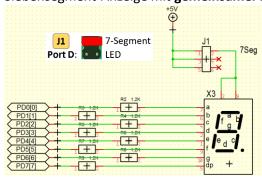
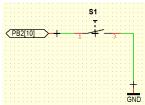


## Projekt: Countdown-Zähler mit Siebensegment-Anzeige

Für ein Quizspiel soll ein Countdown-Zähler entworfen werden, der vom Quizmaster (mit S1=Start) gestartet wird und dann von 9 bis 0 im Sekundentakt herunterzählt. Die Anzeige erfolgt über eine Siebensegment-Anzeige mit **gemeinsamer Anode**.





# Arbeitsauftrag

- Entwerfen Sie mit Hilfe der nebenstehenden Tabelle die Siebensegment-Codes. Jede Ziffer 0...9 wird dabei durch eine eigene Hexadezimalzahl repräsentiert.
- Erstellen Sie zunächst die Initialisierung für die Siebensegment-Anzeige und den Taster "Start".

#define	SiebenSegment	_PORTD_
#define	Taster	_PORTB_
#define	Start	2

 Programmieren Sie die Funktionalität des Start-Tasters in der Endlosschleife und

Port		PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	Codes
Se	gment	dp	gg	f	е	d	С	b	а	hex
0		1	1	0	0	0	0	0	0	0xc0
1		1	1	1	1	1	0	0	1	0xf9
2										
3										
4										
5										
6										
7										
8										
9										

überlegen Sie sich, wie Sie möglichst effektiv die Siebensegment-Codes nacheinander auf den PORTD (SiebenSegment) ausgeben können. Der Countdown wird mit dem Taster gestartet. Zwischen der Anzeige von zwei Zahlen soll jeweils eine Sekunde gewartet werden. Ist der Countdown abgelaufen, kann er durch erneuten Tasterdruck wieder gestartet werden.

• Testen Sie das Programm auf der Mikrocontrollerhardware und Erstellen Sie ein Versuchsprotokoll.



#### Mikrocontroller-Labor

Datum: 17.10.2016 2\_2\_Countdown\_Siebensegm

Siebensegment-Anzeige (Byte I/O)

2.2.2



Eine elegante Methode um eine Aktion auf mehrere Variablen gleichen Typs durchzuführen, ist die Verwendung von Arrays (Datenfeldern).

In Arrays (Feldern) werden mehrere Variablen gleichen Typs zusammengefasst, die mit einem Namen angesprochen werden.

Bsp.: Definition eines Array aus 4 Byte-Werten (Es werden 4 Byte im RAM reserviert)

```
uint8_t feld[4];
                                                    feld[0] | feld [1] | feld [2] | feld [3] |
```

Häufiger Fehler: Bei der Definition des Arrays wird die Anzahl der Elemente eingesetzt. Die Nummerierung beginnt aber bei 0.

Arrays können bei der Definition initialisiert werden:

```
zahl[0]
                                                        zahl[1]
                                                                 zahl[2]
                                                                          zahl[3]
uint16_t zahl[] = {43,128,23,492};
                                                = 43
                                                         = 128
                                                                  = 23
                                                                          = 492
```

Tatsächlich werden hier 8 Byte (4 \* 16Bit) reserviert und währende der Laufzeit des Programms im data-Segment angelegt. Damit können Array-Elemente vom Programm verändert werden. Für die Definition konstanter Arrays, werden diese im Code-Segment angelegt.

```
const uint16_t zahl[] = { 43,128,23,492};
```

Der Vorteil eines Arrays ist, dass es sich sehr einfach in Zählschleifen verarbeiten lässt.

Bsp.:

```
for(n=0;n<=3;n++)
  feld[n] = n;
```

### Arbeitsaufträge

Ändern Sie Ihr Programm für den Countdown-Zähler so ab, dass die 7-Segment-Codes in einem

konstanten Array abgelegt werden:

```
const uint8_t seg7code[] = { 0xc0,0xf9, ... };
```

Lesen Sie das Array mit einer for-Schleife rückwärts aus. Damit die Bedingung n >= 0 falsch werden kann, muss die Indexvariable negativ werden können. Die Deklaration muss als vorzeichenbehafteter Integer erfolgen: int8\_t n;

Wenn der Zähler auf O gezählt hat, soll mit dem Speaker (PORTB.3) eine Sekunde ein Alarmton mit 1000Hz ausgegeben werden. Auch dies können Sie mit einer for-Schleife realisieren. Für die Pulsdauer und Pulspause des

Rechtecksignals verwenden Sie die Funktion:

delay\_100us(5);

func main: endlos warten bis Taster gedrückt n = 9solange n >= 0 seg7code[n] ausgeben 1s warten n = n - 1

PORTB

J2 Speaker:

#define B3

#define Speaker



#### Mikrocontroller-Labor

Name: Rahm
Datum: 17.10.2016
2\_2\_Countdown\_Siebensegment.docx

Siebensegment-Anzeige (Byte I/O)

2.2.3

### **Ergebnis**

		Port	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	Codes
0	Se	gment	Dp	80	f	е	d	С	b	а	hex
	0		1	1	0	0	0	0	0	0	0xc0
	1		1	1	1	1	1	0	0	1	0xf9
	2		1	1	1	1	1	0	0	1	0xa4
	3	8.	1	0	1	1	0	0	0	0	0xb0
4	4	9.	1	0	0	1	1	0	0	1	0x99
	5	8.	1	0	0	1	0	0	1	0	0x92
	6	8.	1	0	0	0	0	0	1	0	0x82
	7	<b>.</b>	1	1	1	1	1	0	0	0	0xf8
-	8	8.	1	0	0	0	0	0	0	0	0x80
	9	9.	1	0	0	1	0	0	0	0	0x90

```
#include "controller.h"
                    _PORTD_
#defineSiebenSeg
                    _PORTB_
#define Taster
#define Start
#define Speaker
                    _PORTB_
#define B1
uint8_t seg7code[] = { 0xc0,0xf9,0xa4,
                        0xb0,0x99,0x92,0x82,0xf8,0x80,0x90 };
void setup (void)
{ /* Initialisierungen */
  byte_init(SiebenSeg,OUT);
 byte_write(SiebenSeg,0xff);
                                 // Anzeige dunkelschalten
 bit_init(Taster,Start,IN);
  bit_init(Speaker,B1,OUT);
// Funktion main()
void main(void)
  int8_t n;
  uint16_t i;
 uint8_t beep = 0x01;
 setup();
 while(1)
                                        // Endlosschleife loop()
    while(bit_read(Taster,Start)==1);  // Warten bis Taster betätigt
    for(n=9;n>=0;n--)
     byte_write(SiebenSeg,seg7code[n]);
      delay_ms(1000);
    }
    for(i=0;i<2000;i++)</pre>
     bit_write(Speaker,B1,beep);
      delay_100us(5);
      beep=~beep;
 }
}
```