

Arbeiten mit Tabellen im Codespeicher

1. Aufgabenstellung: Durch Blinken sollen an Digit0 nacheinander die Ziffern von Telefonnummern angezeigt werden.

2. Beschreibung der 7-Segment-Anzeige:
Siehe Anhang

3. Tabellen im Mikrocontroller:

siebensegment:

.byte 0b00000011,0b10011111,0b00100101,...

Telefonnummer:

.byte 1,1,8,3,3

Fragen:

Analysieren Sie obige Programmzeile.

Welchen Wert hat R0 nach folgende Instruktionen:

mov R0,#3

ldr R7,=Telefonnummer

ldrb R0,[R7,R0]

Aufgaben:

Aufgabe 1: Erstellen Sie eine Codetabelle für die Telefonnummer: 11833. Als zusätzliche Zahl soll als letztes die Zahl 255 in Ihrer Tabelle erscheinen (kennzeichnet das Tabellenende, Telefonnummern können unterschiedlich lang sein)

Aufgabe 2: Erstellen Sie eine Codetabelle für die Siebensegmentanzeige

Aufgabe 3: Erstellen Sie ein PAP, in dem nacheinander die Ziffern der Telefonnummer aus der Codetabelle geholt werden. Die Ziffern sollen unter Verwendung der Codetabelle aus Aufgabe 2 auf die 7 Segmentanzeige gegeben werden. Die Anzeigedauer soll 1s betragen. Zwischen den Ziffern soll 1 Sekunde Pause sein. Wenn alle Ziffern angezeigt sind (Code 255) soll 3 s Pause sein.

Die Wartezeit muss in einem Unterprogramm `warte1s` mit Zeitschleife realisiert werden

Aufgabe 4:

Schreiben Sie den Programmcode in Assembler

Testen Sie Ihr Programm.

Aufgabe 5: Jetzt sollen mehrere Telefonnummern gespeichert werden. Die Auswahl erfolgt mit den Schaltern (GPIOB).

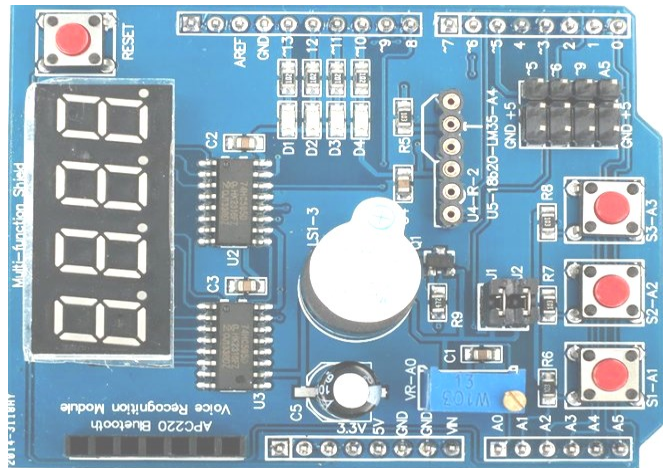
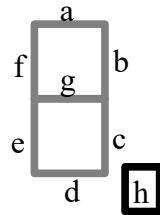
Anhang: Multifunctionshield

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

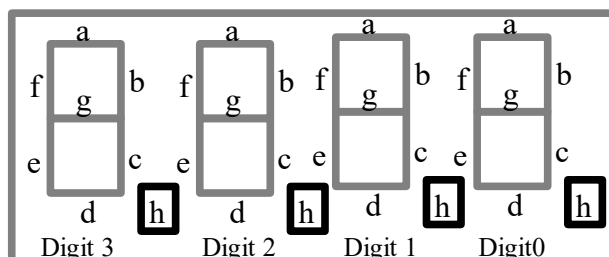
Poti:	A0 (PA0)	Konfigurieren als ADC_IN in CubeIDE
Taste S1:	A1 (PA1)	Konfigurieren als Digitaleingang in CubeIDE
Taste S2:	A2 (PA4)	Konfigurieren als Digitaleingang in CubeIDE
Taste S3:	A3 (PB0)	Konfigurieren als Digitaleingang in CubeIDE
(alle aktiv Low)		
LED D1:	D13 (PA5)	Konfigurieren als Digitalausgang 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

Lösung:

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

main:

```
    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
    ldr     R7,=TelNr
    bl      MFS_serial_init      //Initialisierung der Schnittstelle
schleife: //Endlosschleife
    ldrb    R0,[R7,R5]          //R0=TelNr[R5]
    cmp     R0,#255
    beq     vonVorne
    ldrb    R0,[R6,R0]          //R0=seg7[R0]
    add     R0,Einerstelle      //Einerstelle hinzufügen
    bl      MFS_sendWord
    mov     R0,#1000
    bl      HAL_Delay

    mov     R0,#0                //zwischen 2 Ziffern
    bl      MFS_sendWord
    mov     R0,#100
    bl      HAL_Delay

    add     R5,#1                //nächste Ziffer

    b       schleife
vonVorne:
    mov     R0,0xFF              //Aus
    bl      MFS_sendWord
    mov     R0,#3000             //3s
    bl      HAL_Delay
    mov     R5,#0                //von vorne
    b       schleife
```

seg7:

```
.byte 0b00000011,0b10011111,0b00100101,0b00001101,0b10011001,0b01001001,0b01000001,0b00011111,0b000000
01,0b00001001
```

TelNr:

```
.byte 1,1,8,3,3,255
.end
```