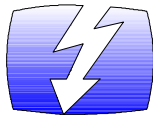

Prof. Dr.-Ing. Kühne **April 2005**
Bearb.:Dipl.-Ing. Sbieschni

Versuch 4: **Zeitmessung**

Zielstellung:

- Kennenlernen von 8-Bit-Mikrocontrollern,
- Arbeit mit einem Mikrocontrollerentwicklungssystem,
- Eingabe und Editieren von Quellprogrammen in der Programmiersprache C,
- Compilieren und Linken von Anwenderprogrammen,
- Download von Anwenderprogrammen in die Zielhardware,
- Starten und praktischer Test von Anwenderprogrammen,
- Kennenlernen der Interruptbearbeitung von Mikrocontrollern (Timerinterrupt),
- Betriebsweise von Zählern und Timern

Versuchsvorbereitung

Informieren Sie sich über die Leistungsmerkmale (Taktfrequenz, Speicherausstattung, Gehäuse des Controllers, Schnittstellen usw.) des verwendeten Mikrocontrollerboards auf der Basis des 8-Bit-Controllers Dallas DS80C320 (http://www.phytec.de/phytec/fuer_8-bit_module/rapid_development_kit_micromodul-8051.html)!

Informieren Sie sich mithilfe der Kurzbedienungsanleitung (https://www.hs-zigr.de/e-technik/stud/Lehrmatr/P_51_000_NEU.pdf) über die Vorgehensweise bei der Programmentwicklung in der Programmiersprache C mithilfe der Programmieroberfläche *µVision2* der Firma *KEIL*!

Informieren Sie sich mithilfe des „High Speed Microcontroller User Guides“ Section 9, Seite 107-113 (https://www.hs-zigr.de/e-technik/stud/ Lehrmatr/kuehne/hsmicro_ug.pdf) über die vorhandenen Interrupte und deren Abarbeitungsweise, insbesondere über die Timerinterrupte!

Mithilfe des „High Speed Microcontroller User Guides“ Section 11, Seite 121-131 (https://www.hs-zigr.de/e-technik/stud/ Lehrmatr/kuehne/hsmicro_ug.pdf) sind die Betriebsmodi der Timer des Mikrocontrollers zu ermitteln und entsprechend zu charakterisieren!

Welche Steuerregister des DS80C320 sind zum Betrieb eines interruptgesteuerten Timers zu programmieren!

Welche Vorteile bietet ein automatisches Reload der Zeitkonstante gegenüber einem mithilfe der Software programmierten Reload zu Beginn einer jeden Interruptserviceroutine in Hinblick auf die Genauigkeit der Zeitmessung?

Worin besteht bei Mikrocontrollern der Unterschied zwischen Timern und Countern?

Ermitteln Sie den Wert der Zeitkonstante eines aufwärts zählenden 16Bit-Zählers, der als Timer die quarzgesteuerte Taktfrequenz von 24 MHz des DS80C320 verarbeitet und der bei einem Übergang des Zählerstandes von FFFFH nach 0000H einen Interrupt auslöst, wobei eine Interruptserviceroutine mit einer Zykluszeit von 1 ms aufgerufen wird! Berücksichtigen Sie zusätzlich einen internen Vorteiler von 4!

Informieren Sie sich über die Betriebsweise von Quarzgeneratoren zur Taktsteuerung von Mikrorechnern, insbesondere über die Genauigkeit von Frequenz und des Tastverhältnis der bereitgestellten Taktfrequenz!

Worin besteht der Unterschied zwischen Unterprogrammen und Interruptserviceroutinen (Aufruf, Parameterübernahme, Parameterrückgabe)?

Charakterisieren Sie die Befehlszeile zur Vereinbarung einer Programmsequenz als Interruptserviceroutine (z.B. static void timer0(void) interrupt 0x01 using 1)!

Versuchsaufbau

Für die Steuerung der Zeitmessung sind zwei Umschalter erforderlich, die wahlweise auf Low- oder High-Pegel geschaltet werden können (vgl. Abb.1). Digitale Ausgänge werden nicht benötigt.

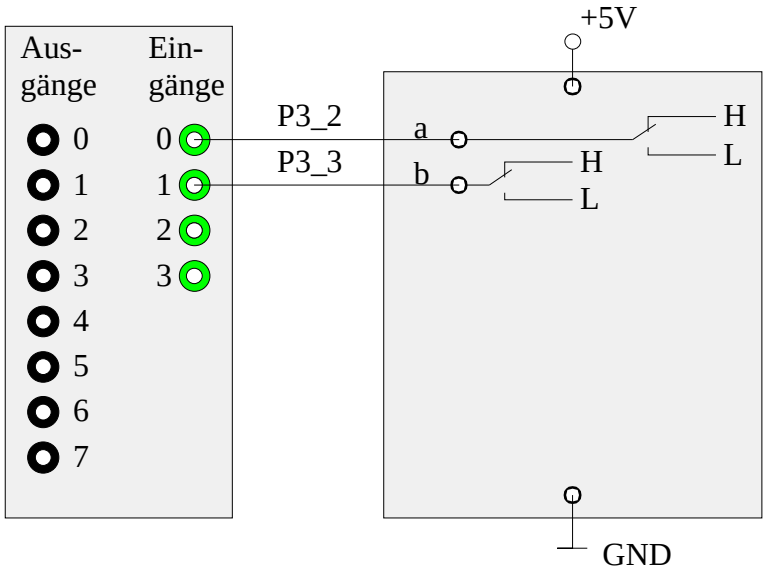

Abb.1: Verschaltung der digitalen Eingänge für die Zeitmessung

Aufgabenstellung:

Mithilfe eines C-Pogramms ist eine Zeitmessung zu programmieren, welche die Stunden, Minuten, Sekunden und Tausendstelsekunden seit dem Start der Zeitmessung liefert. Die aktuelle Zeit ist mit einem printf-Befehl mithilfe einer formatierten Ausgabe in der Form *hh : mm : ss : ttt* auf dem Monitor anzuzeigen wobei *hh* eine zweistellige Stunden-, *mm* eine zweistellige Minuten-, *ss* eine zweistellige Sekunden- und *ttt* eine dreistellige Tausendstelsekundenanzeige ist.

Die Zeitmessung wird mithilfe eines HIGH-Pegels am Eingangsbit 1 (P3_3 des Mikrocontrollersystems) gestartet. Bei einem LOW-Pegel an diesem Pin erfolgt ein Rücksetzen der Zeitmessung.
Mithilfe eines HIGH-Pegels am Eingangsbit 0 (P3_2 des Mikrocontrollersystems) soll die Zeitmessung zusätzlich angehalten werden, bei einem LOW-Pegel soll sie fortgesetzt werden.
In der nachfolgenden Tabelle sind noch einmal die Funktionen der beiden Eingabebits P3_2 und P3_3 zusammengefasst.

Digitaler Eingang 0 (P3_2)	Digitaler Eingang 1 (P3_3)	
0 oder 1	0	Rücksetzen der Zeitmessung auf den Wert Null und Warten auf die Freigabe
0	1	Freigabe der Zeitmessung, ständiges Aktualisieren des Zeitwertes auf dem Monitor
1	1	Anhalten der Zeitmessung und Einfrieren des Zeitwertes

Zur Realisierung der Zeitmessung ist eine Interruptserviceroutine im Millisekundentakt aufzurufen und darin die entsprechenden Tausendstelsekunden- Sekunden-, Minuten-, und Stundenwerte entsprechend zu aktualisieren.

Die Ausgabe des Zeitwertes auf dem Monitor erfolgt zyklisch in eine Endlosschleife im Hauptprogramm main(). Das Einlesen der binären Eingangsinformationen zum Start, Anhalten und Rücksetzen der Zeitmessung kann wahlweise in der Interruptserviceroutine und/oder in der Endlosschleife des Hauptprogramms erfolgen.

Hinweise zur Erstellung des Programms:

- Vor dem Aufruf der Programmieroberfläche µVision2 sind alle Dateien der Directory C:\Programme\ phybasic\um8051\DEMOS\Keil\Versuch_4 in eine noch zu erstellende Directory C:\Programme\phybasic\um8051\ DEMOS\Keil\KK_I_P zu kopieren (KK ist dabei die Nummer des Matrikels, I die der Versuchsgruppe und P die Versuchsnummer)! Die kopierten Dateien gehören zu einem bereits vordefinierten Projekt, welches unter anderem die Quellcodedatei Uhr.c besitzt. In diese Datei ist der vorbereitete Anwendercode hinzuzufügen! Nach einem Start

der Programmieroberfläche µVision 2 erfolgt das Öffnen des Projektes mithilfe von „Project - OpenProject“ und ein Laden der Datei Versuch_2.uv2 aus der Directory C:\Programme\phybasic\um8051\ DEMOS\Keil\KK_I_P.

Die zu vervollständigenden Projektdateien sind auch im Internet in der Datei Zeitmessung.zip verfügbar!

- Durch ein Aktivieren der Datei Uhr.c aus der Liste der Projektdateien per Mausklick im Dateifenster von µVision2 wird diese automatisch in den Editor geladen!

- Der Programmcode für das Aktualisieren der Stunden-, Minuten-, Sekunden- und Tausendstelsekundenwerte ist in das bereits vorbereitete, interruptgesteuerte Unterprogramm „static void timer0(void) interrupt 0x01 using 1“ einzufügen! Dieses Unterprogramm wird bei einer fehlerfreien Zuweisung der Zeitkonstante zyklisch im Abstand von 1ms mithilfe eines Timerinterrupts aufgerufen.

- Der aktuelle Zeitwert ist mit einer formatierten Ausgabe in dem vorab erwähnten Format zyklisch auf dem Monitor auszugeben!

- Die 16Bit-Zeitkonstante des Zählers mit den beiden 8-Bit-Bytes TL0 (niederwertiger Teil) und TH0 (höherwertiger Teil ist beim Eintritt in die Interruptserviceroutine am Anfang immer neu zu laden und so ein neues Zeitintervall zu starten.

