```
------------------------------------------------------------
Umwandlung binär -> 7-Seg (RETLW) -> Datentabelle:
____________________________________________________________

; Wandle 4-Bit Zahl in BCD Information
         device    16C83

Ausgang  EQU       6         ; Ausgang
Eingang  EQU       5         ; Eingang
PCL      EQU       2         ; Programmzähler (low)

         ORG       0

; Verschiedenes im Status-Register
         BSF       3,5       ; Umschalten auf Bank 1
         CLRF      6         ; Port B als Ausgang definieren
         BCF       3,5       ; Umschalten auf Bank 0

; Initialisierung
         CLRF      Ausgang

; Hauptschleife
LOOP
         MOVF      Eingang, W; Lese Eingang in W
         ANDLW     0Fh       ; Hole untere 4 Bits
         CALL      TABLE     ; Springe zu TABLE (Unterprogrammaufruf) ->
Schreibe nächste Adresse auf den Stack
         MOVWF     Ausgang   ; Gebe W auf Port B aus
         GOTO      LOOP      ; Schleife

; Tabelle
TABLE
         ADDWF     PCL       ; Addiere W zum Programmzähler (springe in der
Tabelle)
         RETLW     3Fh       ; 7-Seg: 0
         RETLW     06h       ; 7-Seg: 1
         RETLW     5Bh       ; 7-Seg: 2
         RETLW     4Fh       ; 7-Seg: 3
         RETLW     66h       ; 7-Seg: 4
         RETLW     6Dh       ; 7-Seg: 5
         RETLW     7Dh       ; 7-Seg: 6
         RETLW     07h       ; 7-Seg: 7
         RETLW     7Fh       ; 7-Seg: 8
         RETLW     0DFh      ; 7-Seg: 9
         RETLW     76h       ; 7-Seg: H
         RETLW     79h       ; 7-Seg: E
         RETLW     38h       ; 7-Seg: L
         RETLW     73h       ; 7-Seg: P
         RETLW     80h       ; 7-Seg: . (dot)
         RETLW     08h       ; 7-Seg:   (blank)
```

```
END
```

```
------------------------------------------------------------
 Binärzähler -> Flankenerkennung:
____________________________________________________________

; 8-bit Binärzähler
; RB0 ... RB7 - Ausgänge
; RA0 - Zähleingang, RA1 - Reset low, RA2 - Inhibit low, RA3 - Carry

         device    16C83

; Definition der Variablen
Counter  EQU       6         ; Zähler (Port B)
Pegel    EQU       11h       ; Pegelinformation
Zero     EQU       3,2       ; Zero Flag im Status Register
Eingang  EQU       5,0       ; Zähleingang
Reset    EQU       5,1       ; Reset
Inhibit  EQU       5,2       ; Inhibit
Carry    EQU       5,3       ; Carry-Out
Ausgang  EQU       6         ; Ausgang

         ORG       0

; Verschiedenes im Status-Register
         BSF       3,5       ; Umschalten auf Bank 1
         CLRF      6         ; Port B als Ausgang definieren
         BCF       Carry     ; Setze Pin 3 von Port A als Ausgang (5.3)
         BCF       3,5       ; Umschalten auf Bank 0

; Initialisierung der Variablen
         CLRF      Counter   ; Counter = NULL
         CLRF      Pegel     ; Pegel = NULL

; Hauptschleife
LOOP
         BTFSC     Reset     ; Prüfe auf Reset = 0
```

```
         GOTO      LOOP2     ; Springe zu LOOP2
         CLRF      Counter   ; Zähler = NULL
         BCF       Carry     ; Carry = 0
         GOTO      LOOP      ; Reset-Schleife

; Inhibit-Test
LOOP2
         BTFSS     Inhibit   ; Prüfe auf Inhibit = 0
         GOTO      LOOP      ; Inhibit-Schleife

; Abfrage auf steigende Flanke
         MOVF      5,W       ; Schreibe Port A in W
         BTFSC     Pegel,0   ; Prüfe auf Pegel = 0 (Pegel.0)
         GOTO      PEGEL1    ; Springe zu PEGEL1
         ANDLW     1         ; Maskiere W mit 00000001 -> nur LSB (W.0) wird
