**Prof. Bischoff** **21. Juli 2021**
Bearb.:Dipl.-Ing. (FH) E. Schreiter

# Versuch 3: **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**

Für die Versuchsdurchführung ist es hilfreich, eine knapp gehaltene Befehlsreferenz für die Programmiersprache C, das Datenblatt des eingesetzten Controllers Dallas 80C320 und eine ASCII Tabelle zur Verfügung zu haben.

Aus der Vorlesung und Übung sollte der Umgang mit der Entwicklungsumgebung Keil µVision bekannt sein.

Bereiten Sie sich auf die Beantwortung von Fragen aus der separat verfügbaren Datei „Beispielfragen Praktikum Mikrorechentechnik.pdf“ vor.

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?

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 den internen Vorteiler von 4!

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.

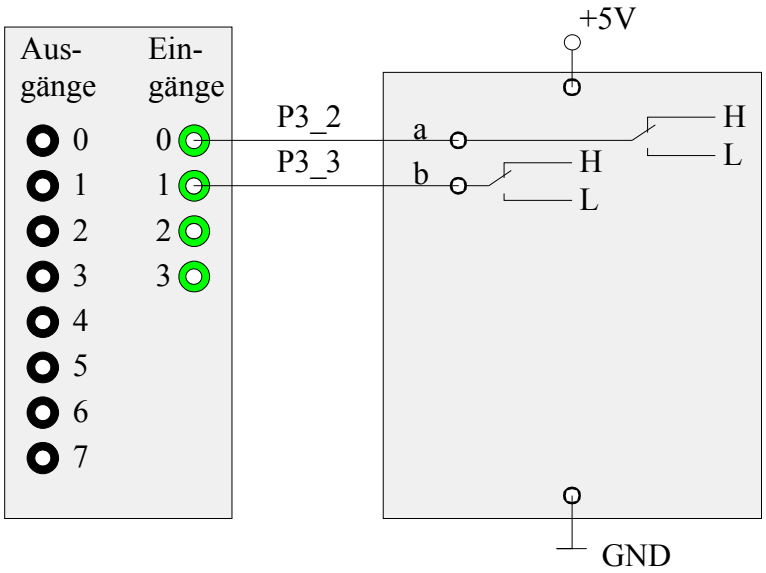


Abb.1: Verschaltung der digitalen Eingänge für die Zeitmessung

**Aufgabenstellung:**

Mithilfe eines C-Programmes ist eine Zeitmessung zu programmieren, die Stunden, Minuten, Sekunden und Tausendstelsekunden seit dem Start der Zeitmessung anzeigt. Die aktuelle Zeit ist mit einem printf()-Befehl in der Art
„*hh : mm : ss : ttt*“ auf dem Monitor anzuzeigen( *hh* eine zweistellige Stunden-, *mm* zweistellige Minuten-, *ss* eine zweistellige Sekunden- und *ttt* eine dreistellige Tausendstelsekundenanzeige).

Details zur Umsetzung:

1. Es ist eine Zeitmessung auf dem Monitor mit printf() darzustellen.
2. Ein Schalter am Eingang 0 startet (logisch 1) bzw. setzt den Zähler zurück (logisch 0).
3. Ein Schalter am Eingang 1 hält den Zähler an (logisch 1) bzw. lässt ihn zählen (logisch 0).
4. Die Darstellung am PC soll immer innerhalb einer Zeile bleiben, d.h. keinen mit Zeilenvorschub enthalten.
5. Die Zeit soll global als Variable verfügbar sein und als struct definiert werden
6. Die Komponenten der Zeit (Stunde, Minute, Sekunde, Millisekunde) sind vom Typ unsigned int anzulegen.
7. Ein Interrupt soll mit 1ms Zykluszeit aufgerufen werden. Die Funktion dazu wird mit „static void timer0(void) interrupt 0x01 using 1“ deklariert.
8. In der Interruptfunktion wird die Variable für die Zeit entsprechend den Eingängen verändert
9. In der Hauptfunktion soll die Zeit ausgegeben werden.

**Listing des Programms versuch3.c**

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

/*************************
 ** Hier eigenen globale
 ** Variablen definieren
 *************************/

static void my_timer0(void) interrupt 1 /*Timer 0 Overflow*/  using 1 {
       /*************************
        ** Hier eigenen Quelltext
        ** einfügen
        *************************/
}
```

```
void main (void) {
/*************************
 ** Hier eigene Variablen
 ** anlegen etc.
 *************************/
init ();
init_interrupt();

printf(" Praktikumsversuch 3: Zeitmessung\n\n");

/*-----------------------------------------------
Note that an embedded program never exits (because
there is no operating system to return to). It
must loop and execute forever...
-----------------------------------------------*/

while (1) {

        /*************************
         ** Hier eigenen Quelltext
         ** einfügen
         *************************/
        }

}
```