```
;*****************************************************************************************************************
***
;* Packet-Radio Transceiver with RFM12-module                                  *
;* ATmega48, 14.31818 MHz Xtal                                                 *
;* Christoph Kessler 2009                          dbluq_at_t-online_dot_de *
;*****************************************************************************************************************
***
.nolist
.include "m48def.inc"
.list
;*****************************************************************************************************************
***
;* register 0-15 (no "immediate"-operations possible):
;*****************************************************************************************************************
***
.def    MultL   = r0                    ; free (multiplier result)
.def    MultH   = r1                    ; free (multiplier result)
.def    Zero    = r2                    ; always zero
.def    SavSrg  = r3                    ; save status during ext. Interrupt
.def    SavTmp  = r4                    ; for fast push/pop (2 cycles, not 4)
.def    EncVal  = r5                    ; set by INT0/INT1, cleared by main prg.
.def    PLL_L   = r6                    ; 12 Bit PLL control word low byte
.def    PLL_H   = r7                    ; 12 Bit PLL control word high byte
.def    SavSrg2 = r8                    ; save status during timer Interrupt
.def    reg09   = r9                    ; free
.def    reg10   = r10                   ; free
.def    reg11   = r11                   ; free
.def    reg12   = r12                   ; free
.def    reg13   = r13                   ; free
.def    reg14   = r14                   ; free
.def    reg15   = r15                   ; free
;*****************************************************************************************************************
***
;* register 16-23 ("immediate"-operations possible):
;*****************************************************************************************************************
***
.def    Cnt     = r16                   ; loop counter
.def    reg21   = r17                   ; free
.def    reg22   = r18                   ; free
.def    reg23   = r19                   ; free
.def    Frq0    = r20                   ; frequency in kHz byte 0 (LSB)
.def    Frq1    = r21                   ; frequency in kHz byte 1
.def    Frq2    = r22                   ; frequency in kHz byte 2 (MSB)
.def    Frq3    = r23                   ; for 32bit multiplication result
;*****************************************************************************************************************
***
;* register 24-31    (16Bit-registers)
;*****************************************************************************************************************
***
.def    TmpL    = r24                   ; Temporary register (low Byte)
.def    TmpH    = r25                   ; Temporary register (high Byte)
;       XL      = r26                   ; free
;       XH      = r27                   ; free
;       YL      = r28                   ; free
;       YH      = r29                   ; free
;       ZL      = r30                   ; used for table and indirect jump
;       ZH      = r31                   ; used for table and indirect jump
;*****************************************************************************************************************
***
;* Equations:
;*****************************************************************************************************************
***
.equ    E_LCD   = 3                     ; E-clock LCD = PortC Bit3
.equ    RW      = 4                     ; Read/Write LCD = PortC Bit4
.equ    RS      = 5                     ; RegisterSelect LCD = PortC Bit5
.equ    Busy    = 7                     ; BusyFlag    = PortD Bit7
.equ    F_Strt  = -430000               ; start frequency in kHz (fq = 10 MHz), must be a negative number!
