```
;File:          DISEQC05.ASM           Version 05/06
;author:        W. Waetzig                                                  03.0
6.2006
;processor:     PIC 16F628
;assembler:     MPASM.EXE by MicrochiP
;modules used:
;       with driver for LCD-display with 2*16 bytes
;       with interface to SAT-bus
;       with short mnemonics output for DiSEQC-commands
;       with driver for push-button-control.
;-------------------------------------------------------------------------
; DISEQC - DiSEqC (TM) monitor.
; The system consists of a microprocessor PIC 16F628,
; a LCD-display with 2*16 bytes and
; 4 push-buttons for function-control
; connected to the SAT-bus.
;
; DiSEqC = Digital Satellite Equipment Control (eutelsat)
; consists of a sequence of bus-commands in bytes.
;
; Method of Data-Bit Signalling:
;       "0" data-bit:           1.0 ms 22-kHz-signal / 0.5 ms pause
;       "1" data-bit:           0.5 ms 22-kHz-signal / 1.0 ms pause
;       A data-bit has the time-length of 1.5 ms;
;       a byte has the time-length of 9 * 1.5 ms = 13.5 ms.
;       Byte:           8 data-bits / 1 parity bit
;
;       Master Command: FRAMING |P| ADDRESS |P| COMMAND |P| DATA |P|
;               byte 1: Framing Byte
;               byte 2: Address Byte
;               byte 3: Command Byte
;               byte 4: Data Byte
;               byte 5: Data Byte ...
;       Slave Reply:            FRAMING |P| DATA |P| DATA |P|
;               byte 1: Framing Byte
;               byte 2: Data Byte
;               byte 3: Data Byte
;
; Up to 80 bytes (max. 20 commands) can be stored in RAM and reviewed.
;
; Pushbutton-functions
; PB1           start / stop data-taking
; PB2           display next command from RAM
; PB3           display previous command from RAM
; PB4           toggle data display-mode
;
;-------------------------------------------------------------------------
;       PORTA,0 comparator-1-input <= 14/18 Volt
;       PORTA,1 comparator-2-input <= 22 kHz signal
;       PORTA,2 reference voltage output + reference-input comparator-1+2
;       PORTA,3 output comparator-1 => LED-indicator for 14/18 Volt
;       PORTA,4 output comparator-2 => LED-indicator for 22 kHz signal
;       PORTB,0 INT_22K (input) interrupt for 22 kHz from PORTA,4
;       PORTB,1 BUT_SEL (input) select bushbutton
;       PORTB,2 LCD_EN          strobe LCD-register
;       PORTB,3 LCD_RS          LCD register select
;       PORTB,4 LCD_DATA 4      in 4-bit-mode / pushbutton 1
;       PORTB,5 LCD_DATA 5      in 4-bit-mode / pushbutton 2
;       PORTB,6 LCD_DATA 6      in 4-bit-mode / pushbutton 3
;       PORTB,7 LCD_DATA 7      in 4-bit-mode / pushbutton 4
;
;-------------------------------------------------------------------------
        processor       16f628          ;Processor type PIC16F628
;
        include "p16f628.inc"
;
        org             2007            ;Fuse settings:
        dw              b'11110100100001'       ;CP1/CP0=off, PWRT=enable;
                                        ;WDT=disable; BODEN=disable.
                                        ;bit4,1,0:   011 =RC-Oscillator
                                        ;            010 =HS-Oscillator *
                                        ;            001 =XT-Oscillator
;-------------------------------------------------------------------------
```

```
;
;       PIC 16F628:     2048*14 FLASH, 224*8 RAM, 128*8 EEPROM
; Variables from        20 (80 bytes) in BANK0
; Variables from        A0 (80 bytes) in BANK1
; Variables from        70 (16 bytes) common for all banks
;
;------interrupt saved registers
intwrsav        equ     0x70            ; save W
intstsav        equ     0x71            ; save STATUS
intpcsav        equ     0x72            ; save PCLATH
VARADRES        equ     0x20
;------delay-counter
timecnt1        equ     VARADRES+00     ;counter for 100 microsec
timecnt2        equ     VARADRES+01     ;counter for 4 ms
timecnt3        equ     VARADRES+02     ;counter for 0,5 s
;------LCD / text-routines
lcdcount        equ     VARADRES+03     ;byte counter for display
lcdvalue        equ     VARADRES+04     ;byte for display
tmpdigit        equ     VARADRES+05
lcdmodus        equ     VARADRES+06
lcdindex        equ     VARADRES+07
txtgroup        equ     VARADRES+08     ; group of command-text
txtindex        equ     VARADRES+09     ; index of command-text
txtcount        equ     VARADRES+0A     ; counter for command-text
LCD_RS          equ     0x02            ;PORTB,2        LCD register select
LCD_EN          equ     0x03            ;PORTB,3        LCD strobe register
;
;------global values
cntimer0        equ     VARADRES+10     ;count TIMER1 interrupts
cntintb0        equ     VARADRES+11     ;count RB0 interrupts
flagtest        equ     VARADRES+12     ; flags
databyte        equ     VARADRES+13     ; byte
databcnt        equ     VARADRES+14     ; bit counter
datapary        equ     VARADRES+15     ; byte parity
dataread        equ     VARADRES+16     ; byte read
datacntr        equ     VARADRES+17     ; byte counter
ctcomand        equ     VARADRES+18     ; number of command
ixcomand        equ     VARADRES+19     ; index of command
comand00        equ     VARADRES+1A     ; command 0
comand01        equ     VARADRES+1B     ; command 1
comand02        equ     VARADRES+1C     ; command 2
comand03        equ     VARADRES+1D     ; command 3
comand04        equ     VARADRES+1E     ; command 4
setfunct        equ     VARADRES+20     ; function-mode
comdchar        equ     VARADRES+21     ; command-byte
