```
;########################  Dog Text Displays ############################################
;########################  Controller ST7036 ############################################


                    ; lcd_init
                    ; init_seq
                    ; lcd_wrcmd
                    ; lcd_wrdat
                    ; print_str
                    ; print_string
                    ; wrddram
                    ; set_ddram
                    ; clear_disp
                    ; go_sec_line
                    ; go_thrd_line
                    ; lcd_home
                    ; spi_out
                    ; del_1ms
                    ; delay30us
                    ; delay40ms
                    ; ;HexToDec
                    ; ;ASCIIToBin
                    ; ;BinToASCII
                    ; ;ByteToHex

;-------------------------------------------------------------------------------------
                    .if use_lcd_init == 1
lcd_init:             push r17
                      push r18
                      push ZL
                      push ZH

SPI_MasterInit:       sbi lcctl,lc_cs           ; Outputs setzen
                      sbi lcctl,lc_rs
                      sbi lcctl,lc_reset
                      sbi lcdat,lc_dat
                      sbi lcdat,lc_clk

                      .if use_reset == 1
                        dog_reset_off
                      .endif

                      rcall delay40ms

                      dog_enab
                      set_inst

                      .if (m48_88_168==1)|(m162==1)
                        ldi r17,(1<<SPE)|(1<<MSTR) ; Enable SPI, Master, set clock rate fck/4
                        out SPCR,r17
                        ldi r17,1<<SPI2X          ; Double Speed
                        out SPSR,r17
                      .endif

                      ldi ZL,Low(2*init_seq)
                      ldi ZH,High(2*init_seq)

                      ldi r17,6
lcd_init10:           lpm r18,Z+
                      rcall lcd_wrcmd
                      dec r17
                      brne lcd_init10

                      ldi r18,5
lcd_init20:           rcall delay40ms           ; 200ms nach 'Follower Control' warten
                      dec r18
                      brne lcd_init20

                      lpm r18,Z+
                      rcall lcd_wrcmd
```

```
                        ldi r17,3
lcd_init30:             lpm r18,Z+
                        rcall lcd_wrcmd
                        cpi r18,$01
                        brne lcd_init50

                        ldi r18,1
lcd_init40:             rcall delay40ms            ; >2ms für Clr/Home
                        dec r18
                        brne lcd_init40

lcd_init50:             dec r17
                        brne lcd_init30

                        pop ZH
                        pop ZL
                        pop r18
                        pop r17
                        ret

;-----------------------------------------------------------------------------------
                        .if (dogm081==1) & (u5v0==1)
;                             _____________________________________________ Function Set (Tab 0)
;                            |  ___________________________________________ Function Set (Tab 1)
;                            |  |  ________________________________________ Bias Set
;                            |  |  |  _____________________________________ Contrast Set
;                            |  |  |  |  __________________________________ Power Control
;                            |  |  |  |  |  _______________________________ Follower Control
;                            |  |  |  |  |  |  ____________________________ Display ON/OFF
;                            |  |  |  |  |  |  |  _________________________ Clear Display
;                            |  |  |  |  |  |  |  |  ______________________ Entry Mode Set
;                            |  |  |  |  |  |  |  |  |  |
init_seq:                     .db $38,$39,$14,$7F,$53,$6A,$0C,$01,$06,$00
;                                $31,$30,$1C,$74,$51,$6A,$0F,$01,$06,$00

                        .endif
;-----------------------------------------------------------------------------------
                        .if (dogm162==1) & (u5v0==1)
;                             _____________________________________________ Function Set (Tab 0)
;                            |  ___________________________________________ Function Set (Tab 1)
;                            |  |  ________________________________________ Bias Set
;                            |  |  |  _____________________________________ Contrast Set
;                            |  |  |  |  __________________________________ Power Control
;                            |  |  |  |  |  _______________________________ Follower Control
;                            |  |  |  |  |  |  ____________________________ Display ON/OFF
;                            |  |  |  |  |  |  |  _________________________ Clear Display
;                            |  |  |  |  |  |  |  |  ______________________ Entry Mode Set
;                            |  |  |  |  |  |  |  |  |  |
init_seq:                     .db $38,$39,$14,$7F,$53,$6A,$0C,$01,$06,$00

;                             .db $39,$1C,$52,$69,$74,$38,$0F,$01,$06,$00
                        .endif
;-----------------------------------------------------------------------------------
                        .if (dogm163==1) & (u5v0==1)
;                             _____________________________________________ Function Set (Tab 0)
;                            |  ___________________________________________ Function Set (Tab 1)
;                            |  |  ________________________________________ Bias Set
;                            |  |  |  _____________________________________ Contrast Set
;                            |  |  |  |  __________________________________ Power Control
;                            |  |  |  |  |  _______________________________ Follower Control
;                            |  |  |  |  |  |  ____________________________ Function Set (Tab 0)
;                            |  |  |  |  |  |  |  _________________________ Display ON/OFF
;                            |  |  |  |  |  |  |  |  ______________________ Clear Display
;                            |  |  |  |  |  |  |  |  |  _____________________ Entry Mode Set
;                            |  |  |  |  |  |  |  |  |  |  |
init_seq:                     .db $38,$39,$1D,$75,$50,$6C,$38,$0C,$01,$06,$00,$00
;                                6   5   4   3   2   1
;                                                    3   2   1

                        .endif
                      .endif
;-----------------------------------------------------------------------------------
;                     Kommando schreiben
;                     in : r18 Kommando
```

```
                    .if use_lcd_wrcmd == 1
lcd_wrcmd:            set_inst
                      rcall spi_out
                      rcall delay30us
                      ret
                    .endif
;-----------------------------------------------------------------------------
;                   Daten schreiben
;                   in : r18 Datenbyte

                    .if use_lcd_wrdat == 1
lcd_wrdat:            set_data
                      rcall spi_out
                      rcall Delay30us
                      ret
                    .endif
;-----------------------------------------------------------------------------
;                   Schreibt nachfolgenden String
;
;                   (r)call print_str
;                   .db "......."