beibehalten (bitweise UND-Verknüpfung)
         BTFSC     Zero      ; Überprüfe auf Zero-Flag = 0
         GOTO      PEGEL1    ; Springe zu PEGEL1
         ; Es liegt eine steigende Flanke vor
         BCF       Carry     ; Setze Carry = NULL
         INCF      Counter   ; Inkrementiere Zähler
         BTFSC     Zero      ; Prüfe Zero-Flag = 0 (gibt Überlauf an -> Zero =
0)
         BSF       Carry     ; Setze Carry-Bit bei Überlauf

; Speichere Pegel
PEGEL1
         MOVWF     Pegel     ; Schreibe W in Pegel (aktualisiere Pegel) -> Pegel
= 0000000x
         GOTO      LOOP      ; Haupt-Schleife


------------------------------------------------------------
 divide: Teilen
____________________________________________________________

         device    16C83

Zahl1    EQU       10h       ; Zähler
Zahl2    EQU       11h       ; Nenner
Erg      EQU       12h       ; Ergebnis
Stellen  EQU       13h       ; Stellenzahl
Carry    EQU       0         ; Carry-Bit im Status Register (Bit 0)
Status   EQU       3         ; Status-Register

         ORG       0

         MOVLW     222       ; Lade 222d in Zahl1
         MOVWF     Zahl1
         MOVLW     36        ; Lade 36d in Zahl2
         MOVWF     Zahl2
         CLRF      Erg       ; Ergebnis = NULL
```

```
        MOVLW     1
        MOVWF     Stellen   ; Stellenanzahl = NULL

; Schleife zur Analyse der Stellenanzahl
LOOP1
        BTFSC     Zahl2,7   ; Überprüfe ob MSB (Zahl2.7) gesetzt, überspringe
bei 0
        GOTO      LOOP2     ; Springe zu LOOP2
        BCF       Status,Carry        ; Lösche Carry-Bit im Status-Register
(3.7)
        RLF       Zahl2     ; Rotiere Zahl2 um 1 Stelle nach links
        INCF      Stellen   ; Inkrementiere Stellenanzahl
        GOTO      LOOP1     ; Schleife

; Schleife zur Division
LOOP2
        MOVF      Zahl2, W  ; Lade Zahl2 in W
        SUBWF     Zahl1     ; Subtrahiere Zahl1 von Zahl2 (in W)
        BTFSS     Status,Carry        ; Überprüfe Carry Bit, überspringe bei 1
(dann ist das Ergebnis positiv)
        GOTO      NEG       ; Springe zu NEG (bei negativem Ergebnis)
        GOTO      POS       ; Springe zu POS (bei positivem Ergebnis)

; Negatives Ergebnis
NEG
        ADDWF     Zahl1     ; Addiere Zahl1 zu Zahl2 (in W) -> macht vorherige
Subtraktion rückgängig
        GOTO      LOOP3     ; Schleife

; Positives Ergebnis
POS
        BSF       Erg,0     ; Setze das LSB von Ergebnis (Erg.0)

; Schleife über die Stellenanzahl
LOOP3
        BCF       Status,Carry        ; Lösche Carry-Bit
        RLF       Erg       ; Rotiere Ergebnis um 1 Stelle nach links
        RRF       Zahl2     ; Rotiere Zahl2 um 1 Stelle nach rechts
        DECFSZ    Stellen   ; Dekrementiere Stellenanzahl und überspringe bei 0
        GOTO      LOOP2     ; Schleife
        BCF       Status,Carry        ; Lösche Carry
        RRF       Erg       ; Rotiere Ergebnis um 1 Stelle nach rechts

;Endlosschleife
ENDE
        GOTO      ENDE
        END


------------------------------------------------------------
  freqdiv: Frequenzteiler (halbieren)
```

```
; Frequenzteiler

          device    16C83

Eingang   EQU       5,0       ; Zähleingang
Ausgang   EQU       5,1       ; Ausgang
Zustand   EQU       10h       ; Zustand

          ORG       0

; Verschiedenes im Status-Register
          BSF       3,5       ; Umschalten auf Bank 1
          BCF       Ausgang   ; Port B als Ausgang definieren
          BCF       3,5       ; Umschalten auf Bank 0