lastaddr        equ     VARADRES+22     ; last address in RAM
portflag        equ     VARADRES+23     ; pusbutton flags
xramflag        equ     VARADRES+24     ; RAM-flags
funcsave        equ     VARADRES+25     ; function-byte
dziffer1        equ     VARADRES+26     ; for bin=>dec
dziffer2        equ     VARADRES+27
dziffer3        equ     VARADRES+28
ctcomnds        equ     VARADRES+29     ; # of commands
;
RAM_ANF equ     0xA0            ; RAM-buffer
RAM_END equ     0xEF
;
;-----------------------------------------------------------
;
                org     0x000           ;RESET vector
                goto    mainprog
                dt      "DME"
;
;======================= INTERRUPT ===================================
                org     0x004           ;INTERRUPT vector
;interrupt-routine
intentry        movwf   intwrsav        ;save W
                swapf   STATUS,W
                clrf    STATUS          ;set to bank0
                movwf   intstsav        ;save STATUS
;               movf    PCLATH,W
;               clrf    PCLATH
;               movwf   intpcsav        ;save PCLATH
```

```
;interrupt handling,    GIE=0 by hardware
; check interrupt flags in INTCON ;GIE/PEIE/T0IE/INTE/RBIE/T0IF/INTF/RBIF
                btfss   INTCON,INTF     ; external interrupt (22 kHz signal)
                goto    int_tim0
; external interrupt from 22 kHz-signal (each 45 micro-sec)
                bcf     INTCON,INTF     ;clear INTF
                movlw   0xFD            ; TIMER0 = -3
                movwf   TMR0            ; set TIMER0 and reset prescaler
                clrf    cntimer0        ;clear timer counter
                incf    cntintb0,F      ; count 22kHz interrupts
                movlw   d'033'
                subwf   cntintb0,W
                btfsc   STATUS,C        ; >33 interrupts?
                decf    cntintb0,F      ; indicate permanent 22kHz signal
                goto    int_rest
; TIMER0 interrupt (at 32*3 + 11 micro-sec) / prescaler=32/TIMER0=-3
int_tim0        btfss   INTCON,T0IF     ;T0IF-overflow flag?
                goto    int_rest
                bcf     INTCON,T0IF     ;clear timer0 int-flag
                movlw   0xFD            ; TIMER0 = -3
                movwf   TMR0            ; set TIMER0 and reset prescaler
                incf    cntimer0,F      ;count timer interrupts
                movlw   d'016'
                subwf   cntimer0,W
                btfss   STATUS,C        ; > 16 wait periodes
                goto    inttnewc
                decf    cntimer0,F      ; permanent wait cycles
                goto    intclchr        ; clear byte-mode
; control-bits
;               flagtest,7              byte is complete
;               flagtest,1              indicator for parity-bit
;               flagtest,0              parity-bit
; check, if a new byte starts
inttnewc        movlw   d'032'
                subwf   cntintb0,W
                btfsc   STATUS,C        ; permanent 22kHz signal before?
                goto    intclchr        ; clear byte-mode
; check for data-bit "0" => 22 pulses + 4 wait-periodes
intchk00        movlw   d'004'          ;=> check wait period
                subwf   cntimer0,W
                btfss   STATUS,Z
                goto    intchk10
                movlw   d'022'
                subwf   cntintb0,W
                btfsc   STATUS,Z        ; = 22 pulses
                goto    intrbit0
                addlw   d'001'
                btfss   STATUS,Z        ; = 21 pulses
                goto    int_rest
intrbit0        btfss   flagtest,1      ; is parity check?
                bcf     databyte,0
                bcf     flagtest,0      ; => parity-bit
                goto    intrbits
; check for data-bit "1"=> 11 pulses + 8 wait-periodes
intchk10        movlw   d'008'          ;=> check wait period
                subwf   cntimer0,W
                btfss   STATUS,Z
                goto    int_rest
                movlw   d'011'
                subwf   cntintb0,W
                btfsc   STATUS,Z        ; = 11 pulses
                goto    intrbit1
                addlw   d'001'
                btfss   STATUS,Z        ; = 10 pulses
                goto    int_rest
intrbit1        btfss   flagtest,1      ; is parity check?
                bsf     databyte,0
                bsf     flagtest,0      ; => parity-bit
; shift the bits into databyte
intrbits        btfsc   flagtest,1      ; parity-check?
                goto    intpchek
                btfsc   databyte,0
                incf    datapary,F      ; parity for data
```

```
                rlf     databyte,F
                decfsz  databcnt,F      ; count bits in databyte
                goto    intclrb0
                rrf     databyte,F      ;compensate
                bsf     flagtest,1      ;set parity-test
                goto    intclrb0
intpchek        nop
                bsf     flagtest,7      ;indicate, that a byte has been read
                movf    databyte,W
                movwf   dataread
intclchr        bcf     flagtest,1      ; clear parity-indicator
                clrf    datapary        ; start new byte
                movlw   d'008'
                movwf   databcnt
intclrb0        clrf    cntintb0        ;reset pulse-counter
;restore registers and return
int_rest
;               movf    intpcsav
;               movwf   PCLATH          ; resore PCLATH
                swapf   intstsav,W
                movwf   STATUS          ;restore STATUS,
                swapf   intwrsav,F      ;swapf does not change STATUS-bits !!!
