```
;***************************************************************************
;* Atmel AVR-microcontroller  and 8 1/2 digit LC-display                   *
;* PCB Layout : Eagle light 4.0                                            *
;* Christoph Kessler 2006 - 2007                    db1uq (at) t-online.de *
;***************************************************************************
.nolist
.include "m8515def.inc"
.list
;***************************************************************************
;* register 0 ( special purpose: 16Bit-LPM-operation ):
;***************************************************************************
.def    LpmReg  = r0    ; for table-operations
;***************************************************************************
;* register 1-15 ( no "immediate"-operations possible ):
;***************************************************************************
.def    reg01   = r1    ;
.def    reg02   = r2    ;
.def    reg03   = r3    ;
.def    reg04   = r4    ;
.def    reg05   = r5    ;
.def    reg06   = r6    ;
.def    reg07   = r7    ;
.def    dd32u0  = r8    ; dividend/result byte 0 (LSB)
.def    dd32u1  = r9    ; dividend/result byte 1
.def    dd32u2  = r10   ; dividend/result byte 2
.def    dd32u3  = r11   ; dividend/result byte 3
.def    dd32u4  = r12   ; dividend/result byte 4
.def    dd32u5  = r13   ; dividend/result byte 5
.def    dd32u6  = r14   ; dividend/result byte 6
.def    dd32u7  = r15   ; dividend/result byte 7 (MSB)
;***************************************************************************
;* register 16-25 ( "immediate"-operations possible ):
;***************************************************************************
.def    Tmp1    = r16   ;
.def    Tmp2    = r17   ;
.def    Tmp3    = r18   ;
.def    Cnt     = r19   ;
.def    fbin0   = r16   ; binary value byte 0 (LSB)
.def    fbin1   = r17   ; binary value byte 1
.def    fbin2   = r18   ; binary value byte 2
.def    fbin3   = r19   ; binary value byte 3 (MSB)
.def    tBCD0   = r20   ; BCD value digits 0 and 1 (same as fbin2)
.def    tBCD1   = r21   ; BCD value digits 2 and 3 (same as fbin3)
.def    tBCD2   = r22   ; BCD value digits 4 and 5
.def    tBCD3   = r23   ; BCD value digits 6 and 7
.def    tBCD4   = r24   ; BCD value digits 8 and 9 (MSD)

.def    dv32u0  = r16   ; divisor byte 0 (LSB)
.def    dv32u1  = r17   ; divisor byte 1
.def    dv32u2  = r18   ; divisor byte 2
.def    dv32u3  = r19   ; divisor byte 3 (MSB)
.def    dres32u0 = r20  ; result byte 0 (LSB)
.def    dres32u1 = r21  ; result byte 1
.def    dres32u2 = r22  ; result byte 2
.def    dres32u3 =r23   ; result byte 3 (MSB)
.def    drem32u0 = r24  ; remainder byte 0 (LSB)
.def    drem32u1 = r25  ; remainder byte 1
.def    drem32u2 = r26  ; remainder byte 2
.def    drem32u3 = r27  ; remainder byte 3 (MSB)
.def    dcnt32u = r28   ; loop counter

;***************************************************************************
;* register 24/25    ( also usable as 16Bit-register )
;***************************************************************************
;.def   reg24   = r24   ; general temporary register (low)
.def    reg25   = r25   ; general temporary register (high)
;***************************************************************************
;* register 26-31 ( 16Bit-register ) already defined in "inc"-file:
;***************************************************************************
;       XL      = r26   ;  low
;       XH      = r27   ;  high
;       YL      = r28   ;  low
;       YH      = r29   ;  high
;       ZL      = r30   ;  low
;       ZH      = r31   ;  high
;***************************************************************************
;* constants :
;***************************************************************************
; Port A connections:
.equ    PrtA7   = 7     ; PortA Bit7 = not used
.equ    PrtA6   = 6     ; PortA Bit6 = not used
.equ    PrtA5   = 5     ; PortA Bit5 = not used
.equ    PrtA4   = 4     ; PortA Bit4 = not used
.equ    Latch   = 3     ; PortA Bit3 = output latch digit active high
.equ    Dig2    = 2     ; PortA Bit2 = output digit bit2
.equ    Dig1    = 1     ; PortA Bit1 = output digit bit1
.equ    Dig0    = 0     ; PortA Bit0 = output digit bit0
; Port B connections:
.equ    PrtB7   = 7     ; PortB Bit7 = not used
.equ    PrtB6   = 6     ; PortB Bit6 = not used
.equ    PrtB5   = 5     ; PortB Bit5 = not used
.equ    PrtB4   = 4     ; PortB Bit4 = output H=count/L=stop
```

```
.equ    PrtB3   = 3     ; PortB Bit3 = input QB
.equ    PrtB2   = 2     ; PortB Bit2 = input QC
.equ    PrtB1   = 1     ; PortB Bit1 = input QD =T1 count inp.