; Initialisierung
          MOVF      5,W       ; Port A einlesen
          ANDLW     1         ; Maskieren 0000 0001b
          MOVWF     Zustand   ; Alten Pegel merken

; HauptSchleife
LOOP
          MOVF      5,W       ; siehe oben
          ANDLW     1
          XORWF     Zustand   ; Entweder Oder (XOR) -> erkennt Flankenwechsel
          MOVWF     Zustand   ; Aktuellen Wert in Zustand speichern
          BTFSC     3,2       ; Überprüfe Zero-Bit, ob gesetzt
          GOTO      LOOP      ; Schleife (ja -> Bits sind gleich)
          BTFSS     Zustand,0 ; Wenn aktueller Zustand gleich 1, dann steigende
Flanke
          GOTO      LOOP
          MOVLW     2         ; Maske laden (0000 0010b)
          XORWF     5         ; Invertiert das 2. Bit (Ausgabe)
          GOTO      LOOP

END
```

```
------------------------------------------------------------
freqdiv2: Frequenzteiler (teile durch 4)
____________________________________________________________

; Frequenzteiler

         device    16C83

Eingang  EQU       5,0       ; Zähleingang
Ausgang  EQU       5,1       ; Ausgang
Zustand  EQU       10h       ; Zustand
Teiler   EQU       11h       ; Teiler

         ORG       0

; Verschiedenes im Status-Register
         BSF       3,5       ; Umschalten auf Bank 1
         BCF       Ausgang   ; Port B als Ausgang definieren
         BCF       3,5       ; Umschalten auf Bank 0

; Initialisierung
         MOVLW     4         ; Lade 4 in W
         MOVWF     Teiler    ; Speichere den Teiler
         MOVF      5,W       ; Port A einlesen
         ANDLW     1         ; Maskieren 0000 0001b
         MOVWF     Zustand   ; Alten Pegel merken

; HauptSchleife
LOOP
         MOVF      5,W       ; siehe oben
         ANDLW     1
         XORWF     Zustand   ; Entweder Oder (XOR) -> erkennt Flankenwechsel
         MOVWF     Zustand   ; Aktuellen Wert in Zustand speichern
         BTFSC     3,2       ; Überprüfe Zero-Bit, ob gesetzt
         GOTO      LOOP      ; Schleife (ja -> Bits sind gleich)
         BTFSS     Zustand,0 ; Wenn aktueller Zustand gleich 1, dann steigende
Flanke
         GOTO      LOOP
         MOVLW     2         ; Maske laden (0000 0010b)
```

```
        XORWF     5         ; Invertiert das 2. Bit (Ausgabe)
        GOTO      LOOP

END
```

---

muliply: Multiplizieren

```
; Multiplikation 8x8 bit -> 16bit Ergebnis

        device    16C83

; Variablendefinition
Zahl1   EQU       10h       ; Multiplikator
Zahl2   EQU       11h       ; Multiplikand
Erg     EQU       12h       ; 16bit reserviert (12h + 13h)
ii      EQU       14h       ; Zählvariable
Carry   EQU       0         ; Bit-Position des Carry Bit (Status.0)

        ORG       0

; Initialisierung der Variablen
        MOVLW     25        ; Lade 25d in Zahl1
        MOVWF     Zahl1
        MOVLW     235       ; Lade 235d in Zahl2
        MOVWF     Zahl2
        CLRF      Erg       ; Setze Erg auf NULL (12h)
        CLRF      Erg+1     ; Setze Erg auf NULL (13h)
```

```
        MOVLW   8       ; Setze Schleifenzähler auf 8
        MOVWF   ii

; Schleife
LOOP
        BCF     3,Carry ; Lösche Carry Bit im Status Register (3.0)
        RLF     ERG     ; LowByte von Erg um 1 nach links rotieren (mit
Carry)
        RLF     Erg+1   ; HighByte von Erg um 1 nach links rotieren,
übernehme Carry (Erg.7 -> (Erg+1).0)

        BTFSS   Zahl1,7 ; Überspringe nachfolgenden Befehl wenn Bit 7 von
Zahl1 gesetzt (Zahl1.7)