                swapf   intwrsav,W      ;restore W
                retfie                  ;return from interrupt, sets GIE=1
;
;-----------------------------------------------------------------
;
;------------fixed text in page 0 for number-print wdigtLCD
fixdchar        clrf    PCLATH          ;page 0
                addwf   PCL,F           ;mod. PC
                dt      "0123456789ABCDEF"
;
txtarray        clrf    PCLATH          ;page 0
                addwf   PCL,F           ;text collection for wtextLCD
                dt      " DiSEqC Monitor",0     ;#0
                dt      "DiSEqC raw data",0     ;#2
                dt      " commands read ",0     ;#4
                dt      "+ error",0     ;#6
                dt      "- error",0     ;#7
                dt      "       ",0     ;#8
                dt      " show hex-data ",0     ;#9
                dt      " show commands ",0     ;#11
                dt      "Elektor V.05/06",0     ;#13
;
                dt      "DiSEqC(TM) Monitor 06-2006"
;
;============================ INIT PERIPHERALS ========================
;
; init the voltage reference module
initvref        bcf     STATUS,RP0      ;set to bank0
                bcf     STATUS,RP1
                clrf    PORTA
                movlw   b'00000110'     ;comparator mode
                movwf   CMCON
                bsf     STATUS,RP0      ;set to bank1
                movlw   b'00000111'     ;RA0:RA2=IN; RA3:RA4=OUT
                movwf   TRISA
                movlw   b'11101100'     ;VREN/VROE/VRR/0/1100
                movwf   VRCON           ;=> Vdd*12/24 reference-voltage
                bcf     STATUS,RP0      ;set to bank0
                return
;
; init the TIMER0
inittmr0        bcf     STATUS,RP0      ;set to bank0
                movlw   0xFF
                movwf   PORTB
                bsf     STATUS,RP0      ;set to bank1
                movlw   b'00000011'     ;RB0:RB1=IN; RB2:RB7=OUT
                movwf   TRISB
                movlw   b'01000100'     ;RPBU/INTEDG/T0CS/T0SE/PSA/100
                movwf   OPTION_REG      ;rising-edge,prescaler=32
                bcf     STATUS,RP0      ;set to bank0
                movlw   0xFD            ; TIMER0 = -3
```

```
                movwf   TMR0
                return
;
; init data-values
initdata        clrf    databyte
                clrf    datapary
                movlw   0x08
                movwf   databcnt
                clrf    cntimer0
                clrf    cntintb0
                clrf    portflag
                clrf    xramflag
                return
;
; init interrupt
initintr        clrf    flagtest
                clrf    datacntr        ; data counter for LCD
                clrf    INTCON          ;GIE/PEIE/T0IE/INTE/RBIE/T0IF/INTF/RBIF
                bsf     INTCON,T0IE
                bsf     INTCON,INTE
                bsf     INTCON,GIE      ;enable interrupts
                return
;
; init RAM-address
initramd        movlw   RAM_ANF
                movwf   FSR
                movwf   lastaddr        ; set start of RAM address
                clrf    ctcomand        ; number of command
                clrf    xramflag
                return
;
;====================== WAIT-ROUTINES ==========================
;
;------------Time loop for 100 microsec for 4 MHz quartz
wait100u        movlw   d'25'
                movwf   timecnt1        ;count totally 100 cycles
tim00100        nop                     ; 1 CYC
                decfsz  timecnt1,F      ; 1 CYC decrement/skip
                goto    tim00100        ; 2 CYC
                return
;
;------------Time loop for 1 millisec (10 * wait100u)
wait01ms        movlw   d'10'
                movwf   timecnt2        ;count 10 cycles
timxms00        call    wait100u        ;=100 microsec delay
                decfsz  timecnt2,F      ;decrement/skip
                goto    timxms00
                return
;
wait02ms        call    wait01ms
                goto    wait01ms
;
;------------Time loop for 0,5 sec (250 * tim2msec)
wait005s        movlw   d'250'
                movwf   timecnt3        ;count 250 cycles
tim05s00        call    wait01ms
                call    wait01ms
                decfsz  timecnt3,F      ;decrement/skip
                goto    tim05s00
                return
;
;================================= LCD-DRIVER ======================
;
;------------Initalize LC-display in 4-bit data-mode
initLCD movlw   b'00100000'     ;init bit-pattern for LCD - 4 bit-mode
                call    wcommLCD        ;write to LCD
                call    wait02ms
                movlw   b'00100000'     ;send command again
                call    wcommLCD        ;write to LCD
                call    wait02ms
                movlw   b'00100000'     ;send command again
                call    wcommLCD        ;write to LCD
                call    wait02ms
```

```
                movlw   b'00101000'     ;db4=0: 4bit data
                                        ;db3=1: 2 lines
                call    wcommLCD        ;write to LCD
                call    wait02ms
                movlw   b'00001110'     ;db2=1: display on
                                        ;db1=1: cursor on
                                        ;db0=0: cursor no blink
                call    wcommLCD        ;write to LCD
                call    wait02ms
                movlw   b'00000001'     ;clear display
                call    wcommLCD        ;write to LCD
                call    wait005s
                return
;
;------------Move cursor to 1st line, 1st char-pos.
