# 1 Fragen

- Was ist ROM?

    a Das Rom ist ein Speicher, dessen Inhalt nur vom Prozessor gelesen werden kann. Sein Inhalt lässt sich durch UV-Licht löschen.

- Wie funktioniert das Status-Register? (Info in den Themen Blättern)
- Welche Register gibt es noch?
- Können wir angeschrieben Programme mitnehmen oder nur als Textdokument?

# 2 Aufgaben

- HEX zahlen lernen. Wie rechnent man diese mit dem Taschenrechner?
- Die erste Aufgabe gibt 35-39,40 der Frage Katalog Meist bekommt man so 20 Punkte (Die Antworten können in Stichpunkten beantwortet werden)
- Kommentare zu Übungsaufgabe 1 hinzufügen

**TABLE 8-3 RESET CONDITION FOR PROGRAM COUNTER AND THE STATUS REGISTER**

| Condition | Program Counter | STATUS Register |
|---|---|---|
| Power-on Reset | 000h | 0001 1xxx |
| $\overline{MCLR}$ Reset during normal operation | 000h | 000u uuuu |
| $\overline{MCLR}$ Reset during SLEEP | 000h | 0001 0uuu |
| WDT Reset (during normal operation) | 000h | 0000 1uuu |
| WDT Wake-up | PC + 1 | uuu0 0uuu |
| Interrupt wake-up from SLEEP | PC + 1 (1) | uuu1 0uuu |

Legend: u = unchanged, x = unknown.
Note 1: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

**TABLE 8-4 RESET CONDITIONS FOR ALL REGISTERS**

| Register | Address | Power-on Reset | $\overline{MCLR}$ Reset during:<br>– normal operation<br>– SLEEP<br>WDT Reset during normal operation | Wake-up from SLEEP:<br>– through interrupt<br>– through WDT Time-out |
|---|---|---|---|---|
| W | — | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF | 00h | ---- ---- | ---- ---- | ---- ---- |
| TMR0 | 01h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCL | 02h | 0000h | 0000h | PC + 1 (2) |
| STATUS | 03h | 0001 1xxx | 000q quuu (3) | uuuq quuu (3) |
| FSR | 04h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA | 05h | ---x xxxx | ---u uuuu | ---u uuuu |
| PORTB | 06h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| EEDATA | 08h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| EEADR | 09h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCLATH | 0Ah | ---0 0000 | ---0 0000 | ---u uuuu |
| INTCON | 0Bh | 0000 000x | 0000 000u | uuuu uuuu (1) |
| INDF | 80h | ---- ---- | ---- ---- | ---- ---- |
| OPTION_REG | 81h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PCL | 82h | 0000h | 0000h | PC + 1 |
| STATUS | 83h | 0001 1xxx | 000q quuu (3) | uuuq quuu (3) |
| FSR | 84h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TRISA | 85h | ---1 1111 | ---1 1111 | ---u uuuu |
| TRISB | 86h | 1111 1111 | 1111 1111 | uuuu uuuu |
| EECON1 | 88h | ---0 x000 | ---0 q000 | ---0 uuuu |
| EECON2 | 89h | ---- ---- | ---- ---- | ---- ---- |
| PCLATH | 8Ah | ---0 0000 | ---0 0000 | ---u uuuu |
| INTCON | 8Bh | 0000 000x | 0000 000u | uuuu uuuu (1) |

Legend: u = unchanged, x = unknown, - = unimplemented bit read as '0',
q = value depends on condition.
Note 1: One or more bits in INTCON will be affected (to cause wake-up).
2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).
3: Table 8-3 lists the reset value for each specific condition.

### PORT RA und TRIS RA Initialisierungssequenz

```
    BSF     status, rp0     ;auf Bank 1 umschalten, dort sind die
    BCF     trisa, 0        ;TRIS-Register. RA0 wird Ausgang
    BCF     trisa, 1        ;RA1 wird ebenfalls Ausgang
    BCF     status, rq0     ;zurueck auf Bank 0 schalten
    ...
    ...
    BSF     porta, 0        ;setzt den Pegel an RA0 auf high
    BCF     porta, 1        ;setzt den Pegel an Ra1 auf low
```

