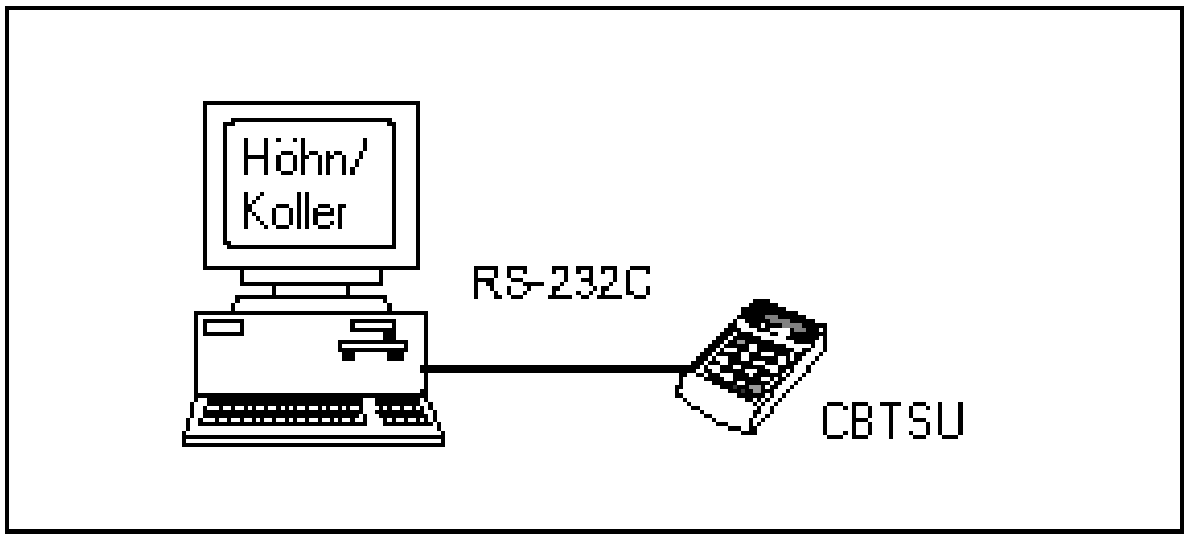
RS-Schnittstelle mit Interrupt bedient

RS232-Projektarbeit

Author: Koller/Höhn



Contents

1 Einleitung

Bei modernen uController Aufgaben spielt die Serielle Schnittstelle (serial comunication Interface: SCI) eine Zentrale Rolle. Durch dieses Interface werden beliebige Daten ausgetauscht zwischen verschiedensten Modulen.

Wir haben das SCI bereits kennen gelernt im uC-Unterricht. Dabei wurde ein String einer Funktion übergeben, welche diesen dann z.B. auf einen Terminal ausgegeben hat, um danach zum Hauptprogramm zurückzukehren. Um Daten zu empfangen wurde in einer Funktion auf die Daten gewartet, und nach Eintreten des Abschlusszeichens <CR> wieder zum Hauptprogramm zurückgesprungen.

Dieser Ablauf ist in einem uC-System nahezu unbrauchbar, da die Daten beliebig hin und her transportiert werden müssen. Um dieses Problem zu lösen bedient man sich mit den dafür vorgesehenen Interruptrequests.

2 Aufgabenstellung

Es soll ein C-Programm erstellt werden, welches die Steuerung der Ein- und Ausgabe über die RS232 Schnittstelle reglet. Das Hauptprogramm wird dabei von Inerrupts unterbrochen.

Es stellt eine fortlaufende Zeit im 1/100 sek. Takt auf dem Display dar. Dies soll verdeutlichen, dass das Hauptprogramm nie merklich unterbrochen wird. Der Timer geniesst eine höhere Interrupt Priorität, was dazu führt, dass es keine Verzögerung gibt.

