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

Versuch 2: **Sieben-Segment-Anzeige**

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 der praktische Test von Anwenderprogrammen,
- Durchführung einer indirekten Adressierung mithilfe von Zeigeroperationen

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*!

Mithilfe des „High Speed Microcontroller User Guides“, Seite 116-118 (https://www.hs-zigr.de/e-technik/stud/Lehrmatr/kuehne/hsmicro_ug.pdf) ist der Schaltungsaufbau der digitalen Ein-/Ausgänge der 8-Bit-Ports P1 und P3 zu ermitteln? Wieso können die Ports P1 und P3 ohne eine Zusatzbeschaltung wahlweise als Ein- und Ausgänge verwendet werden?

Worin besteht das Wesen von Strukturen und Units (Varianten) in der Programmiersprache C? Warum ist es günstig, die Eingabebits P3.2 ... P3.5 und die Ausgabebits P1.0 ... P1.7 zu einer Struktur zusammenzufassen und nachfolgend ein Zuweisen dieser Struktur auf eine Unit durchzuführen?

Worin besteht das Wesen einer indirekten Adressierung in der Programmiersprache C und welche Zeigeroperatoren gibt es?

Wie kann mithilfe von Zeigeroperationen effektiv auf die Elemente einer Tabelle im Speicher zugegriffen werden?

Ermitteln Sie die Größe des Vorwiderstandes R_V entsprechend der nachfolgenden Abb. 2, wenn die Sieben-Segment-Anzeige von einem Low-Power-Schottky-TTL Baustein gespeist wird, die Flussspannung U_F der Diode 1,2 V beträgt und ein Diodenstrom von I_D = 5 mA einzuprägen ist!

Informieren Sie sich über den Aufbau von 7-Segment-Anzeigen, insbesondere über deren Anschlussbelegungen (gemeinsame Kathode, gemeinsame Anode)!

Ermitteln Sie mithilfe der Schaltkreisdokumentation des Mikrocontrollers DS80C320 (http://www.hs-zigr.de/e-technik/stud/Lehrmatr/kuehne/Mikrorechentechnik/ds80c320.pdf) die Adressen der für die Ein- und Ausgabe verwendeten internen Register P1 und P3 des Prozessors!

Was verstehen Sie unter bitadressierbaren Variablen?

Wie groß ist der Ruhestromverbrauch des Controllers bei einer Taktfrequenz von 24 MHz (vgl. Seite 22 http://www.hs-zigr.de/e-technik/ stud/Lehrmatr/kuehne/Mikrorechentechnik/ds80c320.pdf)?

Was verstehen Sie unter einem gemultiplexten und nicht gemultiplexten Daten- und Adressbus und welche Betriebsart verwendet der Prozessor DS80C320 (vgl. http://www.hs-zigr.de/e-technik/stud/Lehrmatr/kuehne/Mikrorechentechnik/ds80c320.pdf)?

Versuchsaufbau

Die Ausgangsbits des Mikrocontrollersystems P1_0 ... P1_7 mit den Anschlüssen 0, 1, 2, 3, 4, 5 und 6 sind entsprechend der nachfolgenden Abb. 1 mit den Leuchtbalken a, b, c, d, e, f und g der Sieben-Segment-Anzeige zu verbinden, das Ausgabebit P1_7 des Mikrocontrollers bleibt unbeschaltet.

