```
/*****************************************************************************
**
 * Projekt Arbeit Datenlogger
 * 29.10.2001
 * M.Richter / M.Hess / D.Büchler
 *
 *****************************************************************************
**
 */
#include "ioh83067.h"
#include "CB-TSU2.h"
#include <stdio.h>
#include <string.h>
#define BYTE unsigned char
#define WORD unsigned int
#define LONG unsigned long

#define max_Values 10000

//prototypes
void timer_config (WORD steps_, unsigned long int time_);
void int_tmr0 (void);
void Write_Data_RS(void);
void IToA(int n, char s[]);
void wait(void);

// Definition der structs
struct SData {unsigned char Switches;
                                unsigned int Time;};

volatile struct SData Data[max_Values];

//Variablen-Definition
unsigned char value;
unsigned int counter;                                           // Anzahl
Messwerte, wird von Int-Routine decrementiert
volatile unsigned long int pre_wait;      // 0=Messen bei jedem Interr.; >0
wait_mode
volatile unsigned long int pre_count;
unsigned int x;
volatile unsigned char done;                    // 0=interrupt, 1=fertig
volatile unsigned char Ok;                                      // volatile
is important!!!
volatile unsigned int steps;                                    // Var. zur
Eingabe der gewün. Anz. zu Loggenden Werte
unsigned char copy;



#pragma interrupt
void int_tmr0 (void)
{
unsigned char measure = 0;
unsigned char input_data;
copy = TM_TCSR0;
TM_TCSR0 = 0x00;
if (pre_wait != 0) {
        if (pre_count != 0)
                {
                pre_count --;
                }
```



```
        else {
                measure = 1;
                pre_count = pre_wait;
                } // of if pre_count
        } else {
        measure = 1;
        } // of if pre_wait
        if (counter == 0) {
                TM_TCR0 = 0x00; // Interrupt sperren
                measure = 0;
                done = 1;
                } // of if counter
        if (measure == 1) {
                input_data = CBTSU_Schalter;
                CBTSU_LED_Gruen = input_data;
                Data[counter].Switches = input_data;
                Data[counter].Time = (steps - counter); // JUST TEST!!!
                counter --;
        } // of if pre_wait
} // of int_tmr0

#pragma interrupt
void ISR_IRQ0(void)
{
unsigned char dummy;
unsigned char blubber;

    dummy = ISR;
    ISR = ISR & 0xfe;   // clears IRQ-flag
    blubber = CBTSU_Keyboard - 0xf0;
    switch (blubber)
    {
    case 13 : value = 0; break;
    case 0 : value = 1; break;
    case 1 : value = 2; break;
    case 2 : value = 3; break;
    case 4 : value = 4; break;
    case 5 : value = 5; break;
    case 6 : value = 6; break;
    case 8 : value = 7; break;
    case 9 : value = 8; break;
    case 10 : value = 9; break;
    case 12 : value = 10; break;       // A
    case 14 : value = 11; break;       // B
    case 15 : value = 12;
                          x = x / 10;
                          break;                                    // C
    case 11 : value = 13; break;                // D
    case 7 : value = 14;
                                Ok = 1;
                                break;
// E
     case 3 : value = 15; break;                // F
    } // of switch
    if (value <= 10) {
        x = (x * 10) + value;
    }
    CBTSU_DispInt(4,x);

} // of ISR_IRQ0

void timer_config (WORD steps_, unsigned long int time_)
```

```
{
        BYTE    cr;

        counter = steps_;
        if (time_ < 100)
        {
                cr = 2250 * (time_/1000);       // 2250 = Fquarz/Tpresc
                pre_count = 0;
        }
        else if ((time_ >= 100) && (time_ < 60000))
                {
                        cr = 225;                                       // entspricht
100ms
                        pre_count = time_ / 100;        // anzahl interrupts
                        pre_wait = pre_count;
                }

        TM_TCORA0       = cr;                                           // compare
reg laden, entspricht 100ms
        //TM_TCORA0     = 0x00;                                         // oberes
byte sicherheitshalber auf 0 init.
        TM_TCR0         = 0x00 + 0x4b;          // interrupt request,cleared
by compare match,presc_8192
   done = 0;
} // of timer_config

main(void)
{
unsigned long int time = 0;             // Var. zur Eingabe der gewünschten
Loggzeit
unsigned char read;
unsigned int choice;                                    // Var. zur Auswahl
des gewünschten Wertes

//Registrieren der Interrup-Service-Routinen
H8interrupt [IRQ_0] H8proc ISR_IRQ0;
H8interrupt [TM_CMIA0] H8proc int_tmr0;

//Initialisierung
LCD_Init();               // Init LCD
LCD_IWrite(0x01);         // LCD clear
PBDR|= 0x20;              // Display beleuchtung ein
steps = 0;
ISCR=0x01;
IER=0x01;

CBTSU_DispString(1,"-----Guten Tag!-----");
CBTSU_DispString(2,"Bitte jetzt         ");
CBTSU_DispString(3,"ueber das Keyboard  ");
CBTSU_DispString(4,"Config. vornehmen   ");
wait();
wait();
LCD_IWrite(0x01);
CBTSU_DispString(1,"Anzahl Messschritte ");
CBTSU_DispString(2," (E)nter (C)lear");
/* Beginn Keyboard-Scanner */
Ok = 0;
do {} while (Ok!=1);              // wait until E pressed!
steps = x;
LCD_IWrite(0x01);
CBTSU_DispInt(3,steps);           // Zeigt anzahl der Schritte auf Disp.
