```
;*****************************************************************
;*                                                               *
;* Title       : DCF                                             *
;* Version     : 1.3    28. September 2006                       *
;* Author      : Uwe Nagel, Parkstr. 46, D-68766 Hockenheim      *
;* Assembler   : MPASM (c) Microchip                             *
;*                                                               *
;*****************************************************************
; 0.1    21.05.02 erster Entwurf
; 0.2    30.05.02 erste lauffähige Version mit LCD
; 0.3    29.09.02 Kosmetik am Quellcode und an der Homepage, keine
;                 Änderung der Funktion
; 0.4    01.04.03 Antennen Symbol auf Wunsch von Christof Rueß
; 0.5    24.07.03 Zeitimpule auf Wunsch von CIOVALLADOLID <CIOVALLADOLID@terra.es>
;                 wahlweise zwei feste Schaltausgänge auf Wunsch von Herrn Eichler
; 0.6    30.08.03 Ports löschen beim Einschalten
;                 PTT-Signal auf PORTA,4 (CIOVALLADOLID)
; 0.7    26.09.03 Filter gegen unsichere Flanken
; 0.8    01.10.03 diverse Optimierungen
; 0.9    15.08.04 anderes Ausgabeformat auf Wunsch von Peter Hilkmann (PE1DCD)
; 0.9a   02.11.04 Fehler bei Monatsausgabe beseitigt, neue Sprachen
; 0.a    06.12.04 Spezialversion mit 16F874 und 4MHz (hier nicht integriert)
; 0.b    03.04.05 serielle Ausgabe der Zeit, Sprachen ausgelagert
; 0.ba   23.10.05 neue Sprache 'swedish', Dank an Volker Lange-Janson
; 0.bb   09.02.06 fehlendes 'RETURN' eingefügt, Dank an Peter Engels
; 0.bc   10.02.06 Antennensymbol auch bei PE1DCD ermöglichen
;                 Programm adaptiert auf 16F628 (Nachfolger 16F84)
; 0.bd   21.02.06 useTimer1 eingeführt
; 0.c    18.03.06 erster Versuch mit Nebenuhr
; 1.0    08.04.06 Hauptuhr mit Sommer-Winterzeit-Umstellung
; 1.1    19.06.06 andere Impulsausgabe für Mutteruhr (PULS500)
; 1.2    02.09.06 blinkendes Antennensymbol
; 1.3    28.09.06 Plausibilitätskontrolle von Puls- und Pausenzeiten
; 1.3a   09.11.06 Fehlerkorrektur MOVLF bei 16f628 raus
;*****************************************************************


#define TITEL "-DCF77-Uhr 1.3a-"

;--- Sprache der Wochentage und Monate
; wenn nichts definiert ist: deutsch
;#define language dutch.inc
;#define language spanish.inc
;#define language english.inc
;#define language french.inc
;#define language italian.inc
;#define language swedish.inc

;--- useRS232 definieren um Ausgabe auf RS232 einzuschalten
;#define useRS232
;--- hier die Portleitung für die RS232-Ausgabe definieren
#define RS232PORT PORTA,1
;--- Pegel ist invertiert, so dass man einen PC ohne MAX232 anschliessen kann.
;* Mit MAX232 muss man bei TxHigh bsf und bei TxLow bcf schreiben.
#define TxHigh bcf RS232PORT
#define TxLow bsf RS232PORT
;*****************************************************************
;* RS232 Ausgabe:
;* Format 095127,190305,6<cr><lf>
;* bedeutet: 'Samstag 19.03.05 09:51:27'
;* beim Wochentag bedeutet 1=Montag .. 7=Sonntag
;* seriellen Format: 9600 Bit/s 8n1
;*****************************************************************

;--- useTimer1 definieren, um Timer1 in 16F628 oder anderen neuen PICs
; zu verwenden. Dann muss ein 4MHz Quarz angeschlossen werden, nicht 3,2768MHz
; geht nicht mit PIC16F84!
#define useTimer1

;--- BAHN definieren, um den Eingang auf PORTB,0 statt auf PORTA,0 zu haben
; für mich, wegen Mutteruhr-Platine
;#define BAHN

;--- alterText definieren, um Monate als Zahl und Jahr 4-stellig anzuzeigen,
; sonst wird der Monat als 3-stellige Abkürzung gezeigt, das Jahr 2-stellig
;#define alterText
```

```
;--- PE1DCD definieren, für lange Wochentage auf 2x20-Display
;#define PE1DCD

;--- DCFSYMBOL definieren für Antennensymbol
; angegeben wird die Cursorposition, oder 0 für kein Symbol
#define DCFSYMBOL 0x8e
;#define DCFSYMBOL 0

;--- CIOVALLADOLID definieren, um Sekundenimpulse und
; Zeitzeichen, wie im Radio, zu erzeugen
;verträgt sich nicht mit useRS232 wegen Portleitung !!
;#define     CIOVALLADOLID

;--- EICHLER definieren um die beiden Schaltzeiten ( Unterprogramm alarm )
; zu aktivieren. Es dürfen nicht CIOVALLADOLID und EICHLER gleichzeitig
; aktiv sein, da die gleichen Portleitungen verwendet werden
;#define EICHLER

;--- MUTTERUHR definieren um auf PORTA,3 einen Minutentakt zum Ansteuern
; von Nebenuhren auszugeben. PORTA,2 muss dazu auf 1 liegen. Zieht man PORTA,2
;  auf Masse, dann wird jede Sekunde ein Takt zum Stellen ausgegeben.
; !!! experimentell !!!
