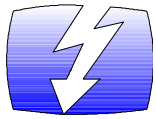

Prof. Dr.-Ing. Kühne **April 2005**
Bearb.:Dipl.-Ing. Sbieschni

Versuch 3: **Digital-Analog-Wandler**

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,
- Realisierung der Ansteuerung von DA-Wandlern mithilfe von Mikrocontrollern,
- Kennenlernen der Interruptbearbeitung von Mikrocontrollern (Timerinterrupt)

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 mit 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!

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 0.5 ms aufgerufen wird! Berücksichtigen Sie zusätzlich einen internen Vorteiler von 4!

Welche Möglichkeiten des Ladens der Zeitkonstanten lassen die Timer des DS80C320 für die Abarbeitung zeitlich äquidistanter Interruptzyklen zu?

Welche Wertebereiche der digitalen Ausgabevariablen ergeben sich bei 8-Bit, 10-Bit, 12-Bit und 16 Bit D-A-Wandlern?

Worin unterscheidet sich die Ansteuerung von D-A- und A-D-Wandlern (Steuersignale, zeitliche Abfolge, ...)?

Informieren Sie sich mithilfe des Datenblatts des verwendeten D-A-Wandlers AD557 der Firma ANALOG DEVICES (http://www.analog.com/Analog_Root/productPage/productHome/0%2C2121%2CAD557%2C00.html) über dessen elektrische Parameter, das Pinlayout, die Funktionsweise und die Betriebsmodi!

Versuchsaufbau

Die 8 Ausgangsbits des Mikrocontrollersystems P1_0 ... P1_7 sind entsprechend der folgenden Abbildung mit den digitalen Eingängen eines D-A-Wandlers zu verbinden. Der D-A-Wandler wird mit seiner eigenen Referenzspannung REF_OUT von 2.5V gespeist (Verbinden von REF_OUT mit REF_IN). Die Referenzausgangsspannung des D-A-Wandlers hat einen Wert von 2.5V, ein digitaler Eingabewert von 0XFFH entspricht 2.5·(1-1/256)V, ein Wert von 0X00H entspricht 0V. Da nur ein Teilnehmer an die Ausgangsbits des Mikrocontrollers geschaltet wird, kann der CE-Eingang ständig aktiv geschaltet sein und auf Massepotential liegen. Die Ausgangsspannung des D-A-Wandlers am Pin OUT ist zu oszillographieren.

