

Auf dem Multifunctionshield befinden sich 4 Siebensegmentanzeigen: Digit 0 .. Digit 3.

**Siebensegmentanzeigen heißen Siebensegmentanzeige weil sie 7 Segmente hat. Alle Digits sind parallel an das Schieberegister angeschlossen. Die Auswahl, welches Digit leuchtet erfolgt mit den Bits 15 .. 12 im Datenwort, das an das Display geschickt wird..**

Siehe Anhang.

Aufgabe 1: Test der Siebensegmentanzeige und Ermittlung der Kodierung.

Mittels folgendem Testprogramm können Sie die Siebensegmentanzeige durchprobieren und die Tabelle ausfüllen:

```
.equ Einerstelle,Bit12
.equ Zehnerstelle,Bit13
.equ Hunderterstelle, Bit14
.equ Tausenderstelle, Bit15

.equ Taster,Bit1          //Low Aktiv

main:
    bl      MFS_serial_init    //Initialisierung der Schnittstelle
    ldr     R5,=GPIOA          //Taster an GPIOA
    mov     R4,#1              //R4 als Schieberegister.

schleife:
    ldrb    R0,[R5,IDR]        //Taster abfragen
    tst     R0,Taster
    bne     weiter             //nicht gedrückt, weiter zur Anzeige
    mov     R0,#20              //entprellen
    bl      HAL_Delay
    lsl     R4,#1              //Schieberegister

warte:
    ldrb    R0,[R5,IDR]        //Warten auf Taste losgelassen
    tst     R0,Taster
    beq     warte
    mov     R0,#20              //entprellen
    bl      HAL_Delay

weiter:
    mov     R0,R4              //Schieberegister R4 nach R0
    add     R0,Einerstelle      //Einerstelle hinzufügen
    bl      MFS_sendWord        //und wegschicken
    b       schleife

.end
```

<i>Ziffer</i>	<i>Code</i>
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

## Aufgabe 2: Verwendung einer ROM-Tabelle (Array)

a) Tabelle anlegen nach **b schleife** und vor **.end**

```
...  
b schleife  
  
seg7:  
.byte 0b00000011,0b10011111,...  
  
.end
```

b) Ausgabe mit Tabelle (Array)

Ausgabe: //R0 beinhaltet den Tabellenindex: Ausgabe=seg7[R1];

```
    ldr    R6,=seg7           //Zuvor R6 mit seg7 initialisiert  
...  
    ldrb   R0,[R6,R5]         //Schreibweise in C: R0=seg7[R5]  
    add    R0,Einerstelle  
    bl     MFS_sendWord
```

c) Schreiben Sie ein Programm, das die Anzahl der Tastenbetätigungen von Taster PA1 in R5 zählt und an der Einerstelle ausgibt. Beachten Sie, nach 9 soll es wieder mit 0 weitergehen. Beachten Sie weiter, dass immer auf Taste losgelassen, gewartet wird.

d) PAP von c)