                    .if use_print_str      == 1
print_str:            push r18
                      push r19
                      push ZL
                      push ZH

                      in ZL,SPL
                      in ZH,SPH               ; Aufrufadress
                      adiw ZH:ZL,5            ; + Adressoffset
                      ld r19,Z+
                      ld r18,Z
                      movw ZH:ZL,r19:r18      ; Adresse String
                      lsl ZL
                      rol ZH                  ; ->Byteadresse

print_str10:          lpm r18,Z+
                      tst r18
                      breq print_str20
                      rcall lcd_wrdat
                      rjmp print_str10

print_str20:          adiw ZH:ZL,1            ; Ungerade Byteantahl
                      lsr ZH                  ; ausgleichen
                      ror ZL
                      movw r19:r18,ZH:ZL      ; Rückkehradresse

                      in ZL,SPL
                      in ZH,SPH
                      adiw ZH:ZL,5            ; Stackadresse

                      st Z+,r19
                      st Z,r18                ; Rückkehradresse speichern

                      pop ZH
                      pop ZL
                      pop r19
                      pop r18
ret_adr:              ret
                    .endif
;-----------------------------------------------------------------------------;-----------
------------------------------------------------------------------
;                   Text schreiben
;                   in : Z Adresse Text (Nullterminiert)
;                   changed : Z

                    .if use_print_string == 1
print_string:         push r18
                      push ZL
                      push ZH

                      lsl ZL
```

```
                        rol ZH

print_string10:         lpm r18,Z+
                        tst r18
                        breq print_string20
                        rcall lcd_wrdat
                        rjmp print_string10
print_string20:         pop ZH
                        pop ZL
                        pop r18
                        ret
                      .endif
;-------------------------------------------------------------------------------
;                     Schreiben auf DDRAM-Adresse
;                     in      :  r19 = Datenbyte
;                                r18 = Adresse (7Bit)
;                     changed: -
;
                      .if use_wrddram == 1
wrddram:                ori r18,$80           ;Bit7 setzen(Befehl Adresse schreiben)
                        rcall lcd_wrcmd       ;zur LCD
                        mov r18,r19           ;Daten
                        rcall lcd_wrdat       ;zur LCD
                        ret
                      .endif

;-------------------------------------------------------------------------------
;                     DDRAM Adresse setzen
;                     in      : r18 = Adresse
;                     out     : -
;                     changed: -

                      .if use_set_ddram == 1
set_ddram:              push r18
                        andi r18,$7F
                        ori r18,ddram_set
                        rcall lcd_wrcmd
                    rcall Delay30us
                        pop r18
                        ret
                      .endif

;-------------------------------------------------------------------------------
;                     Clear Display
;                     in: -

                      .if use_clear_disp == 1
clear_disp:             push r18
                        ldi r18,0b00000001
                        rcall lcd_wrcmd
                        rcall del_1ms
                        pop r18
                        ret
                      .endif

;-------------------------------------------------------------------------------
;                     Anfang 2.Zeile
;                     in: -

                      .if use_go_sec_line == 1
go_sec_line:            push r18
                        ldi r18,ddram_set+sec_line
                        rcall lcd_wrcmd
                        rcall del_1ms
                        pop r18
                        ret
                      .endif

;-------------------------------------------------------------------------------
;                     Anfang 3.Zeile
;                     in: -

                      .if use_go_thrd_line == 1
go_thrd_line:           push r18
```

```
                        ldi r18,ddram_set+thrd_line
                        rcall lcd_wrcmd
                        rcall del_1ms
                        pop r18
                        ret
                      .endif

;-----------------------------------------------------------------------------
;                     return Home
;                     in: -

                      .if use_lcd_home == 1
lcd_home:               push r18
                        ldi r18,0b00000010
                        rcall lcd_wrcmd
                        rcall del_1ms
                        pop r18
                        ret
                      .endif

;-----------------------------------------------------------------------------
;                     CG-Ram laden

;                     in  : Z - Tabelle
;                           r17 Anzahl Zeichen (á 8 Byte)

                      .if use_set_cgram == 1
set_cgram:              push r17
                        push r18
                        push ZL
                        push ZH

                        lsl r17
                        lsl r17
                        lsl r17

                        ldi r18,cgram_set
                        rcall lcd_wrcmd

set_cgram10:            lpm r18,Z+
                        rcall lcd_wrdat

                        dec r17