CBTSU_DispString(4,"Schritte");
```



```
/* End Keyboard-Scanner */

wait();
LCD_IWrite(0x01);
CBTSU_DispString(1,"Log-Zeit [ms]   ");
CBTSU_DispString(2," (E)nter (C)lear");

/* Beginn Keyboard-Scanner */
Ok = 0;
x = 0;
do {} while (Ok != 1);              // wait until E pressed!
time = x;
LCD_IWrite(0x01);
CBTSU_DispInt(3,time);              // Zeigt die ausgew.Loggzeit auf Disp.
CBTSU_DispString(4,"Zeit in (ms)");
/* End Keyboard-Scanner */
wait();

LCD_IWrite(0x01);
CBTSU_DispString(1,"********************");
CBTSU_DispString(2,"     logging...     ");
CBTSU_DispString(3,"********************");

timer_config(steps,time);       // Timer configure

do {} while (done != 1);                        // Warte bis Interrupt
beendet!

CBTSU_DispString(1,"********************");
CBTSU_DispString(2," Logging finished ! ");
CBTSU_DispString(3,"********************");
wait();

do
{
LCD_IWrite(0x01);
CBTSU_DispString(1,"Ausgabe         ");
CBTSU_DispString(2,"0 fuer RS232!   ");
CBTSU_DispString(3," (E)nter (C)lear");
/* Beginn Keyboard-Scanner */
Ok = 0;
x = 0;
do {} while (Ok!=1);                // wait until E pressed!
choice=x;
LCD_IWrite(0x01);
CBTSU_DispInt(1,choice);
CBTSU_DispString(2,"Messschritt");
/* End Keyboard-Scanner */
CBTSU_DispString(3,"betraegt");
if (choice == 0) {
        LCD_IWrite(0x01);
        CBTSU_DispString(2,"Sending data...");
        Write_Data_RS();
        } else {
        choice = steps - choice;
        CBTSU_DispInt(4,(int)Data[choice].Switches);     // Zeigt den
ausgewählten Wert auf Disp.
        wait();
        }
}
while(1);
}
```

```
void Write_Data_RS(void)
{
int i;
char x;
char Str[25];
char Str2[10];

CBTSU_RSInit();
CBTSU_RSWriteLn("\n");
CBTSU_RSWriteLn("** Datalogger **");
CBTSU_RSWriteLn("\n");
CBTSU_RSWriteLn("Value, Time");
CBTSU_RSWriteLn("\n");
for(i=steps;i>0;i--)
        {
        IToA(Data[i].Switches,Str);
        strcat(Str,",");
        IToA(Data[i].Time,Str2);
        IToA(i,Str2);
        strcat(Str,Str2);
        CBTSU_RSWriteLn(Str);
        CBTSU_RSWriteLn("\n");
        }; // of for
CBTSU_RSWriteLn("** End of Data **");
} // of Write_Data_RS

void IToA(int n, char s[])
{
int i;
int j;
int sign;

if ((sign = n) <0)
        n = -n;
i = 0;
do {
        s[i++] = n% 10 + '0';
} while ((n /= 10) < 0);
if (sign < 0)
        s[i++] = '-';
s[i] = '\0';
// s in Reihenfolge umkehren:
for(i=0,j=strlen(s)-1; i> j; i++, j--) {
        sign = s[i];
        s[i] = s[j];
   s[j] = sign;
}
} // of IToA

void wait(void) //Funkt.für kurze Pause zw.Disp.Anzeige
{
unsigned long int i;
for(i=0;i<1000000;i++);
} // of wait
```



```
/*****************************************************************************
**
 * Projekt Arbeit Datenlogger
 * 29.10.2001
 * M.Richter / M.Hess / D.Büchler
 *
 *****************************************************************************
**
 */
#include "ioh83067.h"
#include "CB-TSU2.h"
#include <stdio.h>
#include <string.h>
#define BYTE unsigned char
#define WORD unsigned int
#define LONG unsigned long

#define max_Values 10000

//prototypes
void timer_config (WORD steps_, unsigned long int time_);
void int_tmr0 (void);
void Write_Data_RS(void);
void IToA(int n, char s[]);
void wait(void);

// Definition der structs
struct SData {unsigned char Switches;
                                    unsigned int Time;};

volatile struct SData Data[max_Values];

//Variablen-Definition
unsigned char value;
unsigned int counter;                                           // Anzahl
Messwerte, wird von Int-Routine decrementiert
volatile unsigned long int pre_wait;      // 0=Messen bei jedem Interr.; >0
wait_mode
volatile unsigned long int pre_count;
unsigned int x;
volatile unsigned char done;                    // 0=interrupt, 1=fertig
volatile unsigned char Ok;                                      // volatile
is important!!!