.equ    PrtB0   = 0     ; PortB Bit0 = input QA(=T0 count inp.)
; Port C connections:
.equ    PrtC7   = 7     ; PortC Bit7 = not used
.equ    DP      = 6     ; PortC Bit6 = output store decimal point active low
.equ    ONE     = 5     ; PortC Bit5 = output leading "one" digit
.equ    KILO    = 4     ; PortC Bit4 = output "k"Hz digit
.equ    DB3     = 3     ; PortC Bit3 = output LCD-databus Bit 3
.equ    DB2     = 2     ; PortC Bit2 = output LCD-databus Bit 2
.equ    DB1     = 1     ; PortC Bit1 = output LCD-databus Bit 1
.equ    DB0     = 0     ; PortC Bit0 = output LCD-databus Bit 0
; Port D connections:
.equ    PrtD7   = 7     ; PortD Bit7 = not used
.equ    PrtD6   = 6     ; PortD Bit6 = not used
.equ    PrtD5   = 5     ; PortD Bit5 = not used
.equ    PrtD4   = 4     ; PortD Bit4 = not used
.equ    PrtD3   = 3     ; PortD Bit3 = input INT1 H=count/L=stop
.equ    PrtD2   = 2     ; PortD Bit2 = output Mux Bit 2
.equ    PrtD1   = 1     ; PortD Bit1 = output Mux Bit 1
.equ    PrtD0   = 0     ; PortD Bit0 = output Mux Bit 0
;***************************************************************************
;* reset/interrupt-vectors:
;***************************************************************************
.cseg
        rjmp    Initial         ; after RESET to main program
        nop                     ; interrupt rotator switch channel A
        nop                     ; interrupt rotator switch channel B
        nop                     ; Timer1_capture
        nop                     ; Timer1_compareA
        nop                     ; Timer1_compareB
        nop                     ; Timer1_overflow
        nop                     ; Timer0_overflow
        nop                     ; SPI transfer
        nop                     ; UART Rx
        nop                     ; UART Empty
        nop                     ; UART Tx
        nop                     ; Analog comparator
;***************************************************************************
;* after reset: initialise stack,ports
;***************************************************************************
Initial:
        ldi     Tmp1,Low(RamEnd)
        out     spl,Tmp1        ;
        ldi     Tmp1,High(RamEnd)
        out     sph,Tmp1        ; stack initialised
        clr     Tmp1            ;
        out     PortA,Tmp1      ;
        ldi     Tmp1,$10        ; PortC = % 0001 0000, /LD = high
        out     PortB,Tmp1      ;
        ldi     Tmp1,$40        ; PortC = % 0100 0000, DP latch active low
        out     PortC,Tmp1      ;
        ldi     Tmp1,$07        ; PortD = % 0000 0111, Mux inputD7
        out     PortD,Tmp1      ;
        ldi     Tmp1,$0F        ; PortA direction = % 0000 1111
        out     DDRA,Tmp1       ;
        ldi     Tmp1,$10        ; PortB direction = % 0001 0000
        out     DDRB,Tmp1       ;
        ldi     Tmp1,$7F        ; PortC direction = % 0111 1111
        out     DDRC,Tmp1       ;
        ldi     Tmp1,$07        ; PortD direction = % 0000 0111
        out     DDRD,Tmp1       ;
;***************************************************************************
;* Main program: Test
;***************************************************************************
        ldi     ZH,high(FrqBuf+9)
        ldi     ZL,low(FrqBuf+9)
        ldi     Tmp1,$24        ; Kommapunkte ein
        st      Z,Tmp1          ;