line1LCD        movlw   b'10000000'     ;display 1st line
                goto    wcommLCD
;
;------------Move cursor to 2nd line
line2LCD        movlw   b'11000000'     ;display 2nd line
;
;------------output W to LCD in 2 * 4 bit nibbles
wcommLCD        clrf    lcdmodus        ;switch LCD to command mode
                goto    wnextLCD
writeLCD        bsf     lcdmodus,LCD_RS ;switch LCD to data mode
wnextLCD        bsf     lcdmodus,LCD_EN ;set strobe-bit
                movwf   lcdvalue        ; save data
                andlw   0xF0            ; get data 4..7
                iorwf   lcdmodus,W
                movwf   PORTB           ;data => PORTB
                nop
                bcf     PORTB,LCD_EN    ;strobe-1 data for LCD
                swapf   lcdvalue,W
                andlw   0xF0            ; get data 0..3
                iorwf   lcdmodus,W
                movwf   PORTB           ;data => PORTB
                nop
                bcf     PORTB,LCD_EN    ;strobe-2 data for LCD
                goto    wait100u        ;delay 100 microsec => return
;
;======================== WRITE CHARACTER ROUTINES ====================
;
;------------write text-string from text-array onto LCD; # of text in W
wtextLCD        andlw   0x1F
                movwf   lcdindex
                bcf     STATUS,C        ;C=0
                rlf     lcdindex,F
                rlf     lcdindex,F
                rlf     lcdindex,F      ;index*8
; loop to output the text
wtextL01        movf    lcdindex,W      ;load lcdindex
                call    txtarray        ;get text-byte
                andlw   0xFF            ;mask for NUL-byte
                btfsc   STATUS,Z        ; end-of-text ?
                goto    wtextL02
                call    writeLCD        ;write to LCD
                incf    lcdindex,F      ;count byte
                goto    wtextL01
wtextL02        return
;
;------------output single digit to LCD
wdigtLCD        andlw   0x0F
                call    fixdchar        ;get text-byte
                call    writeLCD        ;write to LCD
                return
;
;------------write dual hex-digit onto LCD, value (upper/lower) in W
wrdigLCD        movwf   tmpdigit
                swapf   tmpdigit,W      ;upper nibble in hexa
                call    wdigtLCD
                movf    tmpdigit,W      ;lower nibble in hexa
                call    wdigtLCD
                return
```

```
;
;------------write value in W as 2-digits decimal
wrdecLCD        call    bin2dec ;convert
                movf    dziffer2,W      ;suppress leading zero
                btfss   STATUS,Z
                goto    wrdec001
                movlw   ' '
                call    writeLCD
                goto    wrdec002
wrdec001        call    wdigtLCD
wrdec002        movf    dziffer3,W
                call    wdigtLCD
                return
;
;-------------------------------------------------------------
;
; BIN to DEC conversion, BIN-byte in W => dziffer1/dziffer2/dziffer3
bin2dec         clrf    dziffer1
                clrf    dziffer2
bindec1         addlw   0x9C            ;d'-100'
                btfss   STATUS,C        ;value-100 < 0?
                goto    bindec2
                incf    dziffer1,F
                goto    bindec1
bindec2         addlw   0x64            ;+100
bindec3         addlw   0xF6            ;d'-10'
                btfss   STATUS,C        ;value-10 < 0?
                goto    bindec4
                incf    dziffer2,F
                goto    bindec3
bindec4         addlw   0x0A            ;+10
                movwf   dziffer3
                return
;
;========================== RAM-READ/WRITE-ROUTINES ============================
;
; store byte in W into RAM (80 bytes from 0A0 to 0EF)
; set xramflag,1 if buffer is full
storcram        bcf     xramflag,1
                movwf   INDF            ; indirect store
                movlw   RAM_END ; 0EF = end-address of RAM
                subwf   FSR,W
                btfsc   STATUS,Z        ; check for end address
                goto    storcerr
                movf    INDF,W
                andlw   0xF8
                sublw   0xE0
                btfsc   STATUS,Z        ; check for 0Ex
                incf    ctcomand,F      ;count # of DiSEqC-commands
                incf    FSR,F
                incf    lastaddr,F      ; pointer to next RAM-address
                goto    storcext
storcerr        bsf     xramflag,1
storcext        return
;
;-------------------------------------------------------------
;
; get previous byte from RAM (error: xramflag,0 = 1)
getpbyte        bcf     xramflag,0
                movf    FSR,W
                subwf   lastaddr,W
                btfsc   STATUS,C        ; check for end-address
                goto    getpbyt0
                movf    lastaddr,W      ;correct pointer
                movwf   FSR
getpbyt0        movlw   RAM_ANF ; 0A0 = start address of RAM
                subwf   FSR,W
                btfss   STATUS,C        ; check for start address
                goto    getpcerr
                movf    INDF,W          ;load byte
                decf    FSR,F           ;point to previous byte
                goto    getpchex
getpcerr        bsf     xramflag,0      ; => error-indicator
```

```
getpchex        return
;
; get next byte from RAM (error: xramflag,0 = 1)
getnbyte        bcf     xramflag,0
                movlw   RAM_ANF
                subwf   FSR,W           ; 0A0 = start address of RAM
                btfsc   STATUS,C        ; check for start address
                goto    getnbyt0
                movlw   RAM_ANF
                movwf   FSR
getnbyt0        movf    FSR,W
                subwf   lastaddr,W
                btfss   STATUS,C        ; check for end-address
