21];
float x=3.1234567;

init_lcd();
blank_lcd();

  for (i=0;i<=51;i++)
  {
    print_lcd(1,3,"**** Test1 ****");
    sprintf(text,"Schleife: ");
    print_lcd(2,1,text);
    printf("Schleife %2d durchlaufen\n",i);
    P4 = i;
    warte(500);
    sprintf(text,"%2d",i+1);
    print_lcd(2,13,text);
    warte(1000);
    blank_lcd();
    print_lcd(1,1,"Float 1.4:");
    sprintf(text,"%1.4f",x);
    print_lcd(1,13,text);
    print_lcd(2,1,"Character: ");
    char_lcd(2,14,('A'+i));
    warte(1000);
    blank_lcd();
  }
}
```

2.3.4 Musterlösung Übung 11

```
/*******************************************************/
/*                      UEB11.C                        */
/*              Ausgeben an die LCD Anzeige            */
/*******************************************************/
/****   Steueranweisungen an den Compiler       ****/
#pragma iv(0x8000) mod517 code debug pl(61)
/****   Angabe der Include Dateien              ****/
#include <reg517.h>
#include <stdio.h>
#include "lcd.h"
unsigned char msec;
char text[21],k;
bit secflag;
unsigned char i,z,s;
void INTTIM0(void) interrupt 1
{
  TL0 = 0x00;
  TH0 = -39;
  msec++;
  if (msec==100)
    { msec=0; secflag=1; }
}
main()
{
z=1;
s=1;
/* Initialisierung */
init_lcd();
blank_lcd();
TMOD=0x01;
EAL=1;
TR0=1;
ET0=1;
/* Laden von Sonderzeichen */
def_balken();
while(1)
{
    while(!secflag);
  {
    secflag = 0;
    i++;k++;
    if(i==6)
      {i=1;  z++; }
    char_lcd(2,z,i);
    sprintf(text,"Counter = %3d%%   ",(int)k);
    print_lcd(1,1,text);
    if (k==100)
      {i=0; z=1; k=0; blank_lcd();}
   }P4 = z;
}}
```

2.3.5 Übung Tasten - LCD

```
/*******************************************************/
/*                      UEBUNG7a.C                     */
/*      Einlesen der Tastatur ueber einen Interrupt    */
/*******************************************************/

/****   Steueranweisungen an den Compiler       ****/
#pragma iv(0x8000)

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

/****   Globale Variablen                       ****/
code char tasten[23]={0,0,2,0,1,0,3,0,8,0,10,0,9,0,11,0,4,0,6,0,5,0,7};
char taste,text[21];
bit gedrueckt;

void TAST (void) interrupt 0
{
  taste = tasten[P7>>3];
  gedrueckt=1;
}

main()
{

/*      Anweisungen zur Behandlung des Interrupts    */

EX0 = 1;        /**** Externer Int0               ****/
EAL = 1;        /**** Interrupt allgemein erlaubt ****/
IT0 = 1;        /* neg. Flanke */
P4 = 0x00;
init_lcd();
blank_lcd();

while (1)
  {
  if(gedrueckt)
    {
    P4=taste;
    sprintf(text,"Taste %2bd gedrueckt\n",taste);
    print_lcd(1,1,text);
    gedrueckt=0;
    } /* endif */
  } /* endwhile */
}
```

Anhang: putlcd.c

```
/*****************************************************************/
/*  AENDERUNG: SERIELLE AUSGABE AUF SCHNITTSTELLE 1 UMGELEITET   */
/*  PUTLCD.C: Ausgabe auf Serielle Schnittstelle 1 oder auf LCD  */
/*  lcd_ser = 1 ==> LCD                                          */
/*  lcd_ser = 0 ==> Ser. Schnittstelle 1                         */
/*  To translate this file use C51 with the following invocation:*/
/*     C51 PUTLCD.C                                              */
/*  To link the modified PUTCHAR.OBJ file to your application use the */
/*  following L51 invocation:                                    */
/*     L51 <your object file list>, PUTLCD.OBJ <controls>        */
/*****************************************************************/
#include <reg517.h>
#define XON  0x11
#define XOFF 0x13
/* Das Bit lcd_ser dient zur Umschaltung LCD / Serielle   */
/* Wird als "extern" definiert, muß im Hauptprogramm ohne */
/* "extern" angegeben werden                              */
extern bit lcd_ser;
/* Prototyp der Ausgabefunktion */
void lcd_write(char c);
char putchar (char c) {
if (lcd_ser)
  {
  lcd_write(c);
  return(c);
  }
else
  {
  if (c == '\n')  {
    if ((S1CON & 0x01) == 1)  {
      if (S1BUF == XOFF)  {
        do  {
          S1CON = S1CON & 0xFE;
          while ((S1CON & 0x01) == 0);
        }
        while (S1BUF != XON);
        S1CON = S1CON & 0xFE;
      }
    }
    while ((S1CON & 0x02) == 0);
    S1CON = S1CON & 0xFD;
    S1BUF = 0x0d;                        /* output CR  */
  }
  if ((S1CON & 0x01) == 1)  {
    if (S1BUF == XOFF)  {
      do  {
          S1CON = S1CON & 0xFE;
        while ((S1CON & 0x01) == 0);
      }
      while (S1BUF != XON);
      S1CON = S1CON & 0xFE;
```

```
    }
  }
  while ((S1CON & 0x02) == 0);
  S1CON = S1CON & 0xFD;
  return (S1BUF = c);
  }
}
```

Anhang putchar.c

```
/*****************************************************************************/
/*  This file is part of the C-51 Compiler package                           */
/*  Copyright KEIL ELEKTRONIK GmbH 1990                                      */
/*****************************************************************************/
/*                                                                           */
/*---------------------------------------------------------------------------*/
/*  AENDERUNG: SERIELLE AUSGABE AUF SCHNITTSTELLE 1 UMGELEITET               */
/*---------------------------------------------------------------------------*/
/*                                                                           */
/*  PUTCHAR.C:  This routine is the general character output of C-51.        */
/*                                                                           */
/*  To translate this file use C51 with the following invocation:            */
/*                                                                           */
/*     C51 PUTCHAR.C                                                         */
/*                                                                           */
/*  To link the modified PUTCHAR.OBJ file to your application use the        */
/*  following L51 invocation:                                                */
/*                                                                           */
/*     L51 <your object file list>, PUTCHAR.OBJ <controls>                   */
/*                                                                           */
/*****************************************************************************/

#include <reg517.h>

#define XON  0x11
#define XOFF 0x13

char putchar (char c)  {

  if (c == '\n')  {
    if ((S1CON & 0x01) == 1)  {
      if (S1BUF == XOFF)  {
        do  {
          S1CON = S1CON & 0xFE;
          while ((S1CON & 0x01) == 0);
        }
        while (S1BUF != XON);
        S1CON = S1CON & 0xFE;
      }
    }
    while ((S1CON & 0x02) == 0);
    S1CON = S1CON & 0xFD;
```

```
      S1BUF = 0x0d;                          /* output CR  */
    }
    if ((S1CON & 0x01) == 1)  {
      if (S1BUF == XOFF)  {
        do  {
            S1CON = S1CON & 0xFE;
          while ((S1CON & 0x01) == 0);
        }
        while (S1BUF != XON);
        S1CON = S1CON & 0xFE;
      }
    }
    while ((S1CON & 0x02) == 0);
    S1CON = S1CON & 0xFD;
    return (S1BUF = c);
  }
```