```
/*=====================================================================*/
/* Projekt      :   Olli - eine Holzkiste lernt laufen                 */
/*                                                                     */
/* Hardware     :   Infineon C509                                      */
/*                                                                     */
/* Dateiname    :   seriell_olli_seriell.c                             */
/*                                                                     */
/* Version      :   1.7 vom 14. Februar 2004                           */
/*                                                                     */
/* Authoren     :   L. Kulf, J. Roos                                   */
/*                                                                     */
/* Dateigroesse :   xxxx kByte                                         */
/*                                                                     */
/* Datei-                                                              */
/* beschreibung :   Hier sind die Funktionen, die die Seriellen Schnittstellen*/
/*                  initialisieren, kontrollieren und interpretieren.  */
/*                                                                     */
/*                                                                     */
/*=====================================================================*/

// Weitere Informationen ------------------------------------------------
//Eine kompletter Befehls-Frame Matlab <> Olli besteht aus 4 Bytes.
//Code Byte 0> steht die Empfänger Adresse ('A' steht für Olli; 'B' = Matlab)
//Code Byte 1> hier steht die Sitzplatz Nr (0...9)
//Code Byte 2> hier steht die Mic pos. (0..9)
//Code Byte 3> Paritätsberechnung (Byte1+Byte2 dann zählen der "Einser"
//             gerade Anzahl "einser" = x; ungerade Anzahl "einser" = y)

//Eine kompletter Befehls-Frame LCD > Olli besteht aus 5 Bytes.
//Code Byte 0: hier steht immer "0x04" als Startzeichen
//Code Byte 1: steht die Empfänger Adresse: "A" für Olli
//Code Byte 2: hier steht das erste Info Byte (eine Zahl von 0..9)
//Code Byte 3: hier steht das zweite Info Byte(eine Zahl von 0..9)
//Code Byte 4: Paritätsberechnung (Byte2+Byte3 dann zählen der "Einser"
//             gerade Anzahl "einser" = x; ungerade Anzahl "einser" = y

//Eine kompletter Befehls-Frame Olli > LCD besteht aus 5 Bytes.
//Code Byte 0:  hier steht immer "0x1B" als Startzeichen
//Code Byte 1:  hier steht immer "M"
//Code Byte 2:  hier steht das erste Info Byte
//              (ASCII Zeichen für die Großbuchstaben 'N' 'P' 'T')
//Code Byte 3:  hier steht das zweite Info Byte (von 0x00 bis 0x255)
//Code Byte 4:  hier steht immer "0x0D" als Abschlußzeichen

// Einbinden von Dateien ------------------------------------------------

#include "main.h"
#include "seriell_olli_seriell.h"

// Globale Variablendeklaration Serielle Schnittstelle 0 & 1-------------------
volatile unsigned char sendebussy;
//nur zur Debuggzwecken: wird 1 wenn zu Debugzwecken gesendet wird

// Globale Variablendeklaration Serielle Schnittstelle 0-----------------------
unsigned char index_s0_outbuf = 0;
//Index Variable die als Zeiger für das "se0_outbuf" Array verwendet wird.
unsigned char s0_outbuf[BUFFER_SIZE_S0_OUT];
//Array das als Output Buffer für die S0 Schnittstelle verwendet wird.
unsigned char befehl_s0_in [4];
//Array in dem der aktuellste Befehls-Frame steht welcher als nächster ausge-
//führt werden soll.

// Globale _bit Variablendeklaration Serielle Schnittstelle 0-------------------
_bit m_new_move_order = 0;
//Diese Bit wird gesetzt wenn der aktuellste Befehls-Frame komplett auf
//plausibilität überprüft wurde und dann zur Ausführung bereit steht.
_bit m_error_s0 = 0;
//Dieses Bit wird gesetzt wenn beim Empfang oder bei der Plausinilitätsüber-
//prüfung der über S0 empfangenen Daten ein Fehler Auftritt. Im Moment setzte
//ich diese Bit immer dann zurück, wenn ein Befehls-Frame neu empfangen wurde.
_bit m_frame_s0_fertiggesendet = 1;
//Wert "1": ein Befehls-Frame ist über S0 fertig versendet worden.
//          Das ist der Ruhezustand.
//Wert "0": ein Befehls-Frame ist noch nicht fertig versendet worden.
//          Das ist der "In Arbeit" Zustand.