                goto    getncerr
                movf    INDF,W          ;load byte
                incf    FSR,F           ;point to next byte
                goto    getnchex
getncerr        bsf     xramflag,0      ; => error-indicator
getnchex        return
;
;====================== DISPLAY COMMANDS ================================
;
; ====== display the previous command in RAM
dsppcram        call    getpbyte        ; ==> skip last byte
                btfsc   xramflag,0      ; test if error
                goto    dispperr
dsppcra1        call    getpbyte        ; ==> skip back to last Framing Byte
                btfsc   xramflag,0      ; test if error
                goto    dispperr
                andlw   0xF8
                sublw   0xE0
                btfss   STATUS,Z        ;check for 0Ex (Framing Byte)
                goto    dsppcra1
dsppcra2        call    getpbyte        ; ==> skip back to last Framing Byte
                btfsc   xramflag,0      ; test if error
                goto    dispperr
                andlw   0xF8
                sublw   0xE0
                btfss   STATUS,Z        ;check for 0Ex (Framing Byte)
                goto    dsppcra2
                incf    FSR,F           ;compensate index-pointer
                decf    ctcomand,F      ; # of DiSEqC-command
                decf    ctcomand,F
; ====== display the next command in RAM
dspncram        call    getnbyte        ; ==> skip forward to next Framing Byte
                btfsc   xramflag,0      ; test if error
                goto    dispcerr
                movwf   comdchar        ; => save byte
                andlw   0xF8
                sublw   0xE0
                btfss   STATUS,Z        ;check for 0xEy (Framing Byte)
                goto    dspncram
                incf    ctcomand,F      ;# of DiSEqC-command
; ======== common code for dsppcram and dspncram
dispcom1        call    line2LCD        ; write on LCD - line 2
                movlw   "#"
                call    writeLCD
                movf    ctcomand,W      ;# of DiSEqC-command
                call    wrdecLCD
                movlw   " "
                call    writeLCD
; store the first 5 bytes separately
                clrf    ixcomand        ; command index
dispcom2        call    storbyte        ;store byte in comand00 .. comand04
                movf    comdchar,W
                call    wrdigLCD        ; display byte in hexa
; get the next byte, end of command by next Framing-Byte or end of RAM-buffer
                call    getnbyte        ; get next byte => comdchar
                movwf   comdchar        ; => save byte
                btfsc   xramflag,0      ; test if error (end of RAM)
                goto    dispcomx
                andlw   0xF8
                sublw   0xE0
```

```
                btfss   STATUS,Z        ;check for 0xEy = (next Framing Byte)
                goto    dispcom2
; ignore commands with length < 3 bytes
dispcomx        movlw   003
                subwf   ixcomand,W
                btfss   STATUS,C
                goto    dispcerr        ; => +error
                decf    FSR,F           ;compensate index-pointer
                movlw   d'08'           ; "      "
                call    wtextLCD
;======== display command with hexa-numbers
                btfss   setfunct,0
                goto    dispcomy        ; display bytes in hexa only
                call    line1LCD
                btfsc   comand00,2      ; ==>byte 1: Framing Byte: MASTER/SLAVE
                goto    dispslav
; ------ display mnemonic command for MASTER on LCD line 1
                movlw   "M"
                call    writeLCD
                movlw   " "
                call    writeLCD
                swapf   comand01,W      ; ==>byte 2: Address Byte
                andlw   0x0F
                addlw   0x70
                call    wrcomand        ; => command DEVICES
                movlw   " "
                call    writeLCD
                movf    comand02,W      ; ==>byte 3: Command Byte
                andlw   0x7F
                call    wrcomand        ; => command FUNCTIONS
                movlw   ":"
                call    writeLCD
                movlw   0x04
                subwf   ixcomand,W
                btfss   STATUS,C
                goto    dispcomy
                movf    comand03,W      ; ==> byte 4: DATA
                call    wrdigLCD        ; display byte in hexa
                movlw   0x05
                subwf   ixcomand,W
                btfss   STATUS,C
                goto    dispcomy
                movf    comand04,W      ; ==> byte 5: DATA
                call    wrdigLCD        ; display byte in hexa
                goto    dispcomy
; ------ display command for SLAVE on line 1
dispslav        movlw   "S"
                call    writeLCD
                movlw   " "
                call    writeLCD
                movf    comand01,W
                call    wrdigLCD        ; DATA 1 in hexa
                movlw   " "
                call    writeLCD
                movf    comand02,W
                call    wrdigLCD        ; DATA 2 in hexa
                movlw   d'08'           ; "      "
                call    wtextLCD
                goto    dispcomy
;======= ERROR +/-
dispperr        call    line2LCD        ; write on LCD - line 2
                clrf    ctcomand        ;# of DiSEqC-command
                movlw   d'07'           ; "-error "
                goto    dispwerr
dispcerr        call    line2LCD        ; write on LCD - line 2
                movf    ctcomnds,W
                movwf   ctcomand        ;# of DiSEqC-command
                movlw   d'06'           ; "+error "
dispwerr        call    wtextLCD
                btfss   setfunct,0      ; command-text output ?