        GOTO    NOADD   ; Überspringe Addition und gehe zu Label NOADD

; Addition ausführen
ADD
        MOVF    Zahl2,W ; Addiere Zahl2 zu Erg (Umweg über W)
        ADDWF   Erg
        BTFSC   3,Carry ; Wenn Überlauf bei Addition (Carry == 1) führe
nachfolgenden Befehl aus
        INCF    ERG+1   ; Inkrementiere HighByte von Erg

; Keine Addition nötig
NOADD
        RLF     Zahl1   ; Rotiere Zahl1 um 1 nach links (Um nächstes Mal
wieder auf MSB prüfen zu können)

        DECFSZ  ii      ; Dekrementiere Zählvariable (überspringe nächsten
Befehl wenn ii == 0)
        GOTO    LOOP    ; Schleife

; Endlosschleife
ENDE
        GOTO    ENDE
        END
```

---

io: Eingänge / Ausgänge definieren -> Bank umschalten

```
; I/O Beispiel

        device  16C83
```

```
Wert     EQU       10h       ; Wert

         ORG       0

         MOVLW     0Fh       ; Bit 0...3 = 1 -> Eingang
         BSF       3,5       ; Umschalten auf Bank 1
         MOVWF     05        ; TRIS RA
         MOVLW     0F0h      ; Bit 0...3 = 1 -> Ausgang
         MOVWF     06        ; TRIS RB
         BCF       3,5       ; zurück auf Bank 0

LOOP
         MOVF      05,W      ; Lade Inhalt vom Port A in W
         MOVWF     Wert      ; W in Wert abspeichern
         MOVWF     06        ; W auf Port B ausgeben
         GOTO      LOOP

ENDE
         GOTO      ENDE
         END
```

---

freq: Teile Frequenz durch n -> Hole Teiler von Port B

```
; Frequenzteiler durch n

         device    16C83

 ; Definition der Variablen
Eingang  EQU       5,0       ; Zähleingang
Ausgang  EQU       5,1       ; Ausgang
Zustand  EQU       10h       ; Zustand
PortB    EQU       6         ; Hole Teiler von Port B
Teiler   EQU       11h       ; Teiler
Zero     EQU       3,2       ; Zero Flag im Status Register

         ORG       0

 ; Verschiedenes im Status-Register
         BSF       3,5       ; Umschalten auf Bank 1
         BCF       Ausgang   ; Port A, Bit 2 -> Ausgang
         BCF       3,5       ; Umschalten auf Bank 0

 ; Initialisierung (einmalig)
         MOVF      5,W       ; Port A einlesen
         ANDLW     1         ; Maskieren 0000 0001b
         MOVWF     Zustand   ; Alten Pegel merken

 ; Initialisierung (wird nach jedem Teilvorgang ausgeführt)
NEW
```

```
         MOVF      PortB,W   ; Lade Inhalte PortB in W (gibt Teiler an)
         MOVWF     Teiler    ; Speichere den Teiler

 ; Hauptschleife
LOOP
         CALL      FLANKE    ; Rufe UP zur Flankenerkennung auf
                             ;BTFSS     Zustand,0 ; Wenn aktueller Zustand
gleich 1, dann steigende Flanke
                             ;GOTO      LOOP
         DECFSZ    Teiler    ; Dekrementiere den Teiler, überpringe nächsten
Befehl, wenn 0
         GOTO      LOOP      ; Schleife
         MOVLW     2         ; Maske laden (0000 0010b)
         XORWF     5         ; Invertiert das 2. Bit (Ausgabe)
         GOTO      NEW

 ; Schleife zur Flankenerkennung
FLANKE
         MOVF      5,W       ; Eingang einlesen
         ANDLW     1         ; Hole letztes Bit (Zähleingang)
         XORWF     Zustand   ; Entweder Oder (XOR) -> erkennt Flankenwechsel
         MOVWF     Zustand   ; Aktuellen Wert in Zustand speichern
         BTFSC     Zero      ; Überprüfe Zero-Bit, ob gesetzt
         GOTO      FLANKE    ; Schleife (ja -> Bits sind gleich)
         RETURN

END
```

---

freqmul: Frequenz Multiplikator

```
 ; Frequenzverdoppler

         device    16C83

 ; Variablendeklaration
RA       EQU       5         ; Port A