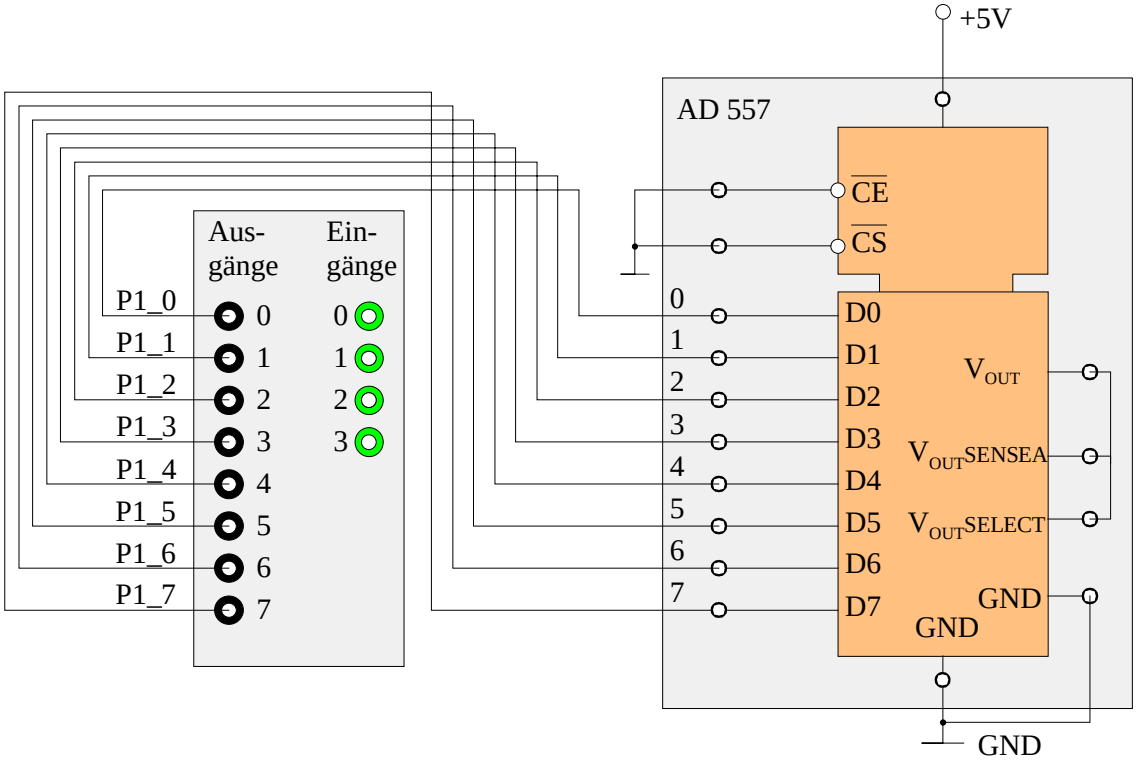

Abb.1: Verschaltung von digitalen Ausgängen und D-A-Wandler

Aufgabenstellung:

Mithilfe eines C-Programms ist ein programmierbarer Rechteckgenerator zu erzeugen, wobei dessen Frequenz und Amplitude einstellbar sind. Dazu soll mit der getkeyy()-Funktion eine Zahl von 1 bis 9 von der PC-Tastatur eingelesen und in das Mikrocontrollerboard übertragen werden. Diese Zahl im char-Format - getkeyy() liefert einen Funktionswert im char-Format - ist in eine Integer-Zahl (Variablenname „Wert“) umzuwandeln und dient als Teiler für die Zeitkonstante eines Zeitgebers. Wird die Zahl 1 eingegeben, so ist eine Frequenz von 1 kHz und einem Tastverhältnis von 1, bei Eingabe der Zahl 2 von 0.5 kHz und einem Tastverhältnis von 1 usw. zu erzeugen (vgl. Abb. 2). Die Zeitkonstante wird in die vordeklarierten Variablen TL0 und TH0 geladen, wobei TL0 die 8 niederwertigen und TH0 die 8 höherwertigen Bits des 16Bit-Zeitkonstantenwertes beinhalten. Diese 16Bit-Zeitkonstante dient als Startwert eines Timers. Das Programm ist interruptgesteuert entsprechend des Wertes der Zeitkonstante im 0.5ms, 1ms, 2ms ... Raster aufzurufen und bei jedem Aufruf die digitalen Ausgabewerte neu zu ermitteln. Innerhalb der Interrupt-

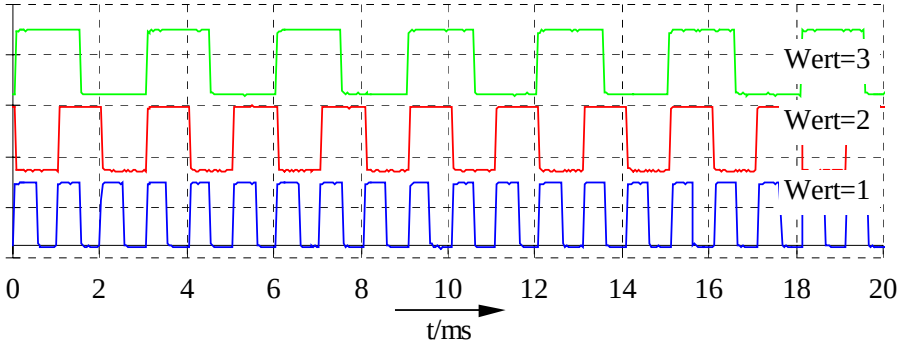

Abb.1: Ausgangsspannungen des D-A-Wandlers

serviceroutine sind entsprechende Ausgaben auf die digitalen Ausgänge P1_0 ... P1_7 durchzuführen und der D-A-Wandler mit Digitalwerten zu speisen. Mithilfe eines Oszillographen sind Amplitude und die Frequenz der Rechteckschwingung zu überprüfen (vgl. Abb.1).
Die vom D-A-Wandler erzeugte Ausgangsfrequenz ist hinsichtlich ihrer Anstiegs- und Abfallzeit mithilfe des Oszilloskops zu analysieren und die Wandlungsgeschwindigkeit des verwendeten Schaltkreistyps zu ermitteln!