                goto    dispcomy
                movlw   d'08'           ; "      "
                call    wtextLCD
```

```
                call    line1LCD        ;LCD line 1
                movlw   d'08'           ; "      "
                call    wtextLCD
                movlw   d'08'           ; "      "
                call    wtextLCD
dispcomy        movlw   d'08'           ; "      "
                call    wtextLCD
                return
;
;============================ PUSHBUTTON-FUNCTIONS ====================
;
;function PB1:  start / stop data-taking
butnfct1        movlw   0x80
                xorwf   portflag,F      ; toggle bit7
                btfss   portflag,7
                goto    butnfc01
; start data-taking
                call    dispcont        ;write text " DiSEqC raw data"
                movlw   "*"
                call    wrfctcod        ; write function-code on LCD
                call    initramd        ; set FSR, lastaddr, ctcomand
                call    initintr        ; => switch on interrupt
                goto    butnfc03
; stop data-taking
butnfc01        bcf     INTCON,GIE      ; => interrupt off
                clrf    portflag        ; reset all flags
                clrf    xramflag
                call    line1LCD
                movlw   d'00'           ; "DiSEqC Monitor"
                call    wtextLCD
                movlw   " "
                call    writeLCD
                call    line2LCD
                movf    ctcomand,W      ;# of DiSEqC-command
                movwf   ctcomnds        ; # of DiSEqC-commands in decimal
                call    wrdecLCD
                movlw   d'04'           ;" commands read"
                call    wtextLCD
; correct lastaddr
                movf    lastaddr,W
                sublw   RAM_END
                btfsc   STATUS,Z
                goto    butnfc02
                movf    lastaddr,W
                sublw   RAM_ANF
                btfss   STATUS,Z
                decf    lastaddr,F
butnfc02        movlw   RAM_ANF ; set start of RAM-address
                movwf   FSR
                clrf    ctcomand        ; number of command
                clrf    flagtest
butnfc03        return
;
; function PB2  display next command from RAM
butnfct2        btfsc   portflag,7
                call    butnfc01        ; stop data-taking
                movlw   "+"
                call    wrfctcod        ; write function-code on LCD
                call    dspncram
                return
;
; function PB3  display previous command from RAM
butnfct3        btfsc   portflag,7
                call    butnfc01        ; stop data-taking
                movlw   "-"
                call    wrfctcod        ; write function-code on LCD
                call    dsppcram
                return
;
; function PB4  toggle data mode
butnfct4        btfsc   portflag,7
                call    butnfc01        ; stop data-taking
                movlw   ">"
```

```
                call    wrfctcod        ; write function-code on LCD
                movlw   0x01
                xorwf   setfunct,F      ;toggle function-mode
                btfss   setfunct,0
                movlw   d'09'           ;"show hex-data"
                btfsc   setfunct,0
                movlw   d'11'           ;"show commands"
                call    wtextLCD
                movlw   " "
                call    writeLCD
                btfsc   setfunct,0
                goto    butnfc44
                call    line1LCD
                movlw   d'00'           ; "DiSEqC Monitor"
                call    wtextLCD
                movlw   " "
                call    writeLCD
butnfc44        call    wait005s
                return
;
;=================================================================
;
;------display  |DiSEqC raw data|
;                 |               |
dispcont        call    line1LCD        ;LCD line 1
                movlw   " "
                call    writeLCD
                movlw   d'02'           ; " DiseqC raw data"
                call    wtextLCD
                call    line2LCD        ;LCD line 2
                movlw   d'08'           ; "      "
                call    wtextLCD
                movlw   d'08'           ; "      "
                call    wtextLCD
                movlw   d'08'           ; "      "
                call    wtextLCD
                call    line2LCD
                call    wait005s
                return
;
; write function-code in W onto LCD
wrfctcod        movwf   funcsave
                call    line1LCD
                movf    funcsave,W
                call    writeLCD
                call    line2LCD
                return
;
;======================== MAIN PROGRAM LOOP ==============================
;
;------------main program-loop
mainprog        call    initvref        ; initialize reference voltage
                call    inittmr0        ; initialize TIMER0
                call    initdata        ; set data bytes
                call    initLCD ; initialize the LCD
                call    wait005s
                call    line1LCD
                movlw   d'00'           ; "DiSEqC Monitor"
                call    wtextLCD
                movlw   " "
                call    writeLCD
                call    line2LCD
                movlw   d'13'           ; "Elektor V.05/06"
                call    wtextLCD
                movlw   " "
                call    writeLCD
                call    wait005s
                call    wait005s
                call    wait005s
                call    wait005s
                call    wait005s
                call    wait005s
                call    wait005s
                call    initramd        ; RAM
```

```
                clrf    portflag        ;push-button-control
                clrf    setfunct        ;function-mode
                call    butnfct1        ; => switch on interrupt + start data-taking
;
;---------------------------------------------------------------
;loop for control-byte-input
;               portflag,7              start / stop data-taking
;               portflag,4              push-button 4
;               portflag,3              push-button 3