                        brne set_cgram10

                        pop ZH
                        pop ZL
                        pop r18
                        pop r17
                        ret
                      .endif

;-----------------------------------------------------------------------------
;                     Byte über SPI ausgeben
;                     in: r18 Datenbyte
;                     changed : -

;                     ATTiny 2313

                      .if use_spi_out == 1
                        .if t2313==1
spi_out:                  push r18
                          out USIDR,r18
                          ldi r18,(1<<USIOIF)      ; Overflow Interrupt Flag löschen
                          out USISR,r18
                          ldi r18,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)  ; 3-Wire
spi_out_10:               out USICR,r18            ; pos.Edge,Soft CLK, Toggle Clk
                          sbis USISR,USIOIF        ; Fertig?
                          rjmp spi_out_10
                          pop r18
                          ret
                          .message "Target: ATTiny2313"
                        .endif
;-----------------------------------------------------------------------------
```

```
;                       ATMega 8/48/88/168/162

                          .if (m48_88_168 ==1)||(m8==1)||(m162==1)
spi_out:                    push r18
                            dog_enab
                            out SPDR,r18                ; Start transmission of data (r16)
spi_out10:                  in r18, SPSR                ; Wait for transmission complete
                            sbrs r18, SPIF
                            rjmp spi_out10
                            dog_disab
                            pop r18
                            ret
                            .message "Target: ATMega8/48/88/168/162 SPI"
                          .endif
                        .endif                      ;use_spi_out

;-------------------------------------------------------------------------------------
;                         Software SPI

;                         in r18 : Zeichen

                        .if soft_spi== 1
spi_out:                  push r16
                          push r18
                          ldi r16,8
spi_out10:                cbi lcdato,lc_dat
                          cbi lcdato,lc_clk
                          sbrc r18,7
                          sbi lcdato,lc_dat
                          lsl r18
                          nop
                          sbi lcdato,lc_clk
                          nop
                          dec r16
                          brne spi_out10
                          cbi lcdato,lc_clk
                          pop r18
                          pop r16
                          ret
                          .message "Software SPI"
                        .endif

;-------------------------------------------------------------------------------------
;                         Fast SPI using USI

                        .if fast_usi == 1
spi_out:                  push r16
                          push r17
                          out USIDR,r18
                          ldi r16,(1<<USIWM0)|(1<<USITC)
                          ldi r17,(1<<USIWM0)|(1<<USITC)|(1<<USICLK)
                          out USICR,r16 ; MSB
                          nop
                          out USICR,r17
                          nop
                          out USICR,r16
                          nop
                          out USICR,r17
                          nop
                          out USICR,r16
                          nop
                          out USICR,r17
                          nop
                          out USICR,r16
                          nop
                          out USICR,r17
                          nop
                          out USICR,r16
                          nop
                          out USICR,r17
                          nop
                          out USICR,r16
                          nop
                          out USICR,r17
```

```
                        nop
                        out USICR,r16
                        nop
                        out USICR,r17
                        nop
                        out USICR,r16 ; LSB
                        nop
                        out USICR,r17
                        pop r17
                        pop r16
                        ret

                        .message "Fast SPI with USI"
                      .endif
;-----------------------------------------------------------------------------------
                      ; Delay 770µs @ 4,00000 MHz

                      .if use_del_1ms == 1
del_1ms:                push r16
                        push r17
                        push r18

                        .if Clk8MHz == 1
                          ldi r18,2
                        .elif Clk12MHz == 1
                          ldi r18,4
                        .elif Clk16MHz == 1
                          ldi r18,8
                        .else
                          ldi r18,1
                        .endif

del_1ms_10:              ldi r17,170  ;86   ;171
del_1ms_20:              ldi r16,5
del_1ms_30:              dec r16
                         brne del_1ms_30

                         dec r17
                         brne del_1ms_20

                         dec r18
                         brne del_1ms_10

                         pop r18
                         pop r17
                         pop r16
                         ret
                      .endif
;-----------------------------------------------------------------------------------
                      ; Delay 30µs @ 4,00000 MHz