;#define MUTTERUHR
; wenn PULS500 definiert ist, wird jede Minute ein 500ms-Puls ausgegeben
; sonst wechselt PORTA,3 jede Minute den Pegel
;#define PULS500


      title        TITEL

#ifdef __16F84
; configs für 16F84
      __config     _XT_OSC & _WDT_ON & _PWRTE_ON & _CP_OFF
      include      <P16F84.INC>
#endif

#ifdef __16F84A
; configs für 16F84A
      __config     _XT_OSC & _WDT_ON & _PWRTE_ON & _CP_OFF
      include      <P16F84A.INC>
#endif

#ifdef __16F628
; configs für 16F628
      __config     _XT_OSC & _WDT_ON & _PWRTE_ON & _CP_OFF & _DATA_CP_ON & _MCLRE_OFF &
_LVP_OFF & _BODEN_ON
      include      <p16f628.inc>
#endif

      errorlevel   -302         ; keine bank select message
      radix  dec

;****** Variablen ************************************************
 cblock      0x20
      save_W        ; w und
      save_STATUS   ; STATUS während interrupt
      save_FSR      ; FSR während Interrupt
      flag          ; diverse Bits
      temp
      tmp1
      tmp2
      zweihund      ; 200 Interupts sind 1 Sek
      puls_zeit     ; misst Pulsdauer
      pause_zeit    ; misst Pausendauer
;*** diese Zeit wird angezeigt
      dispSec, dispMin, dispStd, dispTag, dispMon, dispJahr, dispWtag, dispStat
;*** hier kommt die gerade empfangene Zeit rein
      dcfMin, dcfStd, dcfTag, dcfMon, dcfJahr, dcfWtag, dcfStat
;*** zum Vergleich, die letzte Minute
      dcf1Min, dcf1Std, dcf1Tag, dcf1Mon, dcf1Jahr, dcf1Wtag, dcf1Stat
      filter, altsec, flag2
 endc
#ifdef CIOVALLADOLID
 cblock
      SekPuls, BeepPuls, beeps
 endc
```

```
#endif
#ifdef useRS232
 cblock
       rs232state, rs232ptr, rs232tx, rs232tmp, rs232bitcnt
 endc
#endif
#ifdef MUTTERUHR
 cblock
       pulshi, pulslo, pulsdauer
 endc
#endif


;****** bits in flag *************************************************
#define       neusync flag,0
#define       alt     flag,1
#define       sekunde flag,2
#define       syncd   flag,3
#define       dcfsign flag,4
#define       ms5     flag,5
#define       comm    flag,7

#define SommerWinter       flag2,0
#define WinterSommer       flag2,1
#define Pulsstart          flag2,2
#define Pulsende           flag2,3

lc_data       equ tmp1        ; local in lcd_write
lcd_init_cnt  equ tmp2        ; only used in lcd_init
temp_lcd      equ tmp2        ; temporary in write_lcd_char, clear_display
temp_byte     equ tmp2        ; temporary in write_byte


;****** Konstanten ***************************************************
; TICKS berechnet sich folgendermassen (für Timer0):
; Oszillatorfrequenz/4/256/Prescaler/TICKS=1 Hz
; oder
; TICKS=Oszillatorfrequenz/4/256/Prescaler
; wenn Ticks nicht ganzzahlig ist,
; geht die Uhr im nicht synchronisierten Betrieb ungenau
; bei useTimer1 und 4MHz muss auch 200 stehen
#define       TICKS  200    ; für 3,2768 MHz Quarz bei 16F84 4MHz bei 16F628
;#define      TICKS  218    ; für NTSC Quarz

;*********************************************************************
;* Programmcode ...
;*********************************************************************
       org    0
       goto   main

;*********************************************************************
;* interrupt :
;*********************************************************************
       org    4
       goto   interrupt
;*********************************************************************
;* Tabellen in Page 0:
;*********************************************************************
lcd_string_table
       addwf  PCL,f
lcd_init_table
       retlw  0x02 | 0x80   ; switch to 4 bit mode
       retlw  0x28 | 0x80   ; better do it twice
       retlw  0x28 | 0x80   ; init for 2x16 Display
       retlw  0x0c | 0x80   ; Display on, Cursor off, no blinking
       retlw  0x06 | 0x80   ; increment cursor, no display shift
       retlw  0x01 | 0x80   ; Clear Display, Cursor home
       dt     TITEL         ; Einschaltmeldung

#if DCFSYMBOL>0
; Antennensymbol definieren
       retlw  0x40 | 0x80   ; Pointer auf CG-Ram

       retlw  0x00   ; .....
       retlw  0x00   ; .....
       retlw  0x00   ; .....
```

```
        retlw  0x03   ; ...##
        retlw  0x04   ; ..#..
        retlw  0x01   ; ....#
        retlw  0x00   ; .....
        retlw  0x00   ; .....

        retlw  0x00   ; .....
        retlw  0x00   ; .....
        retlw  0x00   ; .....
        retlw  0x18   ; ##...
        retlw  0x04   ; ..#..
        retlw  0x02   ; ...#.
        retlw  0x11   ; #...#
        retlw  0x09   ; .#..#

        retlw  0x06   ; ..##.
        retlw  0x01   ; ....#
        retlw  0x05   ; ..#.#
        retlw  0x04   ; ..#..
        retlw  0x0e   ; .###.
        retlw  0x0e   ; .###.
        retlw  0x1f   ; #####
        retlw  0x1f   ; #####

        retlw  0x09   ; .#..#
        retlw  0x01   ; ....#
        retlw  0x02   ; ...#.
        retlw  0x00   ; .....
        retlw  0x00   ; .....
        retlw  0x00   ; .....
        retlw  0x00   ; .....
        retlw  0x00   ; .....