- Der aktuelle Zeitwert ist in einer Struktur vom Typ time mit den Komponenten Stunde, Minute, Sekunde und Tausendstelsekunde (alle vom Typ Integer) zu speichern! Eine Aktualisierung des Zeitwertes erfolgt durch eine Änderung der Komponenten der Struktur!

Die Programmieraufgabe ist vollständig umgesetzt, wenn mit dem Eingabebit P3_2 die Zeitmessung aktiviert und angehalten und mithilfe des Eingabebits P3_3 die Zeitmessung rückgesetzt werden kann sowie der aktuelle Zeitwert im geforderten Format mit dem Stunden-, Minuten-, Sekunden- und Tausendstelsekundenwert angezeigt wird.

Hinweis:
Bei den Internetadressen, die auf Ziele außerhalb der Hochschule verweisen kann es sein, dass diese unter Umständen nicht mehr gültig sind. Dann ist unter Verwendung der in der Regel unveränderten Domain-Adresse die gesuchte Internetseite selbständig aufzufinden!

Listing des Programms Uhr.c

```
#include <reg320.h>          /* include 8051 header file      */
#include <stdio.h>           /* Standard I/O functions        */

sbit P1_0 = P1^0;
sbit P1_1 = P1^1;
sbit P1_2 = P1^2;
sbit P1_3 = P1^3;
sbit P1_4 = P1^4;
sbit P1_5 = P1^5;
sbit P1_6 = P1^6;
sbit P1_7 = P1^7;

sbit P3_2 = P3^2;
sbit P3_3 = P3^3;
sbit P3_4 = P3^4;
sbit P3_5 = P3^5;

/*Deklaration einer Struktur für die Eingabebits*/
struct bitfeld_in
     {
     unsigned bit_in_nr0 :1;
     unsigned bit_in_nr1 :1;
     unsigned bit_in_nr2 :1;
     unsigned bit_in_nr3 :1;
     unsigned dummy :12;
    }flags_in;

/*Deklaration einer Struktur für die Ausgabebits*/
struct bitfeld_out
```

```
      {
      unsigned bit_out_nr0 :1;
      unsigned bit_out_nr1 :1;
      unsigned bit_out_nr2 :1;
      unsigned bit_out_nr3 :1;
      unsigned bit_out_nr4 :1;
      unsigned bit_out_nr5 :1;
      unsigned bit_out_nr6 :1;
      unsigned bit_out_nr7 :1;
      unsigned dummy :8;
     }flags_out;

/*Deklaration einer union für die Eingabebits */
union flagss_in
      {
      struct bitfeld_in bits_in;
      unsigned int bit_wert_in_uint;
      char bit_in[2];
      }input_bit;

/*Deklaration einer union für die Ausgabebits */
union flagss_out
      {
      struct bitfeld_out bits_out;
      unsigned int bit_wert_out_uint;
      char bit_out[2];
      }output_bit;

struct time
    {
    int tausendstel;
    int sekunde;
    int minute;
    int stunde;
   }time_1;

void init_interrupt(void)
   {
   ET0=1;                       /* Freigabe Interrupt-Enable-Bit IE.1*/
   EA=1;                        /* globales Interrupt enable Bit setzen*/
   TMOD=TMOD&0xFB;      /* Rücksetzen von TMOD2 select internal clock*/
   TMOD=TMOD|0x01;      /* Rücksetzen TMOD.0 - M0 */
   TMOD=TMOD&0xFD;      /* Setzen TMOD.1 - M1 */
   CKCON = CKCON|0X08;    /* Vorteiler 4 setzen  CKCON.3 = 1
                             Vorteiler 12 setzen CKCON.3 = 0*/
   TL0 = 0X00;                  /* count-value low*/
   TH0 = 0X00;                  /* count-value high*/
   TF0=0;                       /* Timer0 Overlowflow TCON.5*/
   TR0=1;                       /* runs Timer0 TCON.6*/
   }

void init()
    {
    TI=1;
    ES0=0;                      //SerialInterrupt Disabled
    }

void control()
    {
    if (SBUF0==0x1b)         //Stop bei ESC
        {
        printf("\n SYSTEM GESTOPPT");
        ES0=1;                  //SerialInterrupt Enabled
        while(1);               //RESET um neu zu beginnen
```

```
        }
       RI=0;
    }

static void timer0(void) interrupt 0x01 using 1
   {

/*   hier den Kode für die Uhr eintragen */
   }

void main (void)
    {
    init_interrupt();
    init();
    …..
    while (1)   /* loop forever                    */
       {
       /* Zuweisung der Eingabebits auf die Struktur */
       flags_in.bit_in_nr0 = P3_2;
       flags_in.bit_in_nr1 = P3_3;
       flags_in.bit_in_nr2 = P3_4;
       flags_in.bit_in_nr3 = P3_5;

      /* Zuweisung der Struktur auf die union und Ausblenden von nicht benötigten Bits */
       input_bit.bits_in = flags_in;
       input_bit.bit_wert_in_uint &= 0X000F;

/*****************************************************************
hier den Programmkode für die Ein- und Ausgaben (printf getkey usw.) eintragen
*****************************************************************/

        control();
       }
    }
```