STATUS   EQU       3         ; Status-Register
AWERT    EQU       10h       ;
COUNT1   EQU       11h       ; Zähler 1 (16bit)
COUNT2   EQU       13h       ; Zähler 2 (16bit)

         ORG       0

 ; Initialisierung
         BSF       STATUS,5  ; Springe zu Bank 1
         MOVLW     0FDh      ; Lade 253 in W
         MOVWF     RA        ; Setze Ausgangs-Pins
         BCF       STATUS,5  ; Zurück zu Bank 0
```

```
        CLRF    COUNT2    ; Lösche unteres Byte von Zähler 2
        CLRF    COUNT2+1  ; Lösche oberes Byte

 ; Erste Flanke erkennen
        MOVF    RA,W      ; Lade Port A in W
        ANDLW   1         ; Maskiere um LSB zu bekommen
        MOVWF   AWERT     ; In AWert speichern

 ; Warte auf erste Flanke
WARTE
        MOVF    RA,W      ; Lade Port A in W
        XORWF   AWERT,W   ; XOR-Verknüpfung mit dem alten Wert
        ANDLW   1         ; Maskiere um LSB zu erhalten
        BTFSC   STATUS,2  ; Überspringe, wenn Zero-Bit = 0
        GOTO    WARTE
        COMF    AWERT     ; Invertiere AWert

 ; Hauptschleife
LOOP
        CLRF    COUNT1    ; Lösche unteres Byte von Zähler 1
        CLRF    COUNT1+1  ; Lösche oberes Byte

LOOP2
        DECF    COUNT2    ; Dekrementiere Zähler 2
        INCF    COUNT2,W  ; Inkrementiere Zähler 2 und speichere Ergebnis in
W
        BTFSC   STATUS,2  ; Überprüfe Zero-Bit
        DECF    COUNT2+1  ; Wenn Überlauf im unteren Byte, dann Übertrag ins
obere Byte

        MOVF    COUNT2,W  ; Lade unteres Byte von Zähler 2 in W
        IORWF   COUNT2+1,W; ODER-Verknüpfung mit oberem Byte von Zähler 2
        BTFSC   STATUS,2  ; Prüfe Zero-Bit (Beide Bytes von Zähler 2 waren 0)
        GOTO    Verz1     ; Springe zu Verzögerung 1 (4 Takte)
        MOVLW   2         ; Lade die Maske 0000 0010b in W
        XORWF   RA        ; Invertiere 2. Bit bei Port A
        GOTO    EINLESEN

VERZ1
        GOTO    Verz2
VERZ2
        GOTO    Einlesen  ; Verzögerung insgesamt: 4 Takte (2x GOTO)

 ; Lese die Zeit ein
EINLESEN
        INCF    COUNT1    ; Inkrementiere unteres Byte von Zähler 1
        BTFSC   STATUS,2  ; Prüfe auf Zero-Bit -> Dann Überlauf von FF -> 00
        INCF    COUNT1+1  ; Inkrementiere oberes Byte von Zähler 1

        MOVF    RA,W      ; Siehe oben
        XORWF   AWERT,W
```

```
         ANDLW     1
         BTFSC     STATUS,2
         GOTO      Verz3     ; Springe zu Verzögerung 3
         BCF       STATUS,0  ; Lösche Carry-Bit
         RRF       COUNT1+1  ; Rotiere zuerste oberes Byte von Zähler 1
(eventueller Übertrag im Carry)
         RRF       COUNT1    ; Rotiere unteres Byte (mit evtl Carry)
         MOVF      COUNT1,W  ; Lade unteres Byte von Zähler 1 in W
         MOVWF     COUNT2    ; Speichere in Zähler 2
         MOVF      COUNT1+1,W; lade oberes Byte von Zähler 1 in W
         MOVWF     COUNT2+1  ; Speichere in Zähler 2
         COMF      AWERT     ; Invertiere AWert
         GOTO      LOOP

 ; Verzögerung 3 (7 Takte)
VERZ3
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         GOTO      LOOP2  ; Verzögerung von 7 Takten (7x 1 Takt (NOP))

END
```

---

rs232: RS232-Schnittstelle

```
 ; R232 - Aufgabe 10

         device 16C83

 ; Variablendeklaration
In       EQU       5         ; Eingang
Time     EQU       10h       ; Scheduling
Counter  EQU       11h       ; Bits zählen
```

```
Save     EQU       12h       ; Speicherregister (+1)
Wait     EQU       12h       ; Warten-Counter
w_time   EQU       36        ; Wie oft warten?