// Globale Variablendeklaration Serielle Schnittstelle 1-----------------------
unsigned char index_s1_outbuf = 0;
//Index Variable die als Zeiger für das "se1_outbuf" Array verwendet wird.
unsigned char s1_outbuf[BUFFER_SIZE_S1_OUT];
//Array das als Output Buffer für die S0 Schnittstelle verwendet wird.
unsigned char befehl_s1_in [5];
//Array in dem der aktuellste Befehls-Frame steht welcher als nächster ausge-
//führt werden soll.

// Globale _bit Variablendeklaration Serielle Schnittstelle 1-------------------
_bit m_new_lcd_order = 0;
//Diese Bit wird gesetzt wenn der aktuellste Befehls-Frame komplett auf
//plausibilität überprüft wurde und dann zur Ausführung bereit steht.
```

```
_bit m_error_s1 = 0;
//Dieses Bit wird gesetzt wenn beim Empfang oder bei der Plausinilitätsüber-
//prüfung der über S1 empfangenen Daten ein Fehler Auftritt. Im Moment setzte
//ich diese Bit immer dann zurück, wenn ein Befehls-Frame neu empfangen wurde.
_bit m_frame_s1_fertiggesendet = 1;
//Wert "1": ein Befehls-Frame ist über S1 fertig versendet worden.
//          Das ist der Ruhezustand.
//Wert "0": ein Befehls-Frame ist noch nicht fertig versendet worden.
//          Das ist der "In Arbeit" Zustand.

//Programmanfang ----------------------------------------------------------

//*****************************************************************************
//interrupt(4)void interrupt_seriell_S0 (void)
//Das ist die Interrupt Service Routine für Interrupts der Seriellen
//Schnittstelle S0
_interrupt(4) void interrupt_seriell_S0 (void)
{
    if ( RI0 )
    {
        seriell_S0_int_receive();
    }
    if ( TI0 )
    {
        seriell_S0_int_transmit();
    }
}

//*****************************************************************************
//_interrupt(16) void interrupt_seriell_S1 (void)
//Das ist die Interrupt Service Routine für Interrupts der Seriellen
//Schnittstelle S1
_interrupt(16) void interrupt_seriell_S1 (void)
{

    if (S1CON & 0x01)
    {
        seriell_S1_int_receive();
    }
    if (S1CON & 0x02)
    {
        seriell_S1_int_transmit();
    }
}

//*****************************************************************************
//Initialisierung der Seriellen Schnittstelle 0
void seriell_S0_init (void)
{
        SM0 = 0;
        SM1 = 1;
        //im SFR S0CON(98h) sind die Bits 6 und 7 für die ModeEinstellung
        //zuständig. SM0 = 0; SM1 =1 für Mode 1 oder: S0CON = 0xC0h
        REN0 = 1;
        //Receiver Enable Bit im S0CON Byte, ermöglicht seriellen Empfang, wenn
        //gesetzt. Wird nur durch Software beeinflußt
        BD = 1;
        //im SFR ADCON0(D8h) BD=1 (7Bit) um den Baudratengenerator anzuschalten:
        //muß auf "1" gesetzt werden
        //Nach Reset ist diese Bit auf "0"
        //Diese Bit ist direkt ansprechbar
        PCON = (PCON & 0x7F);
        //im SFR PCON(87h) ist SMOD das 7Bit,
        //damit kann die Baudrate verdoppelt werden.
        //wenn Bit SMOD: "1" ist, dann wird Baudrate verdoppelt.
        //Nach Reset ist dieses Bit und das Byte auf "0"
        //Diese Bit ist nicht direkt ansprechbar
        PRSC  = (PRSC | 0x40);
        //im SFR PRSC(B4h) ist S0P das 6Bit, damit wird die Baudrate verdoppelt
        //Nach Reset ist diese Bit auf "1"
        //Nach Reset ist dieses Byte auf "1101 0101b"
        //Diese Bit ist nicht direkt ansprechbar.
        S0RELH  = 0x03;
        //damit werden die 2 oberen MSBits vom 10Bit Zählregister eingestellt
        //diese sind das 0 und 1 Bit im S0RELH SFR
        //nach Reset ist dieses Byte auf "0000 0011b" = "0x03h" eingestellt.
        //für 9600Baud muß diese Byte auf "xxxx xx11b" = "0x03h"
        //eingestellt sein.