### Vergleich zweier Speicherstellen

```
  MOVF    adr1, W         ;ein Argument ins W-Register holen
  XORWF   adr2, W         ;XOR verknuepfen und Ergebnis in
                          ;W-Register, so bleiben adr1 und
                          ;adr2 unveraendert
  BTFSC   status, Zflag
  GOTO    sindGleich
  GOTO    sindUngleich
```

### Vergleich zweier Speicherstellen auf größer / kleiner

```
  MOVF    adr1, W         ;ein Argument ins W-Register holen
  SUBWF   adr2, W         ;subtrahiere W von Inhalt von adr2
                          ;und schreib Ergebnis ins
                          ;W-Register, so bleiben adr1 und adr2
                          ;unveraendert
  BTFSC   status,Zflag
  GOTO    sindGleich
  BTFSC   status, CFlag
  GOTO    kleiner         ;es gab einen Ueberlauf im Carry
  GOTO    groesser
```

### 5 Multiplikator

In `HWert1` steht wie viele überläufe es gab.51

```
CLRF    HWert1      ;Loescht HWert1
BCF     status, 0   ;Carryflag wird geloescht Carryflag ist auf 0
MOVF    LWert1, w   ;LWert in das W-Register
RLF     LWert1      ;verdoppelt den LWert
RLF     HWert1      ;moeglicher Ueberlauf der Operation wird in
                    ;HWert von rechts geschoben. Im Carray steht 0.
RLF     LWert1      ;verdoppelt den LWert
RLF     HWert1      ;moeglicher Ueberlauf der Operation wird in
                    ;HWert von rechts geschoben. Im Carray steht 0.
```

```
ADDWF   LWert       ;Der urspruengliche LWert wird noch dazu addiert
BTFSC   status, 0   ;Es wird geprueft ob Carry 0 ist wenn nicht
INCF    HWert1      ;wird HWert1 inkrementiert.
```

### 16-Bit-Zähler + Addierer

In `HWert1` steht wie viele überläufe es gab.51

```
;16-Bit_Increment -> 2Variablen
;WertL & WertH
incf    WertL
btfsc   status,zero     ;Wenn 1 dann nicht springen
incf    WertH
;16-Bit_Addition
movf    Wert1L,w
addwf   Wert2L          ;Addiere die beiden unteren 8 Bit
btfsc   status,carry    ;Wenn es Ueberlauf gab, dann Hoeheres Byte +1
incf    Wert1H,w
addwf   Wert2H          ;Wieder zusammenaddieren
```

# Eingangsimpuls erfassen

Schreiben Sie ein Assemblerprogramm für den 16C83, das einen Eingangsimpuls (0,1 ms bis 0,5 ms) erfasst und daraus einen 8 x so langen Ausgangsimpuls erzeugt. Der Ausgangsimpuls soll erst dann erscheinen, wenn der Eingangsimpuls wieder weg ist. Die Quarzfrequenz beträgt 4 MHz; ein einfacher Befehl benoetigt somit 1 $\mu s$.

`Pegel prüfen bis eine 1 kommt`

```
Label1
  BTFSC   porta, 0
  GOTO    Label1        ;warten auf low
Label2
  BTFSS   porta, 0
  GOTO    Label2        ;wenn das ueberspringt gab es eine ssteigende Flanke
                        ;Ab jetzt messen -> solange incrementieren

;8-bitzaehler geht bis 512 s

Label3
  INCF    Dauer, 1      ;1 Takt
  BTFSC   porta, 0      ;1
  GOTO    Label3        ;2 Vier Takte fuer den letzten 3

;Ausgabe
Label3
  MOVLW   8             ;ein Register wird auf 8 gesetzt
  MOVWF   Schleife      ;ist eine Variable
  BSF     porta, 1
```

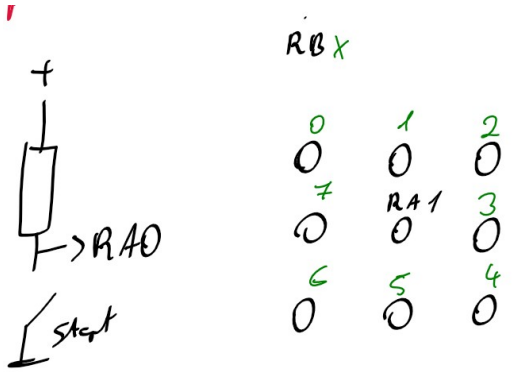
```
Label5
  MOVF    Dauer, W
  MOVWF   Counter
Label4
  DECF    Counter
  BTFSS   status,zero
  GOTO    Label4
  DECTSZ  Schleife
  GOTO    Label5
  BCF     porta, 1
  GOTO    Label2
```