;               portflag,2              push-button 2
;               portflag,1              push-button 1
;               xramflag,1              buffer is full
;               xramflag,0              error on RAM-address
;
; check for pushbutton 1
waitkeys        movlw   b'11101111'
                movwf   PORTB           ; set PORTB,4 for button-test
                call    wait01ms
                btfss   PORTB,1 ;test if button #1 is pressed
                goto    pbuttn11
                btfss   portflag,1
                goto    pbuttn02
                call    butnfct1        ; ===> execute function for button 1
                bcf     portflag,1      ; reset button-request
                goto    pbuttn05
pbuttn11        bsf     portflag,1      ; button is pressed
; check for pushbutton 2
pbuttn02        movlw   b'11011111'
                movwf   PORTB           ; set PORTB,5 for button-test
                call    wait01ms
                btfss   PORTB,1 ;test if button #2 is pressed
                goto    pbuttn22
                btfss   portflag,2
                goto    pbuttn03
                call    butnfct2        ; ===> execute function for button 2
                bcf     portflag,2      ; reset button-request
                goto    pbuttn05
pbuttn22        bsf     portflag,2      ; button is pressed
; check for pushbutton 3
pbuttn03        movlw   b'10111111'
                movwf   PORTB           ; set PORTB,6 for button-test
                call    wait01ms
                btfss   PORTB,1 ;test if button is #3 pressed
                goto    pbuttn33
                btfss   portflag,3
                goto    pbuttn04
                call    butnfct3        ; ===> execute function for button 3
                bcf     portflag,3      ; reset button-request
                goto    pbuttn05
pbuttn33        bsf     portflag,3      ; button is pressed
; check for pushbutton 4
pbuttn04        movlw   b'01111111'
                movwf   PORTB           ; set PORTB,7 for button-test
                call    wait01ms
                btfss   PORTB,1 ;test if button is #4 pressed
                goto    pbuttn44
                btfss   portflag,4
                goto    pbuttn05
                call    butnfct4        ; ===> execute function for button 4
                bcf     portflag,4      ; reset button-request
                goto    pbuttn05
pbuttn44        bsf     portflag,4      ; button is pressed
; ====== display bytes read during data-taking (raw data)
pbuttn05        btfss   flagtest,7      ;check, if a byte has been read
                goto    waitkeys
                movf    dataread,W
                bcf     flagtest,7      ; clear flag
                call    wrdigLCD        ; write byte read in hexa
                movf    dataread,W
                call    storcram        ; => store byte into RAM
; ====== stop data-taking if buffer is full
                btfss   xramflag,1      ; buffer is full
                goto    pbuttn55
```

```
                call    butnfc01        ; switch off data-taking
                goto    waitkeys
pbuttn55        incf    datacntr,F      ; count bytes written
                movf    datacntr,W
                sublw   008
                btfss   STATUS,Z        ; counter = 8 ?
                goto    waitkeys
                call    line2LCD        ;LCD line 2
                call    wait02ms
                clrf    datacntr        ; data counter
                goto    waitkeys
;
;-----------------------------------------------------------
;
                org     0x300           ;=> page 3
;
; store the first 5 bytes separately,
; index in ixcomand (0..4), byte in comdchar
storbyte        movlw   005
                subwf   ixcomand,W
                btfsc   STATUS,C
                goto    storbytx        ; (ix >= 5)
                movlw   003
                movwf   PCLATH          ;=> page 3
                movf    ixcomand,W
                addwf   ixcomand,W
                addwf   ixcomand,W      ; => ixcomand * 3
                addwf   PCL,F           ; ===> computed GOTO
                movf    comdchar,W      ;=0
                movwf   comand00
                goto    storbytx
                movf    comdchar,W      ;=1
                movwf   comand01
                goto    storbytx
                movf    comdchar,W      ;=2
                movwf   comand02
                goto    storbytx
                movf    comdchar,W      ;=3
                movwf   comand03
                goto    storbytx
                movf    comdchar,W      ;=4
                movwf   comand04
storbytx        incf    ixcomand,F      ; count # of bytes for command
                return
;
; ---------- write mnemonic-text-string onto LCD; # of text in W
; format: |0gggtttt| ggg group-nr. of text, tttt # of text
wrcomand        movwf   txtcount        ; save text-address
                andlw   0x0F
                movwf   txtindex
                bcf     STATUS,C        ;C=0
                rlf     txtindex,F
                rlf     txtindex,F      ;index*4 (0..63)
                swapf   txtcount,W
                andlw   0x07
                movwf   txtgroup        ;text-group (0..7)
                movlw   0x04
                movwf   txtcount
; loop to output the text
wmtext01        call    rdtextcr        ; read char. from text-group
                call    writeLCD        ;write to LCD
                incf    txtindex,F      ;count byte
                decfsz  txtcount,F
                goto    wmtext01        ;next byte
                return
;
; read byte of selected text: #of text-group / #of text-line
rdtextcr        movlw   003
                movwf   PCLATH          ;=> page 3
                movf    txtgroup,W      ; text-group    0..7
                addwf   PCL,F           ; ===> computed GOTO
                goto    txtcomd0        ;=0
                goto    txtcomd1        ;=1
```

```
                goto    txtcomd2        ;=2
                goto    txtcomd3        ;=3
                goto    txtcomd4        ;=4
                goto    txtcomd5        ;=5
                goto    txtcomd6        ;=6