        S0RELL  = 0xE6;
        //damit werden die 8 unteren LSBits vom 10Bit Zählregister eingestellt.
        //nach Reset ist dieses Byte auf "1101 1001b" = "0xD9h" eingestellt.
        //für 9600Baud muß diese Byte auf "1110 0110b" = "0xE6h" eingestellt.
        //Die Formel zu Berechung der 10Bits (SOREL)ist:
        //[1024 - ( (16Mhz) / (64*Bdrate) ) ]= "998d"
        //lt. SiemensBuch ist die Formel:
        //[(2hochSM0D x 16hoch6) / (64 x (1024-S0REL) x 2hochS0P)]
}

//*****************************************************************************
```

```
    //seriell_S0_int_transmit
180         {
                m_frame_s0_fertiggesendet = true;
                REN0 = 1;
                //Der Empfang über S0 ist jetzt wieder eingeschaltet.
            }
185         else
            {
                index_s0_outbuf ++;
            }
        }
190 }

    //*****************************************************************************
    //seriell_S0_int_receive
    //Der Aufruf dieser Funktion erfolgt aus der Funktion: "interrupt_seriell_S0"
195 //Der Aufruf der "interrupt_seriell_S0" Funktion erfolgt automatisch, nachdem
    //über S0 Daten empfangen wurden. Die empfangen Daten werden in ein Array
    //gespeichert. Um einen korrekten Befehlsframe zu empfangen muß als
    //erstes Zeichen das 'A' empfangen werden, ab dann wird wird 'frame_s0_in_ready'
    //bis 4 hochgezählt. Somit fängt ein Befehlsframe immer mit 'A' an und hat
200 //dann noch 3 folgende Zeichen.
    //Sobald ein Befehls-Frame komplett ist, wird es an die Funktion
    //"interpreter_S0_receive" zur Plausibilitätsprüfung übergeben.
    void seriell_S0_int_receive(void)
    {
205     static unsigned char frame_s0_in_ready = 0;
        //wenn die Variable den Wert >=3 angenommen hat(0..3), dann ist ein
        //Befehls-Frame komplett empfangen worden.
        //Der Befehls-Frame kann dann Interpretiert werden
        static unsigned char index_s0_inbuf = 0;
210     //Index Variable die als Zeiger für das "se0_buf_in" Array verwendet wird.
        static unsigned char s0_inbuf[BUFFER_SIZE_S0_IN];
        //Array das als Input Buffer für die S0 Schnittstelle verwendet wird.

        RI0 = 0;
215     //Nach Empfang eines Zeichens wird das Interrupt-Request Bit RI0 gesetzt.
        //Nach dem auslesen des S0BUF wird das RI0 zurückgesetzt.
        s0_inbuf[index_s0_inbuf] = S0BUF;
        //der Inhalt von S0BUF wird der Variablen x zugewiesen
        if (frame_s0_in_ready == 0)
220     {
            if (s0_inbuf[index_s0_inbuf]== 'A')
            {
                index_s0_inbuf ++;
                frame_s0_in_ready ++;
225         }
            else
            {
                if(index_s0_inbuf >= 9)
                {
230                  index_s0_inbuf = 0;
                     seriell_S0_clear_in_buffer(s0_inbuf);
                }
                else
                {
235                 index_s0_inbuf ++;
                }
            }
        }
        else
240     {
            index_s0_inbuf ++;
            frame_s0_in_ready ++;

            if(frame_s0_in_ready == 4)
245         {
                frame_s0_in_ready = 0;
                index_s0_inbuf =  index_s0_inbuf - 4;
                interpreter_S0_receive(index_s0_inbuf, s0_inbuf);
                index_s0_inbuf =  index_s0_inbuf + 4;
250
                if(index_s0_inbuf >=9)
                {
                    index_s0_inbuf = 0;
                    seriell_S0_clear_in_buffer(s0_inbuf);
255             }
            }
        }
    }

260 //*****************************************************************************
    //interpreter_S0_receive
    //Der Aufruf der Funktion erfolgt aus der Funktion: "seriell_S0_int_receive"
    //Dieser Aufruf erfolgt wenn ein Befehls-Frame komplett empfangen wurde. In
    //dieser Funktion wird der Befehls-Frame auf Plausibilität überprüft. Ist diese
265 //Plausibilität gegeben, dann wird "m_new_move_order = 1" gesetzt. Das bewirkt
    //das in der "main" Funktion die Abgefrage nach einem neuen Befehls-Frame mit
    //"ja" beantwortet wird.