## Kegel Aufgabe

```
Start
    BSF     status, rp0
    MOVLW   255
    MOVWF   06/trisb
    MOVLW   255
    MOVWF   trisa
    BCF     status, rp0
    GOTO    Hauptprogramm

Hauptprogramm
    CALL    Unterprogramm
    GOTO    Hauptprogramm

Unterprogramm
    CLRF    counter
    BTFSC   RA0
    RETLW   0
    BTFSC   RA4
    INCF    counter
    MOVLW   8
    MOVWF   loopCnt

Schleife
    BTFSC   RB, 0
    INCF    counter
    RRF     RB
    DECFSZ  loopCnt
    GOTO    Schleife

    MOVF    counter, W
    RETURN
```

### Gib das Bild eines Würfels aus

Es soll gezählt werden wie oft der RA0 Schalter gedrückt wird. Diese Zahl sollen mit einem Würfel ausgegeben werden.




```
1  Start
2      CLRF    counter
3      BSF     status, rp0
4      MOVLW   0           ;beide Zeilen koennte
5      MOVWF   06          ;man auch mit CLRF 06 ersetzten
6      BCF     05,1
7      BCF     status, rp0
8
9  Hauptprogramm
10     MOVLW   1
11     MOVWF   counter
12
13 Loop
14     BTFSC   RA,0
15     GOTO    Ausgabe
16     INCF    counter
17     MOVLW   7
18     XORWF   counter,w
19     BTFSC   status, 2
20     GOTO    Hauptprogramm
21     GOTO    Loop
22
23 Ausgabe
24     MOVF    counter, w
25     CALL    Tabelle1
26     MOVWF   portb
27     MOVF    counter, w
28     CALL    Tabelle2
29     MOVWF   porta
30     GOTO    Hauptprogramm
31
32 Tabelle1
```

```
33      ADDWF   pcl
34      nop
35      RETLW   0
36      RETLW   00010001b
37      RETLW   00010001b
38      RETLW   01010101b
39      RETLW   01010101b
40      RETLW   11011101b
41
42  Tabelle1
43      ADDWF
44      nop
45      RETLW   00000010b
46      RETLW   00000000b
47      RETLW   00000010b
48      RETLW   00000000b
49      RETLW   00000010b
50      RETLW   00000000b
```