Mask     EQU       1
Carry    EQU       3,0
Zero     EQU       3,2
Bank     EQU       3,5
Indirect EQU       0
FSR      EQU       4

 ; Initialisierung
         CLRF      Counter   ; Counter = NULL
         CLRF      Time      ; Time = NULL
         CLRF      Wait      ; Wait = NULL

         MOVLW     Save      ; Lade Wert von Save
         MOVWF     FSR       ; Init FSR

         CALL      Ruhe      ; UP Ruhe aufrufen

 ; Scheduling-Schleife
START
         MOVLW     Mask      ; Lade Maske
         ANDWF     Time,W    ; Maskieren
         BTFSS     Zero
         GOTO      Ausgabe   ; UP Ausgabe
         GOTO      Eingabe   ; UP Eingabe

WEITER
         INCF      Time      ; Time ++
         GOTO      Start     ; Schleife

 ; Unterprogramm Eingabe
EINGABE
         BTFSC     In,7
         GOTO      Weiter    ; UP beenden
         MOVLW     8         ; 8 in W laden
         MOVWF     Counter   ; Counter mit 8 laden
         INCF      FSR       ; Nächstes Register
         CALL      Warten    ; UP Warten

READ
         CALL      Warten
         CALL      Warten
         MOVF      In,W      ; Lade Eingang in W
         ANDLW     1
         BTFSC     Zero      ; Wenn 0
         BSF       Carry     ; Setze Carry
         BTFSS     Zero      ; Wenn 1
         BCF       Carry     ; Lösche Carry
         RRF       Indirect  ; Rotiere Carry in Speicherplatz
         DECFSZ    Counter   ; Counter --
```

```
         GOTO      Read      ; Nächstes Bit einlesen
         GOTO      Weiter    ; UP beenden

 ; Unterprogramm Ausgabe
AUSGABE
         GOTO      Weiter    ; UP beenden

 ; Unterprogramm Warten (104 ms)
WARTEN
         MOVLW     w_time    ; Wartezeit laden
         MOVWF     Wait      ; Wartezeit

LOOP
         DECFSZ    Wait      ; Wait --
         GOTO      LOOP
         RETURN

 ; Unterprogramm Ruhepegel
RUHE
         MOVLW     9         ; 9 in W laden
         MOVWF     Counter   ; Counter laden

LOOP2
         BTFSS     In,7      ; Prüfe Eingang == 1
         GOTO      Ruhe      ; neu beginnen
         DECFSZ    Counter   ; Counter --
         GOTO      Ruhe2     ; 208ms warten
         RETURN              ; UP beenden

RUHE2
         CALL      Warten
         CALL      Warten
         GOTO      LOOP2

         END
```

---

mux: Fahrradtacho, UP Torzeit vorhanden,
Ausgabe 7-seg kodiert (keine Garantie für Richtigkeit!!)