Hilfe: Siehe Eisenbahnvideo

## Anhang Codeschnipsel

```
schleife:      //Endlosschleife
```

```
main:  
//Initialisierung  
    mov    R5,#0        //Zähler in R5 mit 0 beginnen  
    ldr     R4,=GPIOA     //R4 verweist auf GPIOA  
    ldr     R6,=seg7      //R6 verweist auf die Tabelle seg7  
    bl      MFS_serial_init //Initialisierung der Schnittstelle
```

```
ldrb    R0,[R6,R5]      //R0=seg7[R5]
```

```
mov     R5,#0          //Zähler in R1 mit 0 beginnen
```

```
gedrueckt:      //wenn PA1-Taste gedrückt  
    add     R5,#1        //um 1 hochzählen
```

```
seg7:  
.byte 0b00000011,0b10011111,0b00100101,0b00001101,0b10011001,0b01001001,0b01000001,  
0b00011111,0b00000001,0b00001001
```

```
b        schleife      //von vorne
```

```
.equ Einerstelle,Bit12  
.equ Zehnerstelle,Bit13  
.equ Taster,Bit1
```

```
add     R0,Einerstelle  //Anzeige einschalten
```

```
ldr     R4,=GPIOA      //R4 verweist auf GPIOA
```

```
ldr     R0,[R4,IDR]    //GPIOA.IDR einlesen (Bit10=Taster PA1)  
tst     R0,Taster      //Taster prüfen  
beq     gedrueckt      //0 dann gedrückt
```

```
nochgedrueckt:      //warten solange Taste-PA1 noch gedrückt  
    ldr     R0,[R4,IDR] //GPIOA.IDR einlesen  
    tst     R0,Taster    //Bit10 prüfen  
    beq     nochgedrueckt//bei noch gedrückt erneut Taste abfragen
```

```
bl      MFS_sendWord
```

```
strb    R0,[R2,ODR]    //R0 auf die Siebensegmentanzeige ausgeben
```

```
cmp     R5,#10         //R5==10?  
bne     schleife       //nein, dann von vorne  
mov     R5,#0          //ja, dann R1 wieder bei 0 beginnen  
b       Schleife
```

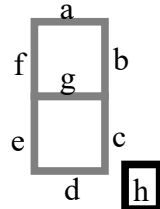
# Anhang: Multifunctionshield

Die Ausgabe der Daten auf das LED-Display erfolgt seriell. Die Segmente sind Low-Aktiv.

Tabelle:

```
char siebenseg[10]=
{0b00000011,0b10011111,0b00100101,
0b00001101,0b10011001,0b01001001,
0b01000001,0b00011111,0b00000001,
0b00001001};
```

segmente: 0Babcdefgh



Anschlüsse:

Seriell Data SDI:	D8 (PA9)	Konfigurieren als Digitalausgang in CubeIDE
Shift Clock SFTCLK:	D7 (PA8)	Konfigurieren als Digitalausgang in CubeIDE
Latch Clock LCHCLK:	D4 (PB5)	Konfigurieren als Digitalausgang in CubeIDE

Poti:	A0 (PA0)	Konfigurieren als ADC_IN in CubeIDE
Taste S1:	A1 (PA1)	Konfigurieren als Digitaleingang in CubeIDE
Taste S2:	A2 (PA4)	Konfigurieren als Digitaleingang in CubeIDE
Taste S3:	A3 (PB0)	Konfigurieren als Digitaleingang in CubeIDE

(alle aktiv Low)

LED D1:	D13 (PA5)	Konfigurieren als Digitalausgang in CubeIDE
LED D2:	D12 (PA6)	Konfigurieren als Digitalausgang in CubeIDE
LED D3:	D11 (PA7)	Konfigurieren als Digitalausgang in CubeIDE
LED D4:	D10 (PB6)	Konfigurieren als Digitalausgang in CubeIDE
Summer:	D3 (PB3)	Konfigurieren als Digitalausgang in CubeIDE (Pegel High einstellen)

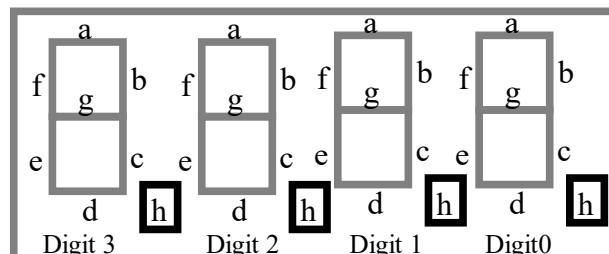
(alle aktiv Low)

Operationen:

	Assembler
Initialisierung des Displays	bl MFS_serial_init
Daten an das Display schicken	mov r0,#Data bl MFS_sendWord
Beispiel: 2 auf Digit 0 = 0x1025	Beispiel: mov r0,#0x1025 bl MFS_sendWord

Format von Data:

Hinweis:  
Das Display arbeitet im  
Multiplexbetrieb!



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Digit 3	Digit 2	Digit 1	Digit 0	0	0	0	0	a	b	c	d	e	f	g	h