Mit einem Schalter kann die Übermittlung der Aktuellen Zeit auf den Terminal starten. Findet ein Übertragungsfehler statt, wird dies auf der 4. Zeile angezeigt. Diese Zeile kann mit einem anderen Schalter wieder gelöscht werden.

Das Handshake Handling wird im Programm implementiert.

3 Lösungansatz

3.1 Transmit Data

Bei Betätigung eines Schalters wird die aktuell dargestellte Zeit in einen Buffer kopiert und danach zur Übertragung per Interrupt bereit gestellt werden

3.2 Recieve Data

Mit Hilfe des Hyper Terminals wird ein Zechen über die Computertastatur eingegeben. Dies löst auf dem Controller Board einen Ineterrupt aus, und wird da in einem Buffer gespeichert.

3.3 Software Handshake

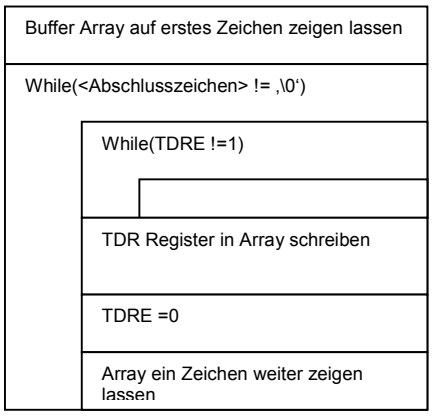
Wird vom Computer ein Xoff (0x13) geschickt, wird dies sofort in eine Merker Variable geschrieben und steht somit in der Transmit Routine als zusätzliche Information zur Verfügung. Das Übermitteln stoppt unverzüglich und es wird erst wieder weiter Übermittelt, wenn die Merker Variable mittels vom Computer erhaltenen Xon (0x11) einen dementsprechender Wert zugewiesen bekommt.

3.4 Error Erkennung und Zuweisung

Erfolgt ein Übertragungserror, wird eine weitere Interrupt Routine ausgelöst. Mit Hilfe der gesetzten Error Flags, ORER, PER und FER wird der Error identifiziert und auf der LCD ausgeschrieben.

4 C-Programmierung

4.1 Übertragung von Zeichen



Das Handshake Handling machen wir im Hauptprogramm:

if(((CBTSU_Schalter & 0x80)==0x80)&& sci_handshake == 0)

Wichtig beim Initialisieren ist, dass das TIE und TEIE flag in SCR erst bei Gebrauch zu setzen sind:

if(((CBTSU_Schalter & 0x80)==0x80)&& sci_handshake == 0)

{strcpy(tra_string_sci , disp_time);