```

```
//Am Schluß wird dann die Funktion "transmit_S0_vorbereitung(0)" aufgerufen
//und es wird Ihr der Wert "0" übergeben. Dieser Wert wird dann in s0_outbuf[2]
//reingeschrieben. Die "0" bedeutet: "Befehl verstanden (B00x)"an Matlab senden
void interpreter_S0_receive(BYTE my_index, BYTE s0_inbuf[])
{
    unsigned char paritaet = 0;
    //Lokale Hilfsvariable
    unsigned char i;
    //Lokale Hilfsvariable Schleifenzähler usw..

//Übergabe des Array s0_buf_in das Array befehl_s0_in++++++++++++++++++++++++++
    for (i=0;i<=3;i++)
    {
        befehl_s0_in[i] = s0_inbuf[my_index];
        my_index ++;
    }
    befehl_s0_in[1] = (befehl_s0_in[1] - '0');
    //Der Inhalt (ASC II für eine Zahl von 0..9)wird in eine Int Zahl gewandelt
    befehl_s0_in[2] = (befehl_s0_in[2] - '0');
    //Der Inhalt wird in eine Int Zahl gewandelt

//Auswertung befehl_s0_in[0] = Code Byte0++++++++++++++++++++++++++++++++++++++++
//muß nicht mehr erfolgen, da nur noch BefehlsFrames an 'interpreter_S0_receive'
//übergeben werden die mit einem 'A' beginnen.....

//Auswertung befehl_s0_in[1] = Code Byte1++++++++++++++++++++++++++++++++++++++++
    if ( (befehl_s0_in[1] > 9) | (befehl_s0_in[1] < 0) )
    //Sitzplatznummer ist unlogisch da nicht 0..9 damit liegt ein Interpreter
    //Fehler vor
    {
        m_error_s0 = true;
    }

//Auswertung befehl_s0_in[2] = Code Byte2++++++++++++++++++++++++++++++++++++++++
    if ( (befehl_s0_in[2] > 6) | (befehl_s0_in[2] < 1) )
    //Fehler: "befehl[2]" kann nur 1 oder 2 oder 3 oder 4 oder 5 oder 6 sein
    //sonst ist die falsche Mic Position angegeben.
    {
        m_error_s0 = true;
    }

//Paritätscheck ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    paritaet = paritaetsberechnung(befehl_s0_in[1],befehl_s0_in[2]);

    if (!(paritaet == befehl_s0_in[3]))
    {
        //Fehler: gerade - ungerade Berechnung ergibt einen Fehler
        m_error_s0 = true;
    }

//m_error_s0_interpreter Check++++++++++++++++++++++++++++++++++++++++++++++++++
    if (m_error_s0 == true)
    //Wenn ein interpreter Fehler aufgetreten ist, wird an Matlab gesendet:
    //"Befehl nicht verstanden (B09x)". Matlab wird daraufhin den Befehls-Frame
    //erneut senden.
    {
        seriell_S0_clear_in_buffer(s0_inbuf);
        // zur Sicherheit wird der ganze Inhalt von 's0_inbuf' gelöscht
        transmit_S0_vorbereitung(9);
        //die Funktion transmit_S0_vorbereitung wird aufgerufen und es wird
        //der Wert "9" übergeben. Dieser Wert wird dann in s0_outbuf[2]
        //geschrieben um den Frame "Befehl nicht verstanden (B09x)" an
        //Matlab zu senden.
        m_error_s0 = false;
    }
    else
    //Wenn kein Fehler aufgetreten ist kann jetzt
    //an Matlab gesendet werden: "Befehl verstanden (B00x)"
    {
        m_new_move_order = true;
        //Der Globale Merker wird gesetzt, um Aufzuzeigen, dass ein neuer
        //Bewegungsbefehl fehlerfrei über S0 empfangen wurde und jetzt zur
        //Abarbeitung bereit steht. In der Globale Variable "befehl_s0_in[0..3]
        //steht an der Stelle [2] die Info für die Mic. Position die der Olli
        //jetzt anfahren soll. Jürgen bitte darauf achten: Bei Abarbeitung von
        //dem neuen Befehl, den "m_new_move_order" wieder zurücksetzten!