        rcall   DPOut           ; gibt DPs im Buffer aus
        sbi     PortC,4         ; "k"

        ldi     Tmp1,$00        ;
        mov     dd32u7,Tmp1     ;
        ldi     Tmp1,$23        ;
        mov     dd32u6,Tmp1     ;
        ldi     Tmp1,$86        ; 23 86F2 6FC1 0000 = (100Mio)^2
        mov     dd32u5,Tmp1     ;
        ldi     Tmp1,$F2        ;
        mov     dd32u4,Tmp1     ;
        ldi     Tmp1,$6F        ;
        mov     dd32u3,Tmp1     ;
        ldi     Tmp1,$C1        ;
        mov     dd32u2,Tmp1     ;
        ldi     Tmp1,$00        ;
        mov     dd32u1,Tmp1     ;
        ldi     Tmp1,$00        ;
        mov     dd32u0,Tmp1     ;

        ldi     dv32u3,$0B      ;
        ldi     dv32u2,$EB      ;
```

```
        ldi     dv32u1,$C2      ; 05F5E100 hex = 100 Mio dez
        ldi     dv32u0,$00      ; 0BEBC200 hex = 200 Mio dez

        rcall   Div6432         ; 32 Bit Division

;       mov     fbin0,dv32u0    ;
;       mov     fbin1,dv32u1    ;
;       mov     fbin2,dv32u2    ;
;       mov     fbin3,dv32u3    ;
        mov     fbin0,dd32u0    ;
        mov     fbin1,dd32u1    ;
        mov     fbin2,dd32u2    ;
        mov     fbin3,dd32u3    ;
        rcall   Bin4BCD         ;
        ldi     ZH,high(FrqBuf)
        ldi     ZL,low(FrqBuf)  ; Z-Ptr zeigt auf Textanfang
        mov     Tmp1,tBCD4      ;
        andi    Tmp1,$01        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD3      ;
        swap    Tmp1            ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD3      ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD2      ;
        swap    Tmp1            ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD2      ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD1      ;
        swap    Tmp1            ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD1      ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD0      ;
        swap    Tmp1            ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        mov     Tmp1,tBCD0      ;
        andi    Tmp1,$0F        ;
        st      Z+,Tmp1         ;
        rcall   LcdOut          ; gibt Ziffern im Buffer aus
Test3:
        rjmp    Test3           ; Dauerschleife

;***************************************************************************
;* Subroutine from Math32 library by Andre Birua
;* Div32 == 32/32 Bit Unsigned Division
;* dd32uL::dd32uH / dv32uL::dv32uH = dres32uL::dres32uH (drem32uL::drem32uH)
;*    dividend      divisor         result     remainder
;* r20r21r22r23 / r16r17r18r19 = r20r21r22r23 r24r25r26r27
;***************************************************************************
Div6432:
        clr     drem32u0        ; clear 4 lower remainde byte
        clr     drem32u1        ;
        clr     drem32u2        ;
        clr     drem32u3        ;
        clc                     ;
        ldi     dcnt32u,64      ; init loop counter
d32u_loop:
        lsl     dd32u0          ; shift left dividend
        rol     dd32u1          ;
        rol     dd32u2          ;
        rol     dd32u3          ;
        rol     dd32u4          ;
        rol     dd32u5          ;
        rol     dd32u6          ;
        rol     dd32u7          ;
        rol     drem32u0        ; shift dividend into remainder
        rol     drem32u1        ;
        rol     drem32u2        ;
        rol     drem32u3        ;