### Binärcodierer ohne Interrupt

```
1   ;*******************************************************
2   ; Binarycounter.src
3   ;*******************************************************
4     device 16f84
5   ;Symbol definieren
6   status    equ 3
7   zero      equ 2
8   rp0       equ 5
9   trisa     equ 5
10  trisb     equ 6
11  porta     equ 5
12  portb     equ 6
13
14  ;Hex-Zahlen: h am Ende, bei Zahlen mit Buchstaben an erster Stelle eine 0
        davor
15  wert      equ 0ch
16  alterw      equ 13
17  counter   equ 14
18
19
20  org       0
21
22  ;Einsprung beim Einschalten (Power on)
23  cold
24    bsf     status,rp0   ;auf Bank 1 umschalten
25    movlw     0
26    movwf     trisb          ;PortB wird komplett als Ausgang geschaltet
27    bcf       trisa,3          ;RA3 wird Ausgang (Carry)
28    bcf     status,rp0   ;zurueck auf Bank 0
```

```
29
30    ;Definieren von alterw mit aktuellem Wert an RA0
31      movf      porta,w          ;PortA lesen
32      andlw     00000001b
33      movwf     alterw
34
35    ;Hauptschleife
36    loop
37      clrf      counter          ;Reset und Startwert
38      clrf      portb
39
40    loop1
41    ;Reset aktiv?
42      btfss     porta,1          ;Reseteingang
43      goto      loop
44    ;Inhibit aktiv?
45      btfsc     porta,2          ;Inhibiteingang
46      goto      loop1
47
48    ;Takteingang lesen
49      movf      porta,w          ;PortA komplett eingelesen
50      andlw     1              ;Nur R0 ist von Interesse
51
52      xorwf     alterw,w     ;Wenn beide gleich, keine Flanke
53      btfsc     status,zero    ;Beide gleich, Zero gesetzt
54      goto      loop1          ;Nichts passiert
55      movlw   1
56      xorwf     alterw           ;Beinhaltet neuen Pegel an RA0
57      btfss     alterw,0
58      goto      loop1
59
60    ;Richtige Flanke gefunden
61    zuruecksetzen
62      incf    counter        ;Zaehler erhoehen
63      movlw   0fh
64      andwf   counter,w
65      xorlw   10             ;Vergleich untere 4 Bits mit 10
66      btfss   status,zero
67      goto    ausgabe
68      movlw   6
69      addwf   counter
70      movlw   0a0h
71      xorwf   counter,w
72      btfss   status,zero
73      goto    ausgabe
74      clrf    counter
75      bsf     porta,3
76   ;ausgabe auf 7-Segmentanzeige
77    ausgabe
78      movf    counter,w      ;zuerst rechte Stelle anzeigen
79      call    convert        ;in 7-Segmentanzeige umsetzen
```

```
80      movwf   portb
81      bcf     porta,4         ;Digit einschalten
82      bsf     porta,4         ;Digit wieder
83    ausschalten
84      swapf   counter,w       ;jetzt linke Stelle anzeigen
85      call    convert
86      movwf   portb
87      bcf     porta,5
88      bsf     porta,5
89      goto    loop1
90
91   ;setzt Binaerzahl in Bitmuster fuer 7-Segmentanzeige um.
92    convert
93      andlw   15
94      addwf   pcl             ;w= offset der zum PCL addiert wird.
95      retlw   3fh             ;0
96      retlw   06h             ;1
97      retlw   5bh             ;2
98      retlw   4fh             ;3
99      retlw   66h             ;4
100     retlw   6dh             ;5
101     retlw   7dh             ;6
102     retlw   07h             ;7
103     retlw   7fh             ;8
104     retlw   6fh             ;9
105     retlw   0               ;ungueltig
106     retlw   0
107     retlw   0
108     retlw   0
109     retlw   0
110     retlw   0
111
112   end
```

### Interrupts

```
1   ;*****************
2   ;************************Interrupts*********************************
3   ;******************
4           device 16f84
5   intcon      equ     0bh
6   inte        equ     4
7   intf        equ     1
8   gie         equ     7
9   status      equ     3
10  rp0         equ     5
11  option      equ     1
12  intedg      equ     6
13  counter     equ     0ch
14  ;startadresse fuer Programm
```

```
15      org 0
16  cold
17      goto    main
18      nop
19      nop
20      nop
21  isr                         ;Interrupt springt immer an Adresse 4
22      btfss   intcon,intf     ;War es ein RB0-Signal
23      goto    intend          ;Nein, Fehler
24      bcf     intcon,intf     ;Interrupt RB0 wird bearbeitet
25      incf    counter         ;Zaehler wird erhoeht intend
26      retfie                  ;Return from Interrupt enable
27
28  main
29  ;Interrupts initialisieren
30      bsf     intcon,inte     ;RB0-Interrupt freischalten
31      bcf     intcon,intf     ;Interruptflag loeschen
32      bsf     status,rp0      ;auf Bank 1 schalten
33      bcf     option,intedg   ;auf Fallende Flanke pruefen
34      bcf     status,rp0      ;zurueck auf Bank0 schalten
35      bsf     intcon,gie      ;Globales Interrupt enable Bit
36  loop
37      goto    loop
```