```
 ; Fahrradtacho
 ; Ausgabe auf 3 7-Segment-Anzeigen

         device    16C83

 ; Variablendeklaration
Counter  EQU       10h       ; Impulszähler: 0 - 20cm, +1 - 1m, +2 - 10m, +3 -
100m, +4 - 1km, +5 - 10km
Timer    EQU       16h       ; Timer für den Schedule-Algorithmus
BWS      EQU       18h       ; Bildwiederholspeicher 0 - aPegel, +1 - 100m, +2 -
1km, +3 - Pointer, +4 - 10km
aPegel   EQU       18h       ; alter Pegel für Flankenerkennung
```

```
Pointer  EQU       1Bh       ; Pointer auf die aktuelle Ausgabestelle (7-Seg)
JJ       EQU       1Dh       ; Schleifenzähler
Dotted   EQU       1Eh       ; Den Dezimalpunkt nach der Zahl anzeigen
var_1sL  EQU       ...       ; Wie viele Schleifendurchläufe entsprechen 1s (für
TORZEIT) - Low Byte
var_1sH  EQU       ...       ; High Byte
F2_maskL EQU       10h       ; Maske, wann Funktion 2 ausgeführt werden soll -
Low Byte
F2_maskH EQU       00h       ; High Byte
F3_maskL EQU       0D0h      ; Maske, wann Funktion 3 ausgeführt werden soll -
Low Byte
F3_maskH EQU       00h       ; High Byte
PortA    EQU       5         ; Port A
PortB    EQU       6         ; Port B
Zero     EQU       3,2       ; Zero-Bit
Carry    EQU       3,0       ; Carry-Bit
Page     EQU       3,5       ; Bit zum Umschalten der Bank
FSR      EQU       4         ; File Select Register
INDIRECT EQU       0         ; Indirekte Adressierung

         ORG       0

 ; Initialisierung
INIT
         BSF       Page      ; Springe zu Bank 1
         MOVLW     00001000b ; Maske für unten
         MOVWF     PortA     ; Setze RA3 als Eingang, Rest Ausgang
         CLRF      PortB     ; Port B komplett Ausgang
         BCF       Page      ; Zurück zu Bank 0
         MOVLW     1         ; Pointer mit 1 initialisieren
         MOVWF     Pointer   ; siehe oben
         CLRF      Timer     ; Timer = NULL
         CLRF      Timer+1
         MOVF      PortA,W   ; Port A einlesen
         MOVWF     aPegel    ; in aPegel speichern
         CLRF      Counter   ; Counter = NULL
         CLRF      Counter+1
         CLRF      Counter+2
         CLRF      Counter+3
         CLRF      Counter+4
         CLRF      Counter+5
         CLRF      BWS+1     ; BWS = NULL
         CLRF      BWS+2
         CLRF      BWS+4

 ; Hauptschleife (Scheduler)
LOOP
         INCF      Timer     ; Timer inkrementieren
         BTFSC     Zero      ; Zero-Bit abfragen (Überlauf)
         INCF      Timer+1   ; High-Byte inkrementieren
         CALL      Impuls    ; Unterprogramm Impuls aufrufen (bei jedem
Schleifendurchlauf)
```

```
        MOVLW   F2_maskL  ; Lade Low Byte der Maske
        ANDWF   Timer,W   ; AND Verknüpfung mit dem Low Byte des Timers
        BTFSS   Zero      ; Überprüfe Zero-Bit == 1
        GOTO    LOOP1
        MOVLW   F2_maskH  ; Lade High Byte der Maske
        ANDWF   Timer+1,W ; AND Verknüpfung mit dem High Byte des Timers
        BTFSC   Zero      ; Überprüfe Zero Bit == 0
        CALL    Mux       ; Unterprogramm MUX aufrufen

LOOP1
        MOVLW   F3_maskL  ; Lade Low Byte der Maske
        ANDWF   Timer,W   ; AND Verknüpfung mit dem Low Byte des Timers
        BTFSS   Zero      ; Überprüfe Zero Bit == 1
        GOTO    LOOP2
        MOVLW   F3_maskH  ; Lade High Byte der Maske
        ANDWF   Timer+1,W ; AND Verknüpfung mit dem High Byte des Timers
        BTFSC   Zero      ; Überprüfe Zero Bit == 0
        CALL    Torzeit   ; Unterprogramm TORZEIT

LOOP2
        GOTO    LOOP      ; Hauptschleife

 ; Unterprogramm IMPULS
IMPULS
        MOVF    aPegel,W  ; alten Pegel in W laden