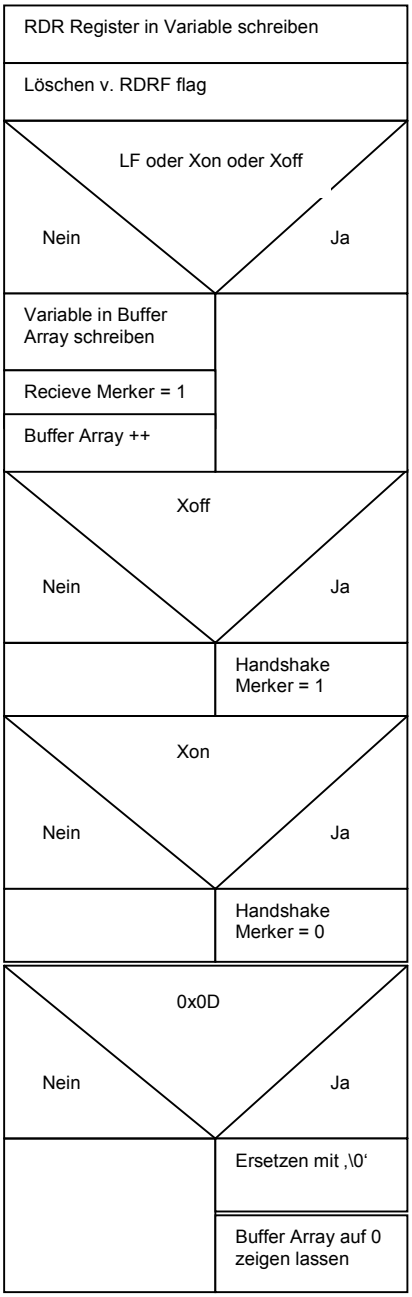
//TIE und TEIE flag in SCR setzen

SCI_SCR1 |= 0x84;}

Dies löst sofort einen Transmit Interrupt aus. Auch sehr wichtig, diese zwei flag sofort nach Gebrauch wieder auf 0 setzten:

SCI_SCR1 &= 0x7b;

4.2 Empfangen von Zeichen



5 Schlusswort

Es ist uns gelungen die geforderte Aufgabe zu bewältigen. Es kann deutlich gezeigt werden, dass das parallele Ablaufen von Timer, Empfangen und Übermitteln von Daten über das serielle Kommunikationsinterface. Die Prioritätenverteilung, und das Erkennen von Fehlern.

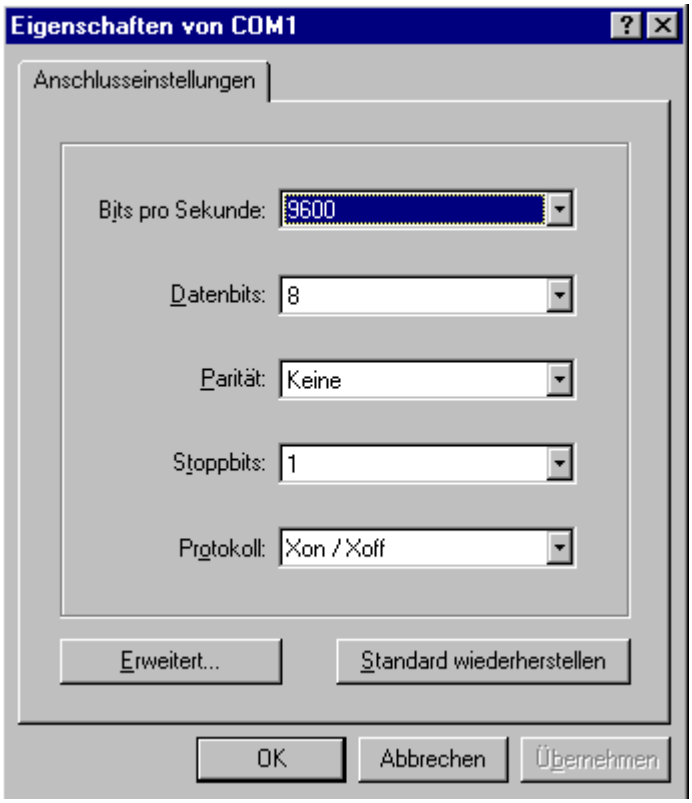
Es ist jedoch nocht nicht ganz fertig. Um die Anwendung auch noch in anderen Programmen zur Verfügung zu stellen, müsste nocht eine Aufteilung in Funktionen gemacht werden, und diese vielleicht sogar in einem separaten Programm unterzubringen.

Leider war es so, dass unser Projekt durch ein falsches Flashprogramm auf dem Controllerboard gar nicht funktioniert hat, und wir dies nicht gewusst haben. Sehr viel Zeit und Nerven gingen dabei verloren. Zu dem Zeitpunkt wo das Flashprogramm dann tatsächlich funktionstüchtig war, hatten wir gerade noch 2 Wochen Zeit.

Ich denke trotzdem, dass wir in dieser Übung sehr viel gelernt haben. Das Aufteilen in Funktionen wäre jetzt zwar schön, aber der eigentliche Sinn des Projektes haben wir erfüllt.