Hinweise zur Erstellung des Programms:

- Vor dem Aufruf der Programmieroberfläche µVision2 sind alle Dateien der Directory C:\Programme\phybasic\um8051\DEMOS\Keil\Versuch_3 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 DA-Wandler.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 AD-Wandler.zip verfügbar!

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

- Der Programmcode für den Rechteckgenerator 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 0.5ms, 1ms, 2ms usw. entsprechend des geladenen Zeitkonstantenwertes mithilfe eines Timerinterrupts aufgerufen.

- Der Wert der Zeitkonstante wird bei einem erneuten Aufruf der Interruptserviceroutine nicht automatisch in den Timer geladen, sondern muss zu Beginn immer erneut in die beiden Register TL0 und TH0 geladen werden.

- Beim Eintreten in die Interruptserviceroutine wird das Interruptrequestbit für den nächsten Zeitinterrupt automatisch gelöscht und braucht vom Anwender nicht rückgesetzt werden.

- Die Zuordnung von Interruptprioritäten kann unberücksichtigt bleiben, da nur ein Interrupt verwendet wird.

- Zum Einlesen des Zeichens von Tastatur ist nicht die Standard-getkey-Funktion der stdio-Library zu verwenden sondern eine bereits definierte Funktion getkeyy() vom Typ char! Zur Verwendung der getkeyy()-Funktion sind auch die Hinweise in https://www.hs-zigr.de/e-technik/stud/Lehrmatr/P_51_000_NEU.pdf zu beachten!

- Der ASCII-Code des von der Tatstatur eingelesenen Zeichen ist in einer char-Variable abzuspeichern!

- Die von der PC-Tastatur eingelesenen Zahl ist zusätzlich mithilfe des printf-Befehls auf dem Bildschirm anzuzeigen!

- Für Wertzuweisungen auf die Ausgabeports P1_0 ... P1_7 sind wenn möglich die im Quellkode bereits vordeklarierten und vordefinierten Strukturen und Units zu verwenden! Dabei ist der Digitalwert zur Vermeidung von Wandlungs- bzw. Ausgabefehlern ganzheitlich mit einer Schreiboperation als Char-Variable auf das Ausgabeport P3 zu schreiben!

- Das Variation bzw. das wiederholte Einlesen des Teilers erfolgt in einer Endlosschleife im Hauptprogramm. Nach erfolgter Eingabe ist die Zeitkonstante anschließend neu zu berechnen und in eine globale Variable zu schreiben. Auf diese globale Variable greift dann die Interruptserviceroutine zu und übernimmt den Wert der neuen Zeitkonstante.

- Die Amplitude der Rechteckfrequenz ist durch Zuweisen eines Wertes im Bereich von 0X00H bis 0XFFH mithilfe einer define-Anweisung am Anfang des Programms festzulegen.

Die Programmieraufgabe ist vollständig umgesetzt, wenn die Zahlen 0 bis 9 von der Tastatur eingegeben, über die serielle Schnittstelle in das Mikrocontrollerboard übertragen, zusätzlich auf dem Monitor gespiegelt und weiterhin eine entsprechende Modifizierung der Zeitkonstante des Timers und somit eine Variation der Rechteckfrequenz des Generators erfolgt.

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!

Quellkode des Programms DA-Wandler.c

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

sfr CKCON  = 0x8E;
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;

struct bitfeld_in              /*Deklaration einer Struktur für die Eingabebits*/
      {
      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;

struct bitfeld_out             /*Deklaration einer Struktur für die Ausgabebits*/
      {
      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;

char getkeyy()
     {
     char c;
```

```
    while (!RI);
    while (SBUF==17) RI=0;
    c = SBUF;
    RI = 0;
    return (c);
    }

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 Programmkode für den Rechteckgenerator 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 eintragen printf getkeyy usw.
```

```
**********************************************************/
        control();
        }
    }
```