## Binarycounter with Interrupts

```
1   ;*****************************************
2   ; binarycounter.src
3   ;*****************************************
4    device 16f84
5   ;Symbole definieren
6   status      equ     3
7   zero        equ     2
8   rp0         equ     5   ;bitadresse
9   trisa       equ     5   ;fileadresse
10  trisb       equ     6
11  porta       equ     5
12  portb       equ     6
13  pcl         equ     2
14  intcon      equ     0bh
15  inte        equ     4
16  intf        equ     1
17  gie         equ     7
18  option      equ     1
19  intedg      equ     6
20  counter     equ     13h
21      org 0
22  ;erster Befehl an Adresse 0, alle weiteren hinten dran
23  ;Einsprung beim Einschalten (Power on)
24  cold
```

```
25      goto    main
26      nop
27      nop
28      nop
29  isr ;Interrupt springt immer an Adresse 4
30      btfss   intcon,intf     ;War es ein RB0-Signal
31      goto    intend          ;Nein, Fehler
32      bcf     intcon,intf     ;Interrupt RB0 wird bearbeitet
33      incf    counter         ;Zoehler wird erhoeht intend
34      retfie                  ;Return from Interrupt enable
35
36      bsf     status,rp0      ;auf Bank 1 umschalten
37      movlw   0
38      movwf   trisb           ;PortB wird komplett als Ausgang geschaltet
39      bcf     trisa,3         ;RA3 wird Ausgang (Carry)
40      bcf     status,rp0      ;zurueck auf Bank 0
41  main
42  ;Interrupts initialisieren
43      bsf     intcon,inte     ;RB0-Interrupt freischalten
44      bcf     intcon,intf     ;Interruptflag loeschen
45      bsf     status,rp0      ;auf Bank 1 schalten
46      bcf     option,intedg   ;auf Fallende Flanke pruefen
47      bcf     status,rp0      ;zurueck auf Bank 0 schalten
48      bsf     intcon,gie      ;Globales Interrupt enable Bit
49  ;Hauptschleife
50  loop
51      clrf    counter         ;Reset und Startwert
52      clrf    portb
53  loop1
54  ;reset aktiv?
55      btfss   porta,1         ;Reseteingang (0=Reset)
56      goto    loop
57  ;inhibit aktiv?
58      btfsc   porta,2         ;Inhibit-Eingang
59      goto    loop1
60      xorwf   counter,w
61      btfss   status,zero
62      goto    ausgabe
63      clrf    counter
64      bsf     porta,3
65  ;ausgabe auf 7-Segmentanzeige
66  ausgabe
67      movf    counter,w       ;zuerst rechte Stelle anzeigen
68      call    convert         ;in 7-Segmentanzeige umsetzen
69      movwf   portb
70      bcf     porta,4         ;Digit einschalten
71      bsf     porta,4         ;Digit wieder ausschalten
72      swapf   counter,w       ;jetzt linke Stelle anzeigen
73      call    convert
74      movwf   portb
75      bcf     porta,5
```

```
76      bsf     porta,5
77      goto    loop1
78
79  ;setzt Binaerzahl in Bitmuster fuer 7-Segmentanzeige um.
80  convert
81      andlw   15
82      addwf   pcl             ;w= offset der zum PCL addiert wird.
83      retlw   01111110b       ;0
84      retlw   00001100b       ;1
85      retlw   10110110b       ;2
86      retlw   10011110b       ;3
87      retlw   11001100b       ;4
88      retlw   11011010b       ;5
89      retlw   11111010b       ;6
90      retlw   00001110b       ;7
91      retlw   11111110b       ;8
92      retlw   11011110b       ;9
93      retlw   0               ;ungueltig
94      retlw   0
95      retlw   0
96      retlw   0
97      retlw   0
98      retlw   0
99
100  end
```

### AUFGABE 12 Delay

```
1   ;*********************************
2   ;    AUFGABE 12
3   ;*********************************
4
5   device 16f84
6   ;Symbole definieren
7   delaycnt    equ
8
9   delay166
10      movlw   52
11      movwf   delaycnt
12  delay166_a
13      decfsz  delaycnt
14      goto delay166_a
15      return
16
17
18  delay625
19      movlw   206
20      movwf   delaycnt
21  delay625_a
22      decfsz  delaycnt
```

```
23      goto    delay625_a
24      return
25
26  warte_null
27      btfss   rb,4            ;Nulldurchgang?
28      goto    warte_null
29  ;Nulldurchgang gefunden
30      call    delay166
31  ;Schalter einlesen
32      movf    portb,w         ;Schalter = 0?
33      andlw   15              ;Obere 4 Bits auf 0 setzen
34      btfsc   status,zero
35      goto    warte_null
36
37      movwf   loopcnt
38
39  loop
40      call    delay625
41      decfsz  loopcnt
42      goto    loop
43
44  ;TRIAC einschalten
45      bsf     rb,5
46      call    delay166
47      bcf     rb,5
48      goto    warte_null
```