6 Anhang:

6.1 Terminal Einstellungen



6.2 C-Code

```
/*****************************************************************************
 * Projekt RS232 Interrupt Handler
 * fuer das Controller-Board der TSU
 * Autor: Hoehn und Koller
 * Dezember 2001
 *****************************************************************************
 */

#define BYTE unsigned char
#define WORD unsigned int
#define LONG unsigned long
#define max_line 21
#include <stdio.h>
#include <string.h>
#include "ioh83067.h"
#include "CB_TSU2.h"


LONG temp_time;
BYTE temp_eri_sci;
BYTE DataIn_sci;
BYTE dummy;
BYTE InBuffer[max_line];
BYTE rec_indi=0;
BYTE merker_schalter_2;
BYTE sci_handshake = 0;
BYTE count_rec_sci;
BYTE count_tra_sci;
BYTE hundreth;
BYTE sec;
BYTE min;
```

```
BYTE hour;
char disp_time[max_line];
char tra_string_sci[max_line];



//Timer Interrupt Routine
#pragma interrupt
void ISR_tic(void)
{
dummy =FRT_TISRA; FRT_TISRA = 0x88 + 0x10;
temp_time ++;
}

//Data Recieve Intrrupt Routine
#pragma interrupt
void ISR_RXI1(void)
{
//empfangener char in DataIn schreiben
DataIn_sci = SCI_RDR1;
//loeschen des RDRF flags in SSR
dummy = SCI_SSR1; SCI_SSR1 &= 0xBF;

//LF und Handshake nicht beachten. Nur 'gültige' zeichen ansch.
if((DataIn_sci != 0x0A) && (DataIn_sci != 0x13) && (DataIn_sci != 0x11))
        {
        InBuffer[count_rec_sci] = DataIn_sci;
        rec_indi=1;
        count_rec_sci++;
        }
//Handshake handler
if(DataIn_sci == 0x13)
        sci_handshake = 1;
if(DataIn_sci == 0x11)
```

```
        sci_handshake = 0;
//Abschlusszeichen setzen
if(DataIn_sci == 0x0D)
        {
        count_rec_sci--;
        InBuffer[count_rec_sci]='\0';
        count_rec_sci=0;
        }
}

//Transmit Data Inerrupt Routine
#pragma interrupt
void ISR_TXI1(void)
{
        count_tra_sci = 0;
//solange kein Abschlusszeichen kommt
        while(tra_string_sci[count_tra_sci])
        {
//warte solagne wie 'Transmit Data Register empty' in SSR nicht 1 ist
                while((SCI_SSR1 & 0x80) != 0x80);
                        {
//write transmit data in TDR
                                SCI_TDR1 = tra_string_sci[count_tra_sci];
//clear TDRE flag to 0 in SSR
                                dummy = SCI_SSR1;SCI_SSR1 &= 0x7F;
                                count_tra_sci ++;
                        }
        }

//Interrupt Anforderung wieder ausschalten
        SCI_SCR1 &= 0x7b;
}
```

```
//SCI Error Handling Interrupt Routine
#pragma interrupt
void ISR_ERI1(void)
{
temp_eri_sci=SCI_SSR1;
//ORER, FER, und PER flags auf Zustand prüfen
temp_eri_sci &= 0x38;
switch(temp_eri_sci)
        {
        case 0x20       :CBTSU_DispString(4,"Overrun Error      ");break;
        case 0x10       :CBTSU_DispString(4,"Framing Error      ");break;
        case 0x08       :CBTSU_DispString(4,"Parity Error       ");break;
        case 0x30       :CBTSU_DispString(4,"Ov & Fr Error      ");break;
        case 0x28       :CBTSU_DispString(4,"Ov & Pa Error      ");break;
        case 0x18       :CBTSU_DispString(4,"Fr & Pa Error      ");break;