volatile unsigned int steps;                                    // Var. zur
Eingabe der gewün. Anz. zu Loggenden Werte
unsigned char copy;



#pragma interrupt
void int_tmr0 (void)
{
unsigned char measure = 0;
unsigned char input_data;
copy = TM_TCSR0;
TM_TCSR0 = 0x00;
if (pre_wait != 0) {
        if (pre_count != 0)
                {
                pre_count --;
                }
```

```
        else {
                measure = 1;
                pre_count = pre_wait;
                } // of if pre_count
        } else {
        measure = 1;
        } // of if pre_wait
        if (counter == 0) {
                TM_TCR0 = 0x00; // Interrupt sperren
                measure = 0;
                done = 1;
                } // of if counter
        if (measure == 1) {
                input_data = CBTSU_Schalter;
                CBTSU_LED_Gruen = input_data;
                Data[counter].Switches = input_data;
                Data[counter].Time = (steps - counter); // JUST TEST!!!
                counter --;
        } // of if pre_wait
} // of int_tmr0

#pragma interrupt
void ISR_IRQ0(void)
{
unsigned char dummy;
unsigned char blubber;

    dummy = ISR;
    ISR = ISR & 0xfe;   // clears IRQ-flag
    blubber = CBTSU_Keyboard - 0xf0;
    switch (blubber)
    {
    case 13 : value = 0; break;
    case 0 : value = 1; break;
    case 1 : value = 2; break;
    case 2 : value = 3; break;
    case 4 : value = 4; break;
    case 5 : value = 5; break;
    case 6 : value = 6; break;
    case 8 : value = 7; break;
    case 9 : value = 8; break;
    case 10 : value = 9; break;
    case 12 : value = 10; break;       // A
    case 14 : value = 11; break;       // B
    case 15 : value = 12;
                        x = x / 10;
                        break;                                  // C
    case 11 : value = 13; break;                // D
    case 7 : value = 14;
                              Ok = 1;
                              break;
// E
     case 3 : value = 15; break;                // F
    } // of switch
    if (value <= 10) {
        x = (x * 10) + value;
    }
    CBTSU_DispInt(4,x);

} // of ISR_IRQ0

void timer_config (WORD steps_, unsigned long int time_)
```



```
{
        BYTE    cr;

        counter = steps_;
        if (time_ < 100)
        {
                cr = 2250 * (time_/1000);       // 2250 = Fquarz/Tpresc
                pre_count = 0;
        }
        else if ((time_ >= 100) && (time_ < 60000))
                {
                        cr = 225;                                       // entspricht
100ms
                        pre_count = time_ / 100;        // anzahl interrupts
                        pre_wait = pre_count;
                }

        TM_TCORA0       = cr;                                   // compare
reg laden, entspricht 100ms
        //TM_TCORA0     = 0x00;                                 // oberes
byte sicherheitshalber auf 0 init.
        TM_TCR0         = 0x00 + 0x4b;          // interrupt request,cleared
by compare match,presc_8192
   done = 0;
} // of timer_config

main(void)
{
unsigned long int time = 0;             // Var. zur Eingabe der gewünschten
Loggzeit
unsigned char read;
unsigned int choice;                                    // Var. zur Auswahl
des gewünschten Wertes

//Registrieren der Interrup-Service-Routinen
H8interrupt [IRQ_0] H8proc ISR_IRQ0;
H8interrupt [TM_CMIA0] H8proc int_tmr0;

//Initialisierung
LCD_Init();               // Init LCD
LCD_IWrite(0x01);         // LCD clear
PBDR|= 0x20;              // Display beleuchtung ein
steps = 0;
ISCR=0x01;
IER=0x01;

CBTSU_DispString(1,"-----Guten Tag!-----");
CBTSU_DispString(2,"Bitte jetzt         ");
CBTSU_DispString(3,"ueber das Keyboard  ");
CBTSU_DispString(4,"Config. vornehmen   ");
wait();
wait();
LCD_IWrite(0x01);
CBTSU_DispString(1,"Anzahl Messschritte ");
CBTSU_DispString(2," (E)nter (C)lear");
/* Beginn Keyboard-Scanner */
Ok = 0;
do {} while (Ok!=1);              // wait until E pressed!
steps = x;
LCD_IWrite(0x01);
CBTSU_DispInt(3,steps);           // Zeigt anzahl der Schritte auf Disp.
CBTSU_DispString(4,"Schritte");
```