                goto    txtcomd7        ;=7
;
;-------------------------------------------------------------
;
; text with short mnemonics for the DiSEqC-commands
; ---------- addresses/devices
txtcomd7        movlw   003
                movwf   PCLATH          ;=> page 3
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "anyD"  ;0x
                dt      "LNBs"  ;1x
                dt      "Polr"  ;2x
                dt      "Posi"  ;3x
                dt      "AID "  ;4x
                dt      "...."  ;5x
                dt      "Adrr"  ;6x
                dt      "ISIs"  ;7x
                dt      "...."  ;8x
                dt      "...."  ;9x
                dt      "...."  ;Ax
                dt      "...."  ;Bx
                dt      "...."  ;Cx
                dt      "...."  ;Dx
                dt      "...."  ;Ex
                dt      "...."  ;Fx
;
; ---------- commands
txtcomd0        movlw   003
                movwf   PCLATH          ;=> page 3
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "RSTm"  ;00
                dt      "clrf"  ;01
                dt      "Stby"  ;02
                dt      "pwrO"  ;03
                dt      "sCon"  ;04
                dt      "cont"  ;05
                dt      "Ccon"  ;06
                dt      "Adrs"  ;07
                dt      "movC"  ;08
                dt      "move"  ;09
                dt      "...."  ;0A
                dt      "...."  ;0B
                dt      "...."  ;0C
                dt      "...."  ;0D
                dt      "...."  ;0E
                dt      "...."  ;0F
;
                org     0x400           ;=> page 4
;------------set peripherals
txtcomd1        movlw   004
                movwf   PCLATH          ;=> page 4
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "rdSt"  ;10
                dt      "rCfg"  ;11
                dt      "...."  ;12
                dt      "...."  ;13
                dt      "Swt0"  ;14
                dt      "Swt1"  ;15
                dt      "Swt2"  ;16
                dt      "Swt3"  ;17
                dt      "...."  ;18
                dt      "...."  ;19
                dt      "...."  ;1A
                dt      "...."  ;1B
                dt      "...."  ;1C
```

```
                dt      "...."  ;1D
                dt      "...."  ;1E
                dt      "...."  ;1F
;
;------------set switches
txtcomd2        movlw   004
                movwf   PCLATH          ;=> page 4
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "stLO"  ;20
                dt      "stVR"  ;21
                dt      "stPA"  ;22
                dt      "sS0A"  ;23
                dt      "stHI"  ;24
                dt      "stHL"  ;25
                dt      "stPB"  ;26
                dt      "sS0B"  ;27
                dt      "sS1A"  ;28
                dt      "sS2A"  ;29
                dt      "sS3A"  ;2A
                dt      "sS4A"  ;2B
                dt      "sS1B"  ;2C
                dt      "sS2B"  ;2D
                dt      "sS3B"  ;2E
                dt      "sS4B"  ;2F
;
;------------set ports
txtcomd3        movlw   004
                movwf   PCLATH          ;=> page 4
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "slep"  ;30
                dt      "wake"  ;31
                dt      "...."  ;32
                dt      "...."  ;33
                dt      "...."  ;34
                dt      "...."  ;35
                dt      "...."  ;36
                dt      "...."  ;37
                dt      "wrN0"  ;38
                dt      "wrN1"  ;39
                dt      "wrN2"  ;3A
                dt      "wrN3"  ;3B
                dt      "...."  ;3C
                dt      "...."  ;3D
                dt      "...."  ;3E
                dt      "...."  ;3F
;
                org     0x500           ;=> page 5
;------------read/write analogous values
txtcomd4        movlw   005
                movwf   PCLATH          ;=> page 5
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "rdA0"  ;40
                dt      "rdA1"  ;41
                dt      "...."  ;42
                dt      "...."  ;43
                dt      "...."  ;44
                dt      "...."  ;45
                dt      "...."  ;46
                dt      "...."  ;47
                dt      "wrA0"  ;48
                dt      "wrA1"  ;49
                dt      "...."  ;4A
                dt      "...."  ;4B
                dt      "...."  ;4C
                dt      "...."  ;4D
                dt      "...."  ;4E
                dt      "...."  ;4F
;
;------------frequencies
txtcomd5        movlw   005
```

```
                movwf   PCLATH          ;=> page 5
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "rLOs"  ;50
                dt      "rLOf"  ;51
                dt      "rLOl"  ;52
                dt      "rLOh"  ;53
                dt      "...."  ;54
                dt      "...."  ;55
                dt      "...."  ;56
                dt      "...."  ;57
                dt      "wrFq"  ;58
                dt      "wrCh"  ;59
                dt      "...."  ;5A
                dt      "...."  ;5B
                dt      "...."  ;5C
                dt      "...."  ;5D
                dt      "...."  ;5E
                dt      "...."  ;5F
;
;------------positioner
txtcomd6        movlw   005
                movwf   PCLATH          ;=> page 5
                movf    txtindex,W      ; text-line 0..15
                addwf   PCL,F           ; ===> computed GOTO
                dt      "halt"  ;60
                dt      "...."  ;61
                dt      "...."  ;62
                dt      "limt"  ;63
                dt      "rPSR"  ;64
                dt      "...."  ;65
                dt      "limE"  ;66
                dt      "limW"  ;67
                dt      "drvE"  ;68
                dt      "drvW"  ;69
                dt      "stoN"  ;6A
                dt      "gotN"  ;6B
                dt      "...."  ;6C
                dt      "...."  ;6D
                dt      "gotX"  ;6E
                dt      "stPO"  ;6F
;
                end
;
;================== END ========================================
```