        transmit_S0_vorbereitung(0);
        //die Funktion transmit_S0_vorbereitung wird aufgerufen und es wird
        //der Wert"0"übergeben. Dieser Wert wird dann in s0_outbuf[2] rein-
        //geschrieben um den Frame"Befehl verstanden(B00x)" an Matlab zu senden
    }
}

//*****************************************************************************
//transmit_S0_vorbereitung
//Der Aufruf dieser Funktion erfolgt aus der Funktion: "interpreter_S0_receive"
//und "main". Der Wert der an die Funktion übergeben wird, ist der Inhalt vom
//Code Byte2. Ein Daten-Frame besteht aus 4 Byte.
void transmit_S0_vorbereitung(unsigned char wert)
```

```
{
    seriell_S0_clear_out_buffer();
    if ((wert == 0)|(wert == 7)|(wert == 8)|(wert == 9))
    {
        s0_outbuf[0] = 'B';
        s0_outbuf[1] = '0';
        s0_outbuf[2] = wert + '0';
        s0_outbuf[3] = paritaetsberechnung(s0_outbuf[1],s0_outbuf[2]);
        m_frame_s0_fertiggesendet = false;
        index_s0_outbuf = 0;
    }
    else
    {
        s0_outbuf[0] = 'B';
        s0_outbuf[1] = befehl_s0_in[1] + '0';
        s0_outbuf[2] = wert + '0';
        s0_outbuf[3] = paritaetsberechnung(s0_outbuf[1],s0_outbuf[2]);
        m_frame_s0_fertiggesendet = false;
        index_s0_outbuf = 0;
    }
}

//*****************************************************************************
//seriell_S0_clear_in_buffer
//Der Aufruf erfolgt aus der Funktion: "interpreter_S0_receive"
//Das ganze "s0_inbuf" Array wird geleert.
void seriell_S0_clear_in_buffer(BYTE s0_inbuf[])
{
    unsigned char i;
    for (i=0; i <= (BUFFER_SIZE_S0_IN -1); i++)
    {
        s0_inbuf[i]= 0x00;
    }
}

//*****************************************************************************
//seriell_S0_clear_out_buffer
//Der Aufruf erfolgt aus der Funktion: "main"
//Das ganze "s0_outbuf" Array wird geleert.
void seriell_S0_clear_out_buffer(void)
{
    unsigned char i;
    for (i=0; i <= (BUFFER_SIZE_S0_OUT -1); i++)
    {
        s0_outbuf[i]= 0x00;
    }
}

//*****************************************************************************
//seriell_S0_clear_index
//Der Aufruf erfolgt aus der Funktion: "main"
//Zähler Indexe für Sendepuffer werden zurückgesetzt. Das wird beim
//Programmstart gemacht.
void seriell_S0_clear_index (void)
{
    index_s0_outbuf = 0;
    m_frame_s0_fertiggesendet  = 1;
}

//*****************************************************************************
//Initialisierung der Seriellen Schnittstelle 1
void seriell_S1_init (void)
{
        S1CON = 0xD0;
        //ist gesetzt auf "1101 0000b" oder "0xD0h"
        //im SFR S1CON sind folgende Bits:
        //Nach Reset ist diese Byte auf "0100 0000b" oder "0x40h"
        //bit 0 (RI1): Rx(receiver)Interrupt flag
        //bit 1 (TI1): Tx(transmitt)Interrupt flag
        //bit 2 (RB81): 9 tes receiver bit
        //bit 3 (TB81): 9 tes transmitter bit
        //bit 4 (REN1): wenn 0 dann ist kein Serieller Empfang möglich
        //bit 5 (SM21): Mutiprocessor enable (bleibt immer auf "0")
        //bit 6 (S1P):  wenn 1(nach Reset)prescaler is active
        //              (für Baudraten Generator)
        //bit 7 (SM):   Mode select bit:0=9bit Uart; 1=8bit Uart
        //Diese Bits in diesem Byte sind nicht direkt ansprechbar !!
        S1RELH  = 0x03;
        //damit werden die 2 oberen MSBits vom 10Bit Zählregister eingestellt
        //diese sind das 0 und 1 Bit im S1RELH SFR
        //nach Reset ist dieses Byte auf "00000011b" = "0x03h" eingestellt.
        //für 9600Baud muß diese Byte auf "xxxxxx11b" ="0x03h" eingestellt sein
        S1RELL  = 0xCC;
        //damit werden die 8 unteren LSBits vom 10Bit Zählregister eingestellt.