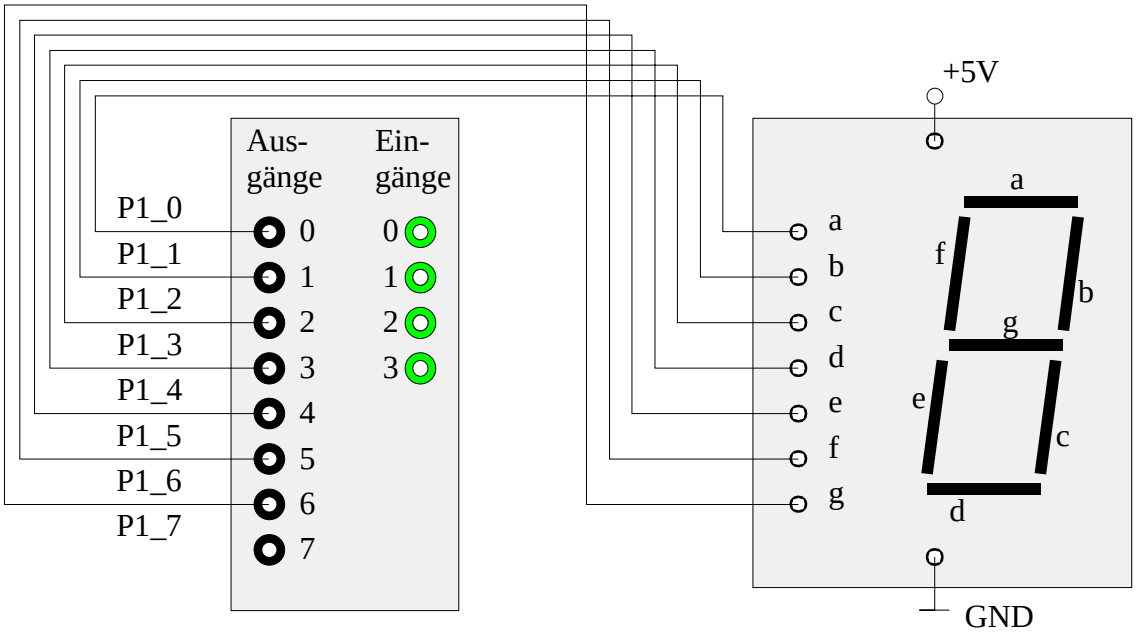

Abb.1: Verdrahtung der Ausgabeleitungen mit der Sieben-Segment-Anzeige

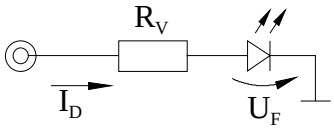

Abb2: Ersatzschaltung eines Leuchtbalkens der Sieben-Segment-Anzeige

Die Ersatzschaltung des an die Ausgabeleitungen angeschlossenen Verbrauchers eines Leuchtbalkens kann mithilfe einer Leuchtdiode und eines Vorwiderstandes dargestellt werden und ist in Abb. 2 dargestellt. Bei der verwendeten Sieben-Segment-Anzeige existiert eine gemeinsame Kathode.

Aufgabenstellung:

Es ist ein Programm zu entwickeln, welches mithilfe der getkey()-Funktion eine Zahl von 0 bis 9 von der Tastatur einliest, eine Umwandlung in den Sieben-Segment-Code durchführt und anschließend eine Ausgabe auf der Sieben-Segment-Anzeige vornimmt. Die mit der PC-Tastatur eingegebene Zahl ist zusätzlich auf dem Monitor anzuzeigen!

Hinweise zur Erstellung des Programms:

- Vor dem Aufruf der Programmieroberfläche µVision2 sind alle Dateien der Directory C:\Programme\phybasic\um8051\DEMOS\Keil\Versuch_2 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 7_Segmentanzeige.c besitzt. In diese Datei ist der 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 Sieben-Segment-Anzeige.zip verfügbar!

- Durch ein Aktivieren der Datei 7_Segmentanzeige.c aus der Liste der Projektdateien per Mausklick im Dateifenster von µVision2 wird diese automatisch in den Editor geladen.

- Der Programmcode ist in eine Endlosschleife im Hauptprogramm main() einzufügen und wird dort zyklisch, entsprechend der Laufzeit des Programms wiederholt abgearbeitet
  (vgl. Seite 1 https://www.hs-zigr.de/e-technik/stud/Lehrmatr/ P_51_000_NEU.pdf).
- Zum Einlesen des Zeichens von Tastatur ist nicht die Standard-getkey-Funktion der stdio-Library zu verwenden sondern eine bereits vordefinierte und im Quellkodeprogramm 7_Segmentanzeige.c vorhandene 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.

- Es ist unter Zuhilfenahme einer ASCII-Tabelle eine Transformation der ASCII-Codes der eingelesenen Zahl (möglich sind Werte von 1 bis 9) in den Sieben-Segment-Code durchzuführen. Die Umrechnungsvorschrift ist dabei effektiv zu gestalten und stellt den Kern des zu entwickelnden C-Programms dar.

- Die eingelesene Zahl ist zusätzlich mithilfe des printf-Befehls auf dem Bildschirm anzuzeigen.

- Für Wertzuweisungen auf die Ausgabeports P1_0 ... P1_7 sind die im Quellkode bereits vordeklarierten und vordefinierten Strukturen und Units zu verwenden. Es ist jedoch auch möglich, jedes Bit des auszugebenden Sieben-Segment-Codes einzeln zu verarbeiten.

Die Programmieraufgabe ist vollständig umgesetzt, wenn die Zahlen 0 bis 9 fortlaufend von der Tastatur eingegeben und über die serielle Schnittstelle in das Mikrocontrollerboard übertragen sowie auf der Sieben-Segment-Anzeige und dem Monitor fehlerfrei angezeigt werden

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 7_Segmentanzeige.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;
      }input_bit;

/*Deklaration einer union für die Ausgabebits */
union flagss_out
      {
      struct bitfeld_out bits_out;
      unsigned int bit_wert_out_uint;
      }output_bit;

char getkeyy()
     {
     char c;
     while (!RI);
     while (SBUF==17) RI=0;
     c = SBUF;
     RI = 0;
     return (c);
     }

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;
    }

void main (void)
    {
    …..
    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 Anwenderkode hinzufügen  ***********************
*****************************************************************************/

      /* Zuweisung der union auf die struktur und Ausblenden von nicht benötigten Bits */
      output_bit.bit_wert_out_uint &= 0X00FF;
      flags_out = output_bit.bits_out;
      /* Zuweisung der Struktur auf die Ausgabebits */
      P1_0 = flags_out.bit_out_nr0;
      P1_1 = flags_out.bit_out_nr1;
      P1_2 = flags_out.bit_out_nr2;
      P1_3 = flags_out.bit_out_nr3;
      P1_4 = flags_out.bit_out_nr4;
      P1_5 = flags_out.bit_out_nr5;
      P1_6 = flags_out.bit_out_nr6;
      P1_7 = flags_out.bit_out_nr7;
      control();
     }
  }
```