        retlw  0x06   ; ..##.
        retlw  0x01   ; ....#
        retlw  0x05   ; ..#.#
        retlw  0x04   ; ..#..
        retlw  0x0a   ; .#.#.
        retlw  0x0a   ; .#.#.
        retlw  0x11   ; #...#
        retlw  0x1f   ; #####

#endif
; define the character 'å'
        retlw  0x68 | 0x80   ; Pointer auf CG-Ram

        retlw  0x06   ; ..##.
        retlw  0x06   ; ..##.
        retlw  0x00   ; .....
        retlw  0x0e   ; .###.
        retlw  0x01   ; ....#
        retlw  0x0f   ; .####
        retlw  0x11   ; #...#
        retlw  0x0f   ; .####

        retlw  0xff          ; Ende der Tabelle

#ifndef language

 ifdef PE1DCD
lcd_mesz_table
        dt     "Sommerzeit",0xff
lcd_mez_table
        dt     "Winterzeit",0xff
 else
lcd_mesz_table
        dt     "MESZ ",0xff
lcd_mez_table
        dt     "MEZ  ",0xff
 endif ; PE1DCD

wtag_table
 ifdef PE1DCD
        dt     "---------",0xff
        dt     "Montag   ",0xff
```

```
        dt      "Dienstag ",0xff
        dt      "Mittwoch ",0xff
        dt      "Donnersta",0xff    ; zu kurz ;-(
        dt      "Freitag  ",0xff
        dt      "Samstag  ",0xff
        dt      "Sonntag  ",0xff
 else
        addwf   PCL,f
        dt      "--MoDiMiDoFrSaSo"  ; deutsch
 endif

month_table
        addwf   PCL,f
        dt      "---JanFebM",0xe1,"rAprMaiJunJulAugSepOktNovDez"

#else
        include language
#endif

;****** serielle Ausgabe ****************************************
;* 9600 Bit/s bei 3,2768MHz sind das 85,333 Takte pro Bit
;* mit 85 Takten ist der Fehler hinreichend klein
;* bei 4MHz müssen es 104 Takte sein
;*****************************************************************
#ifdef useRS232
rs232
        movlw   dispStd
        movwf   rs232ptr
        clrf    rs232state
        return

rs232service
        bcf     ms5
        movf    rs232state,w
        addwf   PCL,f
        goto    rs_state0
        goto    rs_state1
        goto    rs_state2
        goto    rs_state3
        goto    rs_state4
        goto    rs_state5
        goto    rs_state6

rs_state0
        call    rs232byte     ; Uhrzeit ausgeben
        decf    rs232ptr,f
        movf    rs232ptr,w
        xorlw   dispSec-1
        skpz
        return
        incf    rs232state,f
        movlw   dispTag
        movwf   rs232ptr
rs_state6                       ; nichts mehr tun, bis zum Ende der Sekunde
        return

rs_state1                       ; Komma zwischen den Feldern
rs_state3
        movlw   ','
        incf    rs232state,f
        goto    rs232char

rs_state2                       ; Datum ausgeben
        call    rs232byte
        incf    rs232ptr,f
        movf    rs232ptr,w
        xorlw   dispWtag
        skpnz
        incf    rs232state,f
        return

rs_state4                       ; Wochentag ausgeben
        incf    rs232state,f
        movf    rs232ptr,w
        movwf   FSR
```

```
       goto   rs232nibble

rs_state5                       ; CR/LF zum Abschluss
 if rs_state5>0xff
   error "Tabellen zu lang. Alle Sprungziele müssen unter 0x100 sein"
 endif
       incf   rs232state,f
       movlw  0x0d
       call   rs232char
       movlw  0x0a
       goto   rs232char


rs232byte
       movf   rs232ptr,w
       movwf  FSR
       swapf         INDF,w
       andlw  0x0f
       iorlw  0x30
       call   rs232char
rs232nibble
       movf   INDF,w
       andlw  0x0f
       iorlw  0x30
rs232char
       movwf  rs232tx
 TxLow
       movlw  9
       movwf  rs232bitcnt
       nop
       nop
       nop
       nop
       nop
       nop
rsbitloop                  ; Schleife hat 76+9 {95+9}
       call   bitdelay     ; 76 {95} Takte mit call
       btfss  rs232tx,0
 TxLow
       btfsc  rs232tx,0
 TxHigh
       bsf    STATUS,C
       rrf           rs232tx,f
       decfsz        rs232bitcnt,f
       goto          rsbitloop

bitdelay      ; 74 Takte (3*(23-1)+8)  {3*(29-1)+9}
#ifdef useTimer1
       movlw  29
       nop
#else
       movlw  23
#endif
       movwf  rs232tmp
       clrwdt
       nop
delayloop
       decfsz rs232tmp,f
       goto   delayloop
       return

#endif
;*****************************************************************
;* interrupt  service Routine,
;* hinter Tabellen, damit die alle in Page 0 liegen können:
;*****************************************************************
interrupt
       movwf  save_W        ; W und STATUS sichern
       swapf  STATUS,w
       movwf  save_STATUS
       movf   FSR,w
       movwf  save_FSR

       bcf    STATUS,RP0    ; Wähle register page 0
#ifdef useTimer1
       btfss  PIR1,CCP1IF
```

```
#else
       btfss  INTCON,T0IF
#endif
       goto   no_timer_isr ; ist es timer interrupt ?