        //nach Reset ist dieses Byte auf "00000000b" = "0x00h" eingestellt.
        //für 9600Baud muß diese Byte auf "1100 1100b" ="0xCC" eingestellt sein
        //entspricht "972d"
        //Die Formel zu Berechung der 10Bits (S1REL)ist:
        //[ 1 / {(Bdrate*32) / 16hoch6} ] + 1024
```

```
        //lt. SiemensBuch ist die Formel:
        //[ 16hoch6 / (32 x (1024-S1REL) x 2 hoch S1P)]
}

//*****************************************************************************
//seriell_S1_int_transmit
//Der Aufruf der Funktion erfolgt aus der Funktion: "interrupt_seriell_S1"
//Dieser Aufruf erfolgt automatisch, nachdem über S1 Daten gesendet werden.
//Die gesendeten oder zusendeten Daten sind im Array "s1_outbuf" gespeichert.
//Es wird überprüft ob schon alle Daten gesendet wurden, wenn nicht wird der
//Zeiger des Arrays hochgesetzt.
void seriell_S1_int_transmit (void)
{
    S1CON = (S1CON & 0xFD);
    //Ausmaskierung von Bit 1 in S1CON (TI1 wird auf "0" gesetzt)
    sendebussy = 0;
    if (m_frame_s1_fertiggesendet == false)
    {
        if (s1_outbuf[index_s1_outbuf+1] == 0x00)
        //Überprüfung, ob das nächste Zeichen im Array 0x00 ist. Wenn es 0x00
        //ist, dann sind die Daten fertig gesendet worden, egal wieviel Bytes
        //es waren (max 5). Wenn das nächste Zeichen nicht 0x00 ist, wird der
        //Zeiger (index_s1_outbuf) erhöht, und es kann weiter gesendet werden.
        //Diese Überprüfung auf das 0x00 Zeichen geht nur, da ich in der
        //"transmit_S1_vorbereitung" jedesmal das ganze s1_outbuf Array leere.
        {
            m_frame_s1_fertiggesendet = true;
            S1CON = (S1CON | 0x10);
            //Empfang über S1 isr jetzt wieder eingeschaltet.
        }
        else
        {
            index_s1_outbuf ++;
        }
    }
}

//*****************************************************************************
//seriell_S1_int_receive
//Der Aufruf dieser Funktion erfolgt aus der Funktion: "interrupt_seriell_S1"
//Dieser Aufruf erfolgt automatisch, nachdem über S1 Daten empfangen werden.
//Die empfangen Daten werden in ein Array gespeichert.
void seriell_S1_int_receive (void)
{
    static unsigned char frame_s1_in_ready = 0;
    //wenn die Variable den Wert >=4 angenommen hat(0..4), dann ist ein
    //Befehls-Frame komplett empfangen worden.
    //Der Befehls-Frame kann dann Interpretiert werden.
    static unsigned char index_s1_inbuf = 0;
    //Index Variable die als Zeiger für das "se1_buf_in" Array verwendet wird.
    static unsigned char s1_inbuf[BUFFER_SIZE_S1_IN];
    //Array das als Input Buffer für die S0 Schnittstelle verwendet wird.

    S1CON = (S1CON & 0xFE);
    //Ausmaskierung von Bit 0 in S1CON (RI1 wird auf "0" gesetzt)
    s1_inbuf[index_s1_inbuf] = S1BUF;
    //Im Array "se1_inbuf" wird an der Stelle, auf die, die Index Variable
    //"index_se1_inbuf" zeigt, der Inhalt der Variablen x gespeichert.