        case 0x38       :CBTSU_DispString(4,"Ov, Fr & Pa Error  ");break;
        default         :CBTSU_DispString(4,"unbekannter Fehler ");break;
        }
//clear error flags to zero in SSR
        dummy = SCI_SSR1;SCI_SSR1 &= 0xC7;
}

main()
{
H8interrupt [FRT_IMIA0] H8proc ISR_tic;
H8interrupt [SCI_RXI1]  H8proc ISR_RXI1;
H8interrupt [SCI_TXI1]  H8proc ISR_TXI1;
H8interrupt [SCI_ERI1]  H8proc ISR_ERI1;

temp_time=0.0;
count_rec_sci=0;
count_tra_sci=0;
LCD_Init();
```

```
LCD_IWrite(0x01);

//Initialisieren der RS232 Schnittstelle nach Seite 497
//-----------------------------------------------------
//clear TE and RE bits to 0 in SCR1
//also set CKE0 und CKE1 to 0 in SCR1
SCI_SCR1 = 0x00;
//mode: asynch, 8, none, 1stop, phi
SCI_SMR1 = 0x00;
//baudrate: 9600
SCI_BRR1 = 0x3B;
//set RIE = 1 and RE =1 clk an SCK-Pin in SCR1
dummy=SCI_SSR1; SCI_SSR1&=0x04;
//controll flags setzen
dummy=SCI_SCR1; SCI_SCR1 = 0x70;

//Initialiseren des Timers
//------------------------
// GRA als Compare Register aktivieren
FRT_TIOR0 = 0x88 + 0x00;
// Timer Compare Wert laden, da 8 Bit Mode in zwei Schritten
FRT_GRA0L = (BYTE) (23040 % 0x100);
FRT_GRA0H = (BYTE) (23040 / 0x100);
// Automatisches Loeschen von TSTR aktivieren & Phi achtel
FRT_TCR0 = 0x80 + 0x23;
// Match Flag zur Sicherheit l”schen
dummy =FRT_TISRA; FRT_TISRA = 0x88 + 0x10;
// Timer starten
FRT_TSTR = 0xf8 + 0x01;


CBTSU_DispString(1,"Proj RS232 R/W ");
```

```
while(1)
        {
//Timer darstellen
   hundreth = temp_time % 100 ;
   sec = ((temp_time / 100) % 60);
   min = ((temp_time / 6000) % 60);
   hour = (temp_time / 360000);


        sprintf(disp_time,"%02hu:%02hu:%02hu:%02hu",hour,min,sec,hundreth);
   CBTSU_DispString(2,disp_time);



//diese Schleife wird nur durchgeführt, wenn Recieve Interrupt ausgelöst hat
/*in der Recieve Inerrupt Routine wird 'rec_indi' auf 1 gesetzt, und hier wieder auf 0, d.h.
es wird jedes einzelne Zeichen abholt und gebuffert*/
   if(rec_indi==1)
                {
                CBTSU_DispString(3,InBuffer);
                rec_indi=0;
                }
//wenn Schalter 8 umgelegt wird, loest ein Transmit Interrupt aus
//solange wie kein software handshake vorliegt
        if(((CBTSU_Schalter & 0x80)==0x80)&& sci_handshake == 0)
                {
                strcpy(tra_string_sci , disp_time);
//TIE und TEIE flag in SCR setzen
                SCI_SCR1 |= 0x84;
                }
//wenn Schalter 7 umgelegt wird, loest ein Transmit Interrupt aus
//solange wie kein software handshake vorliegt
   if(((CBTSU_Schalter & 0x40)==0x40)&& (merker_schalter_2 == 0)&&(sci_handshake == 0))
```

```
        {
        merker_schalter_2 = 1;
        strcpy(tra_string_sci , disp_time);
                SCI_SCR1 |= 0x84;
        }
   if((CBTSU_Schalter & 0x40)==0x00)
        merker_schalter_2 = 0;
//wenn Schalter 1 umgelegt wird, löschen wir die Error Anzeige
        if((CBTSU_Schalter & 0x01)==0x01)
                CBTSU_DispString(4,"                ");
   }
}
```