#ifdef CIOVALLADOLID
       movf   SekPuls,w
       btfsc  STATUS,Z
       goto   EndPuls1

       decf   SekPuls,f
       goto   noPuls1
EndPuls1
       bcf    PORTA,1
       bcf    PORTA,2
noPuls1
       movf   BeepPuls,w
       btfsc  STATUS,Z
       goto   EndPuls2

       decf   BeepPuls,f
       goto   noPuls2
EndPuls2
       bcf    PORTA,3
noPuls2
#endif

#ifdef PULS500
       movf   pulsdauer,w
       btfsc  STATUS,Z
       bcf    PORTA,3
       btfss  STATUS,Z
       decf   pulsdauer,f
#endif


; increment filter if port=1 and filter if less than 8
; decrement filter if port=0 and filter if greater than 0

#ifdef BAHN
       btfss  PORTB,0
#else
       btfss  PORTA,0
#endif
       goto   port_is_0
port_is_1
       incf   filter,f
       btfsc  filter,3     ; <8 ?
port_is_0
       decf   filter,f
       btfsc  filter,7
       clrf   filter

; set dcfsign if filter becomes 6
; clear dcfsign if filter becomes 1
; 0 1 2 3 4 5 6 7   6 5 4 3 2 1 0
; 0 0 0 0 0 0 1 1   1 1 1 1 1 0 0

       movf   filter,w
       xorlw  6
       btfsc  STATUS,Z
       bsf    dcfsign
       xorlw  6^1
       btfsc  STATUS,Z
       bcf    dcfsign

;** teste auf steigende Flanke
       btfss  alt
       btfss  dcfsign
       goto   NoPos

;** Pause kürzer als 750ms ist ungültig
       movf   pause_zeit,w
       addlw  -150         ; >150*5ms=750ms
       btfss  STATUS,C
```

```
        goto    NoPos

;** positive Flanke...
        bsf     alt
        bsf     Pulsstart
        clrf    puls_zeit

        btfss   neusync         ; erster Impuls nach 59. Sekunde?
        goto    NoNeg

        clrf    dispSec
        movf    dcfMin,w
        movwf   dispMin
        movf    dcfStd,w
        movwf   dispStd
        movf    dcfTag,w
        movwf   dispTag
        movf    dcfWtag,w
        movwf   dispWtag
        movf    dcfMon,w
        movwf   dispMon
        movf    dcfJahr,w
        movwf   dispJahr
        movf    dcfStat,w
        movwf   dispStat

        bcf     neusync
        movlw   TICKS  ; 200 * 5ms
        movwf   zweihund
        goto    end_timer_isr

NoPos:
;** teste auf fallende Flanke
        btfsc   alt
        btfsc   dcfsign
        goto    NoNeg

;** Puls kürzer als 80ms ist ungültig
        movf    puls_zeit,w
        addlw   -16             ; >16*5ms=80ms
        btfss   STATUS,C
        goto    NoNeg

;** fallende Flanke
        bcf     alt
        clrf    pause_zeit
        bsf     Pulsende

;** teste ob 0 (<150ms) oder lang (>150ms)
        movf    puls_zeit,w
        addlw   -30             ; >30*5ms
        call    shiftDCF

NoNeg:
        incfsz  pause_zeit,w  ; bis max 255 zählen damit
        incf    pause_zeit,f  ; bei längeren Pausen nicht wieder
                              ; bei 0 angefangen wird

        decfsz  zweihund,f
        goto    TimeOut

;** Sekunde abgelaufen
        movlw   TICKS
        movwf   zweihund
        call    tick

TimeOut:
        incfsz  puls_zeit,f   ; 1,28 s keine steigende Flanke ?
        goto    end_timer_isr

        call    sec59         ; überprüfe empfangene Zeit

end_timer_isr
        bcf     sekunde
        movf    dispSec,w
```

```
        xorwf  altsec,w      ; hat sich die Sekunde geändert ?
        skpz
        bsf    sekunde
        xorwf  altsec,f      ; aktuelle Sekunde merken


#ifdef useRS232
        bsf    ms5
#endif
#ifdef useTimer1
        bcf    PIR1,CCP1IF     ; lösche timer interrupt flag
#else
        bcf    INTCON,T0IF     ; lösche timer interrupt flag
#endif
no_timer_isr
        movf   save_FSR,w
        movwf  FSR
        swapf  save_STATUS,w
        movwf  STATUS
        swapf  save_W,f
        swapf  save_W,w
        retfie




;*****************************************************************
;* Hauptprogramm :
;*****************************************************************
main
#ifdef __16F628
        movlw  0x07
        movwf  CMCON  ; disable comparators (= enable Port A  16F628)
#endif
#ifdef useTimer1
        clrf   TMR1L
        clrf   TMR1H
        movlw  0x31
        movwf  T1CON  ; prescaler 8, timer 1 enabled
        movlw  0x0b
        movwf  CCP1CON       ; compare mode, trigger special event
        movlw  low 624
        movwf  CCPR1L  ; Fosz/4/(prescaler8)/(200Hz)-1
        movlw  high 624
        movwf  CCPR1H
        bsf    STATUS,RP0    ; select Register-Page 1
        movlw  0x04
        movwf  PIE1   ; enable Compare Interrupt
#endif
        bsf    STATUS,RP0    ; select Register-Page 1
        movlw  0x03          ; TMR0 internal clock/16 = 200Hz @ 3,2768MHz Quarz