    if (frame_s1_in_ready == 0)
    {
        if (s1_inbuf[index_s1_inbuf]== 0x04)
        {
            index_s1_inbuf ++;
            frame_s1_in_ready ++;
        }
        else
        {
            if(index_s1_inbuf >= 10)
            {
                 index_s1_inbuf = 0;
                 seriell_S1_clear_in_buffer(s1_inbuf);
            }
            else
            {
                index_s1_inbuf ++;
            }
        }
    }
    else
    {
        index_s1_inbuf ++;
        frame_s1_in_ready ++;

        if(frame_s1_in_ready == 5)
        {
            frame_s1_in_ready = 0;
            index_s1_inbuf =  index_s1_inbuf - 5;
            interpreter_S1_receive(index_s1_inbuf, s1_inbuf);
            index_s1_inbuf =  index_s1_inbuf + 5;
```

```
            if(index_s1_inbuf >= 10)
            {
                index_s1_inbuf = 0;
                seriell_S1_clear_in_buffer(s1_inbuf);
            }
        }
    }
}

//*****************************************************************************
//interpreter_S1_receive
//Der Aufruf der Funktion erfolgt aus der Funktion: "seriell_S1_int_receive"
//Dieser Aufruf erfolgt wenn ein Befehls-Frame komplett empfangen wurde. In
//dieser Funktion wird der Befehls-Frame auf Plausibilität überprüft. Ist diese
//Plausibilität gegeben, dann wird "m_new_lcd_order = 1" gesetzt. Das bewirkt
//das in der "main" Funktion die Abgefrage nach einem neuen lcd Befehls-Frame mit
//"ja" beantwortet wird.
void interpreter_S1_receive(BYTE my_index, BYTE s1_inbuf[])
{
    unsigned char paritaet = 0;
    //Lokale Hilfsvariable
    unsigned char i;
    //Lokale Hilfsvariable Schleifenzähler usw..

    m_error_s1 = false;
    //error Variable wird gelöscht

//Übergabe des Array s1_inbuf_in das Array befehl_s1_in++++++++++++++++++++++++++
    for (i=0;i<=4;i++)
    {
        befehl_s1_in[i] = s1_inbuf[my_index];
        my_index ++;
    }
    befehl_s1_in[2] = (befehl_s1_in[2] - '0');
    //Der Inhalt (ASCii für eine Zahl von 0..9)wird in eine Int Zahl gewandelt
    befehl_s1_in[3] = (befehl_s1_in[3] - '0');
    //Der Inhalt wird in eine Int Zahl gewandelt

//Auswertung befehl_s1_in[0] = Code Byte0++++++++++++++++++++++++++++++++++++++++
//muß nicht mehr erfolgen, da nur noch BefehlsFrames an 'interpreter_S1_receive'
//übergeben werden die mit einem '0x04' beginnen.....

//Auswertung befehl_s1_in[1] = Code Byte0++++++++++++++++++++++++++++++++++++++++
    if (befehl_s1_in[1] != 'A')
    //Fehler: "befehl_s1_in[1]" muß "A" sein sonst Fehler bei der Addressierung
    {
        m_error_s1 = true;
    }

//Auswertung befehl_s1_in[2] = Code Byte1++++++++++++++++++++++++++++++++++++++++
    if (!((befehl_s1_in[2] >= 0)&(befehl_s1_in[2] <= 9)))
    //Fehler: "befehl_s1_in[2]" kann nur 0..9 sein
    {
        m_error_s1 = true;
    }

//Auswertung befehl_s1_in[3] = Code Byte2++++++++++++++++++++++++++++++++++++++++
    if (!((befehl_s1_in[3] >= 0)&(befehl_s1_in[3] <= 9)))
    //Fehler: "befehl_s1_in[3]" kann nur 0..9 sein
    {
        m_error_s1 = true;
    }

//Paritätscheck ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    paritaet = paritaetsberechnung(befehl_s1_in[2],befehl_s1_in[3]);
    if (!(paritaet == befehl_s1_in[4]))
    {
        //Fehler: gerade - ungerade Berechnung ergibt einen Fehler
        m_error_s1 = true;
    }

//m_error_s1_ Check++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    if (m_error_s1 == true)
    //Wenn ein Fehler aufgetreten ist ??
    {
        //Was soll passieren wenn "m_error_s1 == true" steht?
        seriell_S1_clear_in_buffer(s1_inbuf);
    }
    else
    //Wenn kein Fehler aufgetreten ist kann jetzt der s1 merker gesetzt werden
    {
        m_new_lcd_order = true;
        //Der Globale Merker wird gesetzt, um Aufzuzeigen, dass ein neuer Touch
        //Panel Befehl fehlerfrei über S1 empfangen wurde und jetzt zur
        //Abarbeitung bereit steht. In dem Globalen Variable "befehl_s1_in[0..4]
        //steht an der Stelle [2]&[3] die Info vom LCD Display
        //Jürgen bitte darauf achten: Bei Abarbeitung von dem neuen LCD Befehl
        //den Merker "m_new_lcd_order" wieder zurücksetzten!