        XORWF   PortA,W   ; XOR Verknüpfung von Port A mit dem alten Pegel
        ANDLW   00001000b ; Maskiere um Bit3 zu erhalten
        BTFSC   Zero      ; Überprüfe Zero Bit == 0
        RETURN            ; Beende Unterprogramm
        MOVLW   00001000b ; Maske
        XORWF   aPegel    ; Invertiere den alten Pegel

        MOVLW   6         ; Initialisiere Schleifenzähler
        MOVWF   JJ        ; Speichere Schleifenzähler
        MOVLW   Counter   ; Lade die Adresse von Counter
        MOVWF   FSR       ; Speichere FSR

 ; Schleife zur Überprüfung auf Überlauf
IMPULS1
        INCF    INDIRECT  ; Inkrementiere Register für indirekte Adressierung
        MOVLW   10        ; Lade 10 in W
        SUBWF   INDIRECT,W; Subtrahiere 10 vom indirekt adressierten Register
        BTFSS   Zero      ; Überprüfe Zero Bit == 1
        RETURN            ; Beende Unterprogramm
        CLRF    INDIRECT  ; Setze indirekt adressiertes Register auf 0
        INCF    FSR       ; Inkrementiere FSR für die nächste Stelle
        DECFSZ  JJ        ; Dekrementiere Schleifenzähler und prüfe auf 0
        GOTO    IMPULS1
        RETURN            ; Beende Unterprogramm

 ; Unterprogramm MUX
MUX
```

```
        BCF     Carry     ; Lösche Carry
        BTFSC   Pointer,2 ; Überprüfe ob 2. Bit im Pointer == 0
        BSF     Carry     ; Setze Carry
        RLF     Pointer   ; Rotiere nach links über Carry
        MOVLW   BWS       ; Lade Adresse von BWS in W
        ADDWF   Pointer,W ; Addiere die aktuelle Pointer-Adresse
        MOVWF   FSR       ; Speicher Adresse in FSR
        MOVF    INDIRECT,W; Hole Inhalt aus indirekter Adresse
        MOVWF   PortB     ; Gebe Inhalt auf Port B aus
        MOVF    Pointer,W ; Lade Pointer in W
        MOVWF   PortA     ; Gebe aktuellen Pointer an Port A aus
        RETURN            ; Beende Unterprogramm

 ; Unterprogramm TORZEIT
TORZEIT
        MOVLW   var_1sL   ; Lade Low Byte der Maske
        ANDWF   Timer,W   ; AND Verknüpfung mit dem Low Byte des Timers
        BTFSS   Zero      ; Überprüfe Zero Bit == 1
        RETURN            ; Beende Unterprogramm
        MOVLW   var_1sH   ; Lade High Byte der Maske
        ANDWF   Timer+1,W ; AND Verknüpfung mit dem High Byte des Timers
        BTFSS   Zero      ; Überprüfe Zero Bit == 0
        RETURN            ; Beende Unterprogramm
        MOVF    Counter+3,W ; Lade 100m - Speicher in W
        CALL    Table     ; Hole 7-Segment Kodierung aus der Tabelle
        MOVWF   BWS+1     ; Speichere 100m in BWS
        MOVF    Counter+4,W ; Lade 1km - Speicher in W
        CALL    Table     ; Hole 7-Segment Kodierung aus der Tabelle
        ADDLW   80h       ; Zeige zusätzlich den Dezimalpunkt an
        MOVWF   BWS+2     ; Speichere 1km in BWS
        MOVF    Counter+5,W ; Lade 10km - Speicher in W
        CALL    Table     ; Hole 7-Segment Kodierung aus der Tabelle
        MOVWF   BWS+4     ; Speichere 10km in BWS
        CLRF    Timer     ; Timer = NULL
        CLRF    Timer+1
        RETURN            ; Beende Unterprogramm

 ; Tabelle
TABLE
        ADDWF   PCL       ; Addiere W zum Programmzähler (springe in der
Tabelle)
        RETLW   3Fh       ; 7-Seg: 0
        RETLW   06h       ; 7-Seg: 1
        RETLW   5Bh       ; 7-Seg: 2
        RETLW   4Fh       ; 7-Seg: 3
        RETLW   66h       ; 7-Seg: 4
        RETLW   6Dh       ; 7-Seg: 5
        RETLW   7Dh       ; 7-Seg: 6
        RETLW   07h       ; 7-Seg: 7
        RETLW   7Fh       ; 7-Seg: 8
        RETLW   0DFh      ; 7-Seg: 9
```

Alle 20 zusammen.txt

END