        movwf  OPTION_REG
#ifdef BAHN
        movlw  0x04          ; PORTA2 input, Rest output;
        movwf  TRISA
        movlw  0x01          ; PORTB7..1 output, PORTB,0 input
        movwf  TRISB
#else
#ifdef MUTTERUHR
        movlw  0x05          ; PORTA4 3 1 output; PORTA2 und PORTA0 input
        movwf  TRISA
#else
        movlw  0x01          ; PORTA4-PORTA1 output; PORTA0 input
        movwf  TRISA
#endif
        movlw  0x00          ; PORTB7..0 output
        movwf  TRISB
#endif
        bcf    STATUS,RP0    ; select Register-Page 0


        clrf   PORTA
        clrf   PORTB
        clrf   PCLATH
```

```
        clrf    flag
        clrf    puls_zeit
        movlw   TICKS
        movwf   zweihund

        clrf    dispSec
        clrf    dispMin
        clrf    dispStd
        clrf    dispWtag
        clrf    dispJahr
        movlw   1
        movwf   dispMon
        movwf   dispTag

        clrf    flag2

#ifdef Mutteruhr
        clrf    pulslo
        clrf    pulshi
#endif

#ifdef CIOVALLADOLID
        clrf    SekPuls
        clrf    BeepPuls
        clrf    beeps
#endif

#ifdef useRS232
        call    rs232           ; einmal aufrufen, um Variablen zu initialisieren
        TxHigh                  ; Stop-Pegel
#endif

#ifdef useTimer1
        movlw   0xc0            ; Enable  Peripherial Int
#else
        movlw   0xa0            ; Enable  TMR0-Int
#endif
        movwf   INTCON

;------ Display initialisieren und Einschaltmeldung ------
        call    lcd_init        ; init LC-Display
hallo:  clrwdt
        btfss   sekunde
        goto    hallo

        movlw   0x01            ; Clear Display
        call    write_lcd_comm
;------
mainloop:
#ifdef useRS232
        btfsc   ms5
        call    rs232service
#endif
        clrwdt

;####
#if DCFSYMBOL>0
        btfsc   Pulsstart
        call    AntenneEin
        btfsc   Pulsende
        call    AntenneAus
#endif
;####

        btfss   sekunde
        goto    mainloop

        bcf     sekunde
        call    showtime
        call    special
#ifdef useRS232
        call    rs232
#endif
        goto    mainloop
```

```
#if DCFSYMBOL>0
AntenneEin:
       bcf    Pulsstart
       movlw  DCFSYMBOL     ;0x8e
       call   write_lcd_comm
       movlw  0x00
       call   write_lcd_data
       movlw  0x01
       call   write_lcd_data
       movlw  DCFSYMBOL+0x40       ;0xce eine Zeile tiefer
       call   write_lcd_comm
       movlw  0x02
       btfss  syncd
       movlw  0x04
       call   write_lcd_data
       movlw  0x03
       goto   write_lcd_data
AntenneAus:
       bcf    Pulsende
       movlw  DCFSYMBOL     ;0x8e
       call   write_lcd_comm
       movlw  ' '
       call   write_lcd_data
       movlw  ' '
       call   write_lcd_data
       movlw  DCFSYMBOL+0x40       ;0xce eine Zeile tiefer
       call   write_lcd_comm
       movlw  0x02
       btfss  syncd
       movlw  0x04
       call   write_lcd_data
       movlw  ' '
       goto   write_lcd_data
#endif

;*********************************************************************
;* Spezialprogramme für verschiedene Zwecke
;*********************************************************************
special
#ifdef EICHLER
;****** Zeitvergleich ************************************************
       movf   dispSec,w
       skpz
       return              ; nur zu beginn einer Minute testen

       movf   dispWtag,w
       andlw  0x06
       xorlw  0x06
       skpnz               ; an Tagen 6 und 7 ist Wochenende
       return

       movf   dispStd,w
       xorlw  0x09
       skpz                ; alle Schaltzeiten in der neunten Stunde
       return

       movf   dispMin,w
       skpnz
       bsf    PORTA,2              ; um 9:00 Uhr PORTA,2 an

       xorlw  0x15
       skpnz
       bsf    PORTA,3              ; um 9:15 Uhr PORTA,3 an

       xorlw  0x15 ^ 0x30
       skpz
       return

       bcf    PORTA,2              ; um 9:30 Uhr PORTA,2 aus
       bcf    PORTA,3              ; und PORTA,3 aus

       return

#else
#ifdef CIOVALLADOLID
```

```
;****** Impulsausgabe ********************************************
special      movlw  60
       movwf  SekPuls
       bsf    PORTA,1             ; jede Sekunde 300ms Puls an PORTA,1

       movf   dispSec,w
       xorlw  0x02
       skpnz
       bcf    PORTA,4             ; PTT ab 2. Sekunde aus, egal welche Minute

       movf   dispSec,w
       iorwf  dispMin,w
       iorwf  dispStd,w
       skpnz
       bsf    PORTA,2             ; um 0:00:00 300ms Puls auf PORTA,2

       movf   dispMin,w
       xorlw  0x29          ; in 29.