```

```
    }
}

//*****************************************************************************
//transmit_S1_vorbereitung
//Der Aufruf dieser Funktion erfolgt aus der Funktion: "interpreter_S1_receive"
//und aus der "main" Funktion. Die Werte die an "transmit_S1_vorbereitung"
//übergeben werden, sind dann der Inhalt vom s1_outbuf[2] und s1_outbuf[3].
//Ein Daten-Frame besteht aus 5 Byte.
void transmit_S1_vorbereitung(unsigned char Byte2, unsigned char Byte3)
{
    seriell_S1_clear_out_buffer ();
    s1_outbuf[0] = 0x1B;
    s1_outbuf[1] = 'M';
    s1_outbuf[2] = Byte2;
    s1_outbuf[3] = Byte3;
    s1_outbuf[4] = 0x0D;
    m_frame_s1_fertiggesendet = false;
    index_s1_outbuf = 0;
}

//*****************************************************************************
//seriell_S1_clear_in_buffer
//Der Aufruf erfolgt aus der Funktion: "seriell_S1_int_receive"
//Das ganze "s1_inbuf" Array wird geleert.
void seriell_S1_clear_in_buffer(BYTE s1_inbuf[])
{
    unsigned char i;
    for (i=0; i <= (BUFFER_SIZE_S1_IN -1); i++)
    {
        s1_inbuf[i]= 0x00;
    }
}

//*****************************************************************************
//seriell_S1_clear_out_buffer
//Der Aufruf erfolgt aus der Funktion: "main"
//Das ganze "s0_outbuf" Array wird geleert.
void seriell_S1_clear_out_buffer(void)
{
    unsigned char i;
    for (i=0; i <= (BUFFER_SIZE_S1_OUT -1); i++)
    {
        s1_outbuf[i]= 0x00;
        //Hier wird das ganze out Array geleert
    }
}

//*****************************************************************************
//seriell_S1_clear_index
//Der Aufruf erfolgt aus der Funktion: "main"
//Zähler Indexe für Sendepuffer werden zurückgesetzt. Das wird beim
//Programmstart gemacht.
void seriell_S1_clear_index (void)
{
    index_s1_outbuf = 0;
    m_frame_s1_fertiggesendet  = 1;
}


//*****************************************************************************
//paritaetsberechnung
//Der Aufruf dieser Funktion erfolgt aus folgenden Funktionen:
//"interpreter_S0_receive" und "transmit_S0_vorbereitung" und
//"interpreter_S1_receive"
//bei der Paritätsberechnung wird die Anzahl der "Einser" gezählt. Zuerst wird
//eine Addition der binären Werte von Byte1 und Byte2 gemacht. Wenn die Anzahl
//der "Einser" im Ergebnis eine gerade Zahl ist, dann ist der Returnwert der
//Funktion "x". Bei einer Ungeraden Anzahl von "Einser" wird "y" zurückgegeben
//x = gerade, y = ungerade
//Bsp.: 3 + 1 = 4 > 0011 + 0001 = 0100 > die Anzahl der "Einser" im Ergebnis
//wird gezählt. Die Anzahl ist 1 > ungerade > Returnwert = "y"
unsigned char paritaetsberechnung(unsigned char x,unsigned char y)
{
    unsigned char erg_add = 0;
    unsigned char anzahl_einser  = 0;
    unsigned char i;
    unsigned char paritaet = 0;

    erg_add = x + y;

    for (i=1;i<=8;i++)
    {
        anzahl_einser = anzahl_einser + (erg_add & 0x01);
        erg_add = erg_add >> 1;
    }

    if (anzahl_einser % 2)
    //wenn diese Abfrage wahr ist(>=1), dann ist die Anzahl der "1" ungerade
```

```
        {
            paritaet = 'y';
        }
        else
735     //if Abfrage = 0 dann muß die Anzahl der "1" gerade sein.
        {
            paritaet = 'x';
        }
        return paritaet;
740 }

    //*****************************************************************************
    //pause
    //Der Aufruf erfolgt aus der Funktion: "seriell_olli_seriell" und aus "main"
745 //pause ist nur zu Debugg zwecken!!!!!!
    void pause (BYTE i)
    {
        unsigned char b;
        unsigned char c;
750     for (c=0; c<=i; c++)
        {
            for(b=0;b<250;b++)
            {
            }
755     }
    }
    // Programmende ------------------------------------------------------------
```