                      .if use_del_30us == 1
delay30us:              push r16
                        push r18
                        .if Clk8MHz == 1
                          ldi r18,2
                        .elif Clk12MHz == 1
                          ldi r18,4
                        .elif Clk16MHz == 1
                          ldi r18,8
                        .else
                          ldi r18,2
                        .endif
delay30us_10:           ldi r16,35  ;21   ;35
delay30us_20:           dec r16
                        brne delay30us_20

                        dec r18
                        brne delay30us_10

                        pop r18
                        pop r16
                        ret
                      .endif
```

```
;-------------------------------------------------------------------------------
                    ; Delay 40ms @ 4,00000 MHz

                    .if use_del_40ms == 1
delay40ms:            push r16
                      push r17
                      push r18

                      .if Clk8MHz == 1
                        ldi r18,2
                      .elif Clk12MHz == 1
                        ldi r18,4
                      .elif Clk16MHz == 1
                        ldi r18,8
                      .else
                        ldi r18,1
                      .endif

delay40ms_10:         ldi r17,60   ;236
delay40ms_20:         ldi r16,225
delay40ms_30:         nop
                      nop
                      nop
                      dec r16
                      brne delay40ms_30

                      dec r17
                      brne delay40ms_20

                      dec r18
                      brne delay40ms_10

                      pop r18
                      pop r17
                      pop r16
                      ret
                    .endif
/*
;============================ Zahlenkonvertierungen ====================================
;                 Hexadezimal nach Dezimal
;                 in   : X Hex        <=999D(3E7)
;                 out  : r20 1er
;                        r21 10er
;                        r22 100er
;                        r23 1000er

                    .if use_ASCIIToBin == 1

HexToDec:         push XL
                  push XH
                  push YL
                  push YH
                  clr r20
                  clr r21
                  clr r22
                  clr r23
                  ldi YL,low(1000)
                  ldi YH,high(1000);
HexToDec01:       sub XL,YL
                  sbc XH,YH
                  brcs HexToDec05
                  inc r23
                  rjmp HexToDec01

HexToDec05:       add XL,YL
                  adc XH,YH
HexToDec10:       sbiw XL,50
                  brcs HexToDec20
                  sbiw XL,50
                  brcs HexToDec15
                  inc r22
                  rjmp HexToDec10
HexToDec15:       adiw XL,50
HexToDec20:       adiw XL,50
```

```
HexToDec30:         sbiw XL,10
                    brcs HexToDec40
                    inc r21
                    rjmp HexToDec30
HexToDec40:         adiw XL,10
                    mov r20,XL
                    pop YH
                    pop YL
                    pop XH
                    pop XL
                    ret
                  .endif
;-----------------------------------------------------------------------------------
;                   ASCII (0..9,A..F) nach Bin
;                   in  : r16 ASCII Zahl
;                   out : r16 Bin-Wert

                     .if use_HexToDec == 1

ASCIIToBin:         subi r16,$30
                    cpi r16,10
                    brcs ASCIIToBin10
                    subi r16,7
ASCIIToBin10:       ret
                  .endif
;-----------------------------------------------------------------------------------
;                   Hexadezimal nach ASCII
;                   in   : r20..r23 (bin)
;                   out  : r20..r23 (ASCII)

                     .if use_BinToASCII == 1

BinToASCII:         push r16
                    push r17
                    ldi r16,$30
                    ldi r17,$07
                    add r20,r16
                    cpi r20,$3A
                    brcs BinToASCII10
                    add r20,r17

BinToASCII10:       add r21,r16
                    cpi r21,$3A
                    brcs BinToASCII20
                    add r21,r17

BinToASCII20:       add r22,r16
                    cpi r22,$3A
                    brcs BinToASCII30
                    add r22,r17

BinToASCII30:       add r23,r16
                    cpi r23,$3A
                    brcs BinToASCII40
                    add r23,r17

BinToASCII40:       pop r17
                    pop r16
                    ret
                  .endif
;-----------------------------------------------------------------------------------
;                   in : r16 Byte
;                   out: r9  ASCII LSB's
;                        r10 ASCII MSB's

                     .if use_ByteToHex == 1

ByteToHex:             push r16
                    push r17
                    push r16
                    ldi r17,$30
                    andi r16,$0F
                    add r16,r17
                    cpi r16,$3A
```

```
                    brcs ByteToHex10
                    subi r16,-7
ByteToHex10:        mov r9,r16
                    pop r16
                    swap r16
                    andi r16,$0F
                    add r16,r17
                    cpi r16,$3A
                    brcs ByteToHex20
                    subi r16,-7
ByteToHex20:        mov r10,r16
                    pop r17
                    pop r16
                    ret
                  .endif
;-----------------------------------------------------------------------------------
*/
```