       jz     Beep1
       xorlw  0x29^0x59     ; und 59. Minute
       jnz    CIO_1

Beep1  movf   dispSec,w     ; ab Sekunde 53 PTT an
       xorlw  0x53
       skpnz
       bsf    PORTA,4

       movf   dispSec,w
       xorlw  0x55          ; und ab der 55. Sekunde
       jnz    CIO_1

       movlw  6             ; 6 mal piepsen
       movwf  beeps

CIO_1  movf   beeps,w
       jz     EndCIO
       movlw  100           ; der letzte 500ms
       decfsz beeps,f
       movlw  20            ; die anderen 100ms
       movwf  BeepPuls
       bsf    PORTA,3
EndCIO return
#else
#ifdef MUTTERUHR
       btfss  WinterSommer
       goto   Mutter1
       bcf    WinterSommer
       movlw  low 62
       movwf  pulslo
       movlw  high 62
       movwf  pulshi
Mutter1
       btfss  SommerWinter
       goto   Mutter2
       bcf    SommerWinter
       movlw  low 672
       movwf  pulslo
       movlw  high 672
       movwf  pulshi
Mutter2
       movf   pulslo,w
       iorwf  pulshi,w
       skpnz
       goto   Mutter3

       decf   pulslo,f
       incf   pulslo,w
       skpnz
       decf   pulshi,f

       goto   stellen

Mutter3

       btfss  PORTA,2
```

```
        goto    stellen

        movf    dispSec,w
        skpz
        return

stellen
        movlw   0x08
        xorwf   PORTA,f
#ifdef PULS500
        movlw   100
        movwf   pulsdauer
#endif
        return
#else
        return
#endif
#endif
#endif

;*********************************************************************
;****** neue Sekunde anzeigen ****************************************
;*********************************************************************
showtime:
#ifdef PE1DCD
;  01234567890123456789
; +--------------------+
; |20:03:15 Wintertijd |
; |Donderdag 12.08.2004|
; +--------------------+
;-- 1. Zeile
;Uhrzeit
        movlw   0x80
        call    write_lcd_comm
        movf    dispStd,w
        call    write_byte
        movlw   ':'
        call    write_lcd_data
        movf    dispMin,w
        call    write_byte
        movlw   ':'
        call    write_lcd_data
        movf    dispSec,w
        call    write_byte
        movlw   ' '
        call    write_lcd_data

;Sommerzeit/Winterzeit
        movlw   lcd_mez_table-lcd_init_table
        btfsc   dispStat,4
        movlw   lcd_mesz_table-lcd_init_table
        call    lcd_string
        movlw   ' '
        call    write_lcd_data

;-- 2. Zeile
; Wochentag
        movlw   0xc0
        call    write_lcd_comm

        rlf     dispWtag,w    ; 1..7 *2
        movwf   temp
        rlf     temp,w        ; *4
        andlw   0xfc
        addwf   dispWtag,w    ; *5
        movwf   temp
        clrc
        rlf     temp,w        ; *10

        addlw   wtag_table-lcd_init_table
        call    lcd_string
        movlw   ' '
        call    write_lcd_data

        movf    dispTag,w
        call    write_byte
```

```
       movlw  '.'
       call   write_lcd_data
#ifdef alterText
       movf   dispMon,w
       call   write_byte
       movlw  '.'
       call   write_lcd_data
       movlw  0x20
       call   write_byte
#else
       movlw  ' '
       call   write_lcd_data
       movf   dispMon,w
       call   write_month
       movlw  ' '
       call   write_lcd_data
#endif
       movf   dispJahr,w
       call   write_byte
       movlw  ' '
       call   write_lcd_data

       return

#else
;  0123456789abcdef
; wenn alterText definiert ist
; +----------------+
; |20:03:15 MESZ ##|
; |Di 01.04.2003 ##|
; +----------------+
; sonst
; +----------------+
; |20:03:15 MESZ ##|
; |Di 01. APR 03 ##|
; +----------------+
;-- 1. Zeile
       movlw  0x80
       call   write_lcd_comm
       movf   dispStd,w
       call   write_byte
       movlw  ':'
       call   write_lcd_data
       movf   dispMin,w
       call   write_byte
       movlw  ':'
       call   write_lcd_data
       movf   dispSec,w
       call   write_byte
       movlw  ' '
       call   write_lcd_data

       movlw  lcd_mez_table-lcd_init_table
       btfsc  dispStat,4
       movlw  lcd_mesz_table-lcd_init_table
       call   lcd_string


;-- 2. Zeile
       movlw  0xc0
       call   write_lcd_comm
       movf   dispWtag,w
       call   write_wtag
       movlw  ' '
       call   write_lcd_data
       movf   dispTag,w
       call   write_byte
       movlw  '.'
       call   write_lcd_data
#ifdef alterText
       movf   dispMon,w
       call   write_byte
       movlw  '.'