        cp      drem32u0,dv32u0 ; remainder = remainder - divisor
        cpc     drem32u1,dv32u1 ;
        cpc     drem32u2,dv32u2 ;
        cpc     drem32u3,dv32u3 ;
        brcs    d32u_loop2      ; clear carry to be shifted into res
        inc     dd32u0          ;
        sub     drem32u0,dv32u0 ; if result negative
        sbc     drem32u1,dv32u1 ; restore remainder
        sbc     drem32u2,dv32u2 ;
        sbc     drem32u3,dv32u3 ;
d32u_loop2:
        dec     dcnt32u         ;
        brne    d32u_loop       ; set carry to be shifted into res
        ret                     ;
```

```
;***************************************************************************
;* Subroutine from Math32 library by Andre Birua
;* Bin4BCD == 32-bit Binary to BCD conversion   [ together with Bin2BCD ]
;* fbin0:fbin1:fbin2:fbin3 >>> tBCD0:tBCD1:tBCD2:tBCD3:tBCD4
;*      hex             dec
;* r16r17r18r19 >>> r20r21r22r23r24
;***************************************************************************
Bin4BCD:
        rcall   Bin2BCD         ;
        clr     tBCD3           ;initial highest bytes of result
        ldi     tBCD4,0xfe      ;
binbcd_loop:
        subi    tBCD0,-0x33     ;add 0x33 to digits 1 and 0
        sbrs    tBCD0,3         ;if bit 3 clear sub 3
        subi    tBCD0,0x03      ;
        sbrs    tBCD0,7         ;if bit 7 clear sub $30
        subi    tBCD0,0x30      ;
        subi    tBCD1,-0x33     ;add 0x33 to digits 3 and 2
        sbrs    tBCD1,3         ;if bit 3 clear sub 3
        subi    tBCD1,0x03      ;
        sbrs    tBCD1,7         ;if bit 7 clear sub $30
        subi    tBCD1,0x30      ;
        subi    tBCD2,-0x33     ;add 0x33 to digits 5 and 4
        sbrs    tBCD2,3         ;if bit 3 clear sub 3
        subi    tBCD2,0x03      ;
        sbrs    tBCD2,7         ;if bit 7 clear sub $30
        subi    tBCD2,0x30      ;
        lsl     fbin0           ;
        rol     fbin1           ;shift lower word
        rol     tBCD0           ;through all bytes
        rol     tBCD1           ;
        rol     tBCD2           ;
        rol     tBCD3           ;
        rol     tBCD4           ;
        brmi    binbcd_loop     ;7 shifts w/o correction of MSD
        rol     fbin3           ;since Bin2BCD fbin3 = 0xff
        brcc    binbcd_ret      ;so as to do 16_lsl in total
        subi    tBCD3,-0x33     ;add 0x33 to digits 7 and 6
        sbrs    tBCD3,3         ;if bit 3 clear sub 3
        subi    tBCD3,0x03      ;
        sbrs    tBCD3,7         ;if bit 7 clear sub $30
        subi    tBCD3,0x30      ;
        subi    tBCD4,-0x03     ;add 0x03 to digit 8 only
        sbrs    tBCD4,3         ;if bit 3 clear sub 3
        subi    tBCD4,0x03      ;
        rjmp    binbcd_loop     ;
;***************************************************************************
;* Subroutine from Math32 library by Andre Birua
;* Bin2BCD == 16-bit Binary to BCD conversion
;* fbin2:fbin3 >>> tBCD0:tBCD1:tBCD2
;* hex          dec
;* r18r19 >>> r20r21r22
;***************************************************************************
Bin2BCD:
        ldi     tBCD2,0xff      ;initialize digit 4
binbcd_4:
        inc     tBCD2           ;
        subi    fbin2,low(10000);subiw fbin,10000
        sbci    fbin3,high(10000)
        brcc    binbcd_4        ;
        ldi     tBCD1,0x9f      ;initialize digits 3 and 2
binbcd_3:
        subi    tBCD1,0x10      ;
        subi    fbin2,low(-1000);addiw fbin,1000
        sbci    fbin3,high(-1000)
        brcs    binbcd_3        ;
binbcd_2:
        inc     tBCD1           ;
        subi    fbin2,low(100)  ;subiw fbin,100
        sbci    fbin3,high(100) ;
        brcc    binbcd_2        ;
        ldi     tBCD0,0xa0      ;initialize digits 1 and 0
binbcd_1:
        subi    tBCD0,0x10      ;
        subi    fbin2,-10       ;addi fbin,10
        brcs    binbcd_1        ;
        add     tBCD0,fbin2     ;LSD
binbcd_ret:
        ret                     ;
;***************************************************************************
;* Subroutine: frequency output to LCD
;***************************************************************************
LcdOut:
        ldi     ZH,high(FrqBuf) ;
        ldi     ZL,low(FrqBuf)  ;
        ld      Tmp1,Z+         ; leading 0/1-digit
        lsr     Tmp1            ;
        brcc    lo              ;
        sbi     PortC,5         ; is "1"
        brcs    hi              ;
lo:
        cbi     PortC,5         ; is " "
```

```
hi:
        clr     Cnt             ; next 8 digits
LcdOu1:
        ld      Tmp1,Z+         ; fetch next digit
        in      Tmp2,PinC       ; fetch state of PortC
        andi    Tmp2,$F0        ; save upper 4 bits
        or      Tmp1,Tmp2       ; add BCD code of digit
        out     PortC,Tmp1      ; BCD-code to 4bit Databus
        in      Tmp2,PinA       ; fetch state of PortA
        andi    Tmp2,$F0        ; save upper 4 bits
        or      Tmp2,Cnt        ; add digit number
        out     PortA,Tmp2      ; 3bit address bus, enable=0
        sbi     PortA,3         ; enable _/~
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ; wait for slow CD4543
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        nop                     ;
        cbi     PortA,3         ; enable ~\_
        inc     Cnt             ; next digit
        cpi     Cnt,8
        brne    LcdOu1          ;
        ret                     ;
;***************************************************************************
;* Subroutine: decimal point output to LCD
;***************************************************************************
DPOut:
        ldi     ZH,high(FrqBuf+9)
        ldi     ZL,low(FrqBuf+9)
        ld      Tmp3,Z          ;
        clr     Cnt             ;
DPOu1:
        lsr     Tmp3            ; DP databit into Cy
        clr     Tmp1
        rol     Tmp1            ;
        rol     Tmp1            ;
        rol     Tmp1            ;
        rol     Tmp1            ; DP databit into Bit3
        or      Tmp1,Cnt        ; add counter Bits 2..0
        in      Tmp2,PinC       ; fetch state of PortC
        andi    Tmp2,$F0        ; mask bit 7..4
        or      Tmp1,Tmp2       ; add Bit 7..4
        out     PortC,Tmp1      ; to LCD
        cbi     PortC,6         ; /enable DP-latch ~\_
        sbi     PortC,6         ; /enable DP-latch _/~
        inc     Cnt             ; next digit
        cpi     Cnt,8
        brne    DPOu1           ;
        ret                     ;
;***************************************************************************
;* frequency display buffer in SRAM "199.999.999"=9 Byte + 8 DecimalPoints
;***************************************************************************
.dseg
.org $60
FrqBuf:
.BYTE 10
```