       call   write_lcd_data
       movlw  0x20
       call   write_byte
```

```
#else
        movlw   ' '
        call    write_lcd_data
        movf    dispMon,w
        call    write_month
        movlw   ' '
        call    write_lcd_data
#endif
        movf    dispJahr,w
        call    write_byte
        movlw   ' '
        call    write_lcd_data

        return

#endif
;*********************************************************************

shiftDCF:
        rrf     dcfJahr,f
        rrf     dcfMon,f
        rrf     dcfTag,f
        rrf     dcfStd,f
        rrf     dcfMin,f
        rrf     dcfStat,f
        return

;*********************************************************************
;* in der 59. Sekunde aufgerufen, sortiert erst mal den Empfangspuffer,
;* um die Zeitwerte in einzelnen Bytes zu erhalten
;*********************************************************************
;55555555 54444444 44433333 33333222 22222221 11111111
;87654321 09876543 21098765 43210987 65432109 87654321   Sekunde
;PJJJJJJJ JMMMMMWW WTTTTTTP SSSSSSPm mmmmmm1a ooaR0000   DCF-Bits
;
;JJJJJJJJ 000MMMMM 00tttttt 00ssssss 0mmmmmmm 1aooaR00 00000www
; Jahr     Monat    Tag      Stunde   Minute   Status   Wtag
;*********************************************************************

sec59:  rlf     dcfMon,w
        rlf     dcfJahr,f       ; =Jahr
        rlf     dcfTag,w
        rlf     dcfMon,w
        andlw   0x07
        movwf   dcfWtag                 ; =Wochentag Mo..So = 1..7
        rrf     dcfMon,f
        rrf     dcfMon,f
        movlw   0x1f
        andwf   dcfMon,f        ; =Monat
        rrf     dcfTag,f
        movlw   0x3f
        andwf   dcfTag,f        ; =Tag
        rrf     dcfStd,w
        rrf     dcfMin,f
        rrf     dcfStat,f
        rrf     dcfMin,f
        rrf     dcfStat,f       ; =Status
        movlw   0x7f
        andwf   dcfMin,f        ; =Minute
        rrf     dcfStd,f
        rrf     dcfStd,f
        movlw   0x3f
        andwf   dcfStd,f        ; =Stunde

        movlw   dcf1Min
        movwf   FSR
        call    tickMin                 ; letzte Zeit weiterrechnen

        movf    dcfMin,w        ; und mit gerade empfangener
        xorwf   dcf1Min,w       ; Zeit vergleichen
        skpz
        goto    ungleich
        movf    dcfStd,w
        xorwf   dcf1Std,w
        skpz
        goto    ungleich
```

```
        movf    dcfTag,w
        xorwf   dcf1Tag,w
        skpz
        goto    ungleich
        movf    dcfMon,w
        xorwf   dcf1Mon,w
        skpz
        goto    ungleich
        movf    dcfJahr,w
        xorwf   dcf1Jahr,w
        skpz
        goto    ungleich

;*** zwei aufeinanderfolgende Zeiten haben zueinander gepasst
;*** die aktuelle Zeit bei nächster Sekunde ins Display kopieren

        bsf     syncd
        bsf     neusync
        return

ungleich:
;*** zwei aufeinanderfolgende Zeiten haben nicht eine Minute unterschied
;*** merken wir die gerade empfangene Zeit für die nächste Minute
        movf    dcfMin,w
        movwf   dcf1Min
        movf    dcfStd,w
        movwf   dcf1Std
        movf    dcfTag,w
        movwf   dcf1Tag
        movf    dcfMon,w
        movwf   dcf1Mon
        movf    dcfJahr,w
        movwf   dcf1Jahr
        movf    dcfStat,w
        movwf   dcf1Stat
        bcf     syncd
        return

;*********************************************************************
;* hier wird eine Zeit um eine Minute weitergezählt.
;*********************************************************************
tick:   movlw   dispSec
        movwf   FSR

        call    IncDec ; Sekunde
        movlw   0x60
        xorwf   INDF,w
        skpz
        return
        clrf    INDF    ; Sekunde=0

        incf    FSR,f   ; ->Minute
tickMin:
        call    IncDec ; Minute
        movlw   0x60
        xorwf   INDF,w
        skpz
        return
        clrf    INDF    ; Minute=0
        incf    FSR,f   ; -> Stunde
        call    IncDec ; Stunde

#define sommertest
#ifdef sommertest
        movf    FSR,w   ; Zeiger auf Stunde sichern
        movwf   tmp2

        movlw   4
        addwf   FSR,f   ; -> WTag
        movf    INDF,w
        xorlw   7       ; Sonntag?
        skpz
        goto    keinWechsel

; es ist ein Sonntag
        decf    FSR,f   ; ->Jahr
```

```
        decf    FSR,f  ; ->Monat
        decf    FSR,f  ; ->Tag
        movf    INDF,w
        addlw   -0x25
        btfss   STATUS,C
        goto    keinWechsel   ; Tag ist kleiner 25

; es ist mindestens der 25.
        incf    FSR,f  ; -> Monat
        movf    INDF,w
        xorlw   0x03   ; März?
        skpz
        goto    keinMaerz     ; kein März

; es ist ein Sonntag >=25. im März
        decf    FSR,f  ; -> Tag
        decf    FSR,f  ; -> Stunde
        movf    INDF,w
        xorlw   0x02
        skpz
        goto    keinWechsel   ; es ist nicht 2Uhr

;es ist 2 Uhr --> es wird Sommer
        incf    INDF,f ; Uhr eine Stunde vorstellen
        bsf     WinterSommer
        movlw   5
        addwf   FSR,f  ; -> Status
        movlw   0x30
        xorwf   INDF,f ; Status umstellen
        goto    fertig

keinMaerz:
        xorlw   0x03^0x10     ; Oktober?
        skpz
        goto    keinWechsel

; es ist ein Sonntag >=25. im Oktober
        decf    FSR,f  ; -> Tag
        decf    FSR,f  ; -> Stunde
        movf    INDF,w
        xorlw   0x03
        skpz
        goto    keinWechsel   ; es ist nicht 3 Uhr

; es ist ein Sonntag >=25. im Oktober, 3 Uhr
        movlw   5
        addwf   FSR,f  ; -> Status
        movf    INDF,w
        andlw   0x10   ; ist noch Sommerzeit?
        skpnz
        goto    keinWechsel

        movlw   0x30
        xorwf   INDF,f ; Status umstellen
        movf    tmp2,w
        movwf   FSR    ; -> Stunde
        decf    INDF,f ; Uhr eine Stunde zurück
        bsf     SommerWinter
        goto    fertig

keinWechsel:
        bcf             SommerWinter
        bcf             WinterSommer
fertig:
        movf    tmp2,w
        movwf   FSR           ; Zeiger wieder auf Stunde
#endif


        movlw   0x24   ; Stunde==24?
        xorwf   INDF,w
        skpz
        return
        clrf    INDF   ; Stunde=0
```

```
        incf    FSR,f
tickTag:
        call    IncDec ; Tag

        incf    FSR,f  ; ->Monat
        incf    FSR,f  ; ->Jahr
        incf    FSR,f  ; ->WTag
        incf    INDF,f
        movlw   1
        btfsc   INDF,3 ; WTag==8?
        movwf   INDF
        decf    FSR,f  ; ->Jahr
        decf    FSR,f  ; ->Monat

        movlw   8
        subwf   INDF,w
        movf    INDF,w
        skpnc
        addlw   -7
        andlw   0x01
        addlw   0x31
        movwf   temp
        movf    INDF,w
        xorlw   2       ; Februar ?
        skpz
        goto    nofeb
        movlw   0x30
        movwf   temp
        incf    FSR,f  ; ->Jahr
        movf    INDF,w ; Jahr
        decf    FSR,f  ; ->Monat
        decf    FSR,f  ; -> Tag

; 00 04 08 12 16 sind Schaltjahre, danach wiederholt es sich
        andlw   0x13
        skpnz
        goto    leapyear

        xorlw   0x12
        skpnz
        goto    leapyear
        movlw   0x29
        movwf   temp
leapyear:
nofeb:
        movf    temp,w
        xorwf   INDF,w
        skpz
        return
        movlw   1
        movwf   INDF   ; Tag=1

        incf    FSR,f  ; -> Monat
        call    IncDec
        movf    INDF,w
        xorlw   0x13
        skpz
        return
        movlw   1
        movwf   INDF   ; Monat=1

        incf    FSR,f  ; -> Jahr
;       call    IncDec
;       return

IncDec:         incf    INDF,f          ; Zähler erhöhen
        movlw   6
        addwf           INDF,w          ; +6 (Dez.-korrektur) --> Ergebnis vorerst in w
        btfsc   STATUS,DC       ; Überlauf im Low-Nibble?
        movwf   INDF            ; ja, Korr. Ergebn. zurück
        return

;*****************************************************************
;************* LCD-Routinen ***************************************
;*****************************************************************
```

```
;* Display is in 4 bit mode. Datalines connected to PORTB4..7
;* PORTB3 = Register Select   PORTB2 = Read not Write   PORTB1 = Enable
;*****************************************************************
LCEN equ 1
LCRW equ 2
LCRS equ 3

tst_lcd_busy
       movlw  (1<<LCRW)
       movwf  PORTB
waitbusy
       bsf    PORTB,LCEN          ; get hi-Nibble
       nop
       rlf    PORTB,w             ; Busy-flag to Carry
       bcf    PORTB,LCEN
       nop
       bsf    PORTB,LCEN          ; get lo-Nibble
       nop
       bcf    PORTB,LCEN
       bc     waitbusy
       return

;*****************************************************************
write_lcd_comm
       bsf    comm
write_lcd_data
       movwf  lc_data
       call   tst_lcd_busy
       bsf    STATUS,RP0    ; select Register-Page 1
       movlw  0x01
       andwf  TRISB,f
       bcf    STATUS,RP0    ; select Register-Page 0
       movf   lc_data,w
       call   lcd_nibble
       swapf  lc_data,w
       call   lcd_nibble

       bcf    comm
       bsf    STATUS,RP0    ; select Register-Page 1
       movlw  0xf0
       iorwf  TRISB,f
       bcf    STATUS,RP0    ; select Register-Page 0
       return

lcd_nibble
       andlw  0xf0
       btfss  comm
       iorlw  (1<<LCRS)
       movwf  PORTB
       nop
       bsf    PORTB,LCEN
       nop
       bcf    PORTB,LCEN
       return
;*****************************************************************
lcd_string
       movwf  lcd_init_cnt
       goto   lcd_init_loop
lcd_init
       clrf   lcd_init_cnt
lcd_init_loop
       movf   lcd_init_cnt,w
       call   lcd_string_table
       xorlw  0xff          ; mit 0xff vergleichen
       skpnz
       return
       xorlw  0xff          ; xor rückgängig gemacht
       movwf  lc_data
       bsf    comm
       btfss  lc_data,7
       bcf    comm
       andlw  0x7f
       call   write_lcd_data
       incf   lcd_init_cnt,f
       goto   lcd_init_loop
```

```
;*********************************************************************
write_byte              ; writes a hex-byte
       movwf  temp_byte
       swapf  temp_byte,w
       call   write_nibble
       movf   temp_byte,w

write_nibble
       andlw  0x0f
       addlw  0x06
       btfsc  STATUS,DC
       addlw  0x07
       addlw  0x30-0x06
       goto   write_lcd_data
;*********************************************************************
write_wtag
       andlw  0x07
       movwf  temp_byte
       addwf  temp_byte,f
       movf   temp_byte,w
       call   wtag_table
       call   write_lcd_data
       incf   temp_byte,w
       call   wtag_table
       goto   write_lcd_data
;*********************************************************************
write_month
       movwf  temp_byte    ; save BCD month
       movlw  -6
       btfsc  temp_byte,4  ; less than 0x10 ? then skip
       addwf  temp_byte,f  ; else subtract 6. A very simple BCD-bin conversion
                                   ; but only for BCD less 0x19 ;(
       bcf    STATUS,C
       rlf    temp_byte,w  ; *2
       addwf  temp_byte,f  ; *3
       movf   temp_byte,w
       call   month_table  ; print three characters
       call   write_lcd_data
       incf   temp_byte,f
       movf   temp_byte,w
       call   month_table
       call   write_lcd_data
       incf   temp_byte,w
       call   month_table
       goto   write_lcd_data
;*********************************************************************
       end
```