.equ    F_Step  = 40960                 ; frequency step (= 2.50kHz * 2^14)
.equ    F_CPU   = 3370                  ; DB0CPU 438.425 MHz, PLL start value, F_Step = 2.50 kHz =8425/2,5
;.equ   F_CPU   = 570                   ; DB0MA 431.425 MHz, PLL start value, F_Step = 2.50 kHz =1425/2,5
;.equ   F_CPU   = 3610                  ; DB0MA 439.025 MHz, PLL start value, F_Step = 2.50 kHz =1425/2,5
;*****************************************************************************************************************
***
;* reset/interrupt-vectors:
;*****************************************************************************************************************
***
.cseg
        rjmp    Initial                 ; (2) after RESET to main program
        rjmp    ExtInt0                 ; (2) External Interrupt Request 0
        rjmp    ExtInt1                 ; (2) External Interrupt Request 1
        reti                            ; (4) Pin Change Interrupt Request 0
        reti                            ; (4) Pin Change Interrupt Request 1
        reti                            ; (4) Pin Change Interrupt Request 2
        reti                            ; (4) Watchdog Time-out Interrupt
        rjmp    T2Int1                  ; (4) Timer/Counter2 Compare Match A
        rjmp    T2Int2                  ; (4) Timer/Counter2 Compare Match B
        reti                            ; (4) Timer/Counter2 Overflow
        reti                            ; (4) Timer/Counter1 Capture Event
        reti                            ; (4) Timer/Counter1 Compare Match A
        reti                            ; (4) Timer/Counter1 Compare Match B
        reti                            ; (4) Timer/Counter1 Overflow
        reti                            ; (4) Timer/Counter0 Compare Match A
```

```
        reti                            ; (4) Timer/Counter0 Compare Match B
        reti                            ; (4) Timer/Counter0 Overflow
        reti                            ; (4) SPI Serial Transfer Complete
        reti                            ; (4) USART0, Rx Complete
        reti                            ; (4) USART0 Data register Empty
        reti                            ; (4) USART0, Tx Complete
        reti                            ; (4) ADC Conversion Complete
        reti                            ; (4) EEPROM Ready
        reti                            ; (4) 2-wire Serial Interface
        reti                            ; (4) Store Program Memory Read
;***************************************************************************************************
***
;* after reset: initialise stack,ports
;***************************************************************************************************
***
Initial:
        ldi     TmpL,Low(RAMEND)        ; (1)
        out     SPL,TmpL                ; (1)
        ldi     TmpL,High(RAMEND)       ; (1)
        out     SPH,TmpL                ; (1) stack initialised
        clr     Zero                    ; (1) always Zero
        out     PortB,Zero              ; (1) PortB = % 0000 0000
        out     PortC,Zero              ; (1) PortC = % 0000 0000
        out     PortD,Zero              ; (1) PortD = % 0000 0000
        ldi     TmpL,$2E                ; (1)
        out     DDRB,TmpL               ; (1) PortB direction = % xx10 1110
        ldi     TmpL,$3C                ; (1)
        out     DDRC,TmpL               ; (1) PortC direction = % xx11 1100
        ldi     TmpL,$F2                ; (1)
        out     DDRD,TmpL               ; (1) PortD direction = % 1111 0010
;***************************************************************************************************
***
;* initialize LC-Display
;***************************************************************************************************
***
Init2:
        ldi     TmpL,150                ; (1) 150*100usec = ca.15msec
        rcall   Wait100                 ; (3) wait for display ready
        ldi     TmpL,0b00110000         ; (1) 0011 to LCD
        rcall   LCDOut                  ; (3)
        ldi     TmpL,41                 ; (1) 41*100usec = ca.4,1msec
        rcall   Wait100                 ; (1)
        ldi     TmpL,0b00110000         ; (1) again 0011 to LCD
        rcall   LCDOut                  ; (3)
        ldi     TmpL,1                  ; (1) 100usec
        rcall   Wait100                 ; (3)
        ldi     TmpL,0b00110000         ; (1) 3 times 0011 to LCD
        rcall   LCDOut                  ; (3)
        ldi     TmpL,1                  ; (1) 100usec
        rcall   Wait100                 ; (3)
        ldi     TmpL,0b00100000         ; (1) 0010 = set 4 bit mode
        rcall   LCDOut                  ; (3)
        ldi     TmpL,1                  ; (1) 100usec
        rcall   Wait100                 ; (3)
        ldi     TmpL,$28                ; (1) 4 bit data, 2 lines, 5*7dots
        rcall   LCDCmd                  ; (3)
        ldi     TmpL,$0C                ; (1) display on, cursor off, not blinking
        rcall   LCDCmd                  ; (3)
        ldi     TmpL,$01                ; (1) clear display
        rcall   LCDCmd                  ; (3)
        ldi     TmpL,$06                ; (1) increment adress, no shift
        rcall   LCDCmd                  ; (3)
;***************************************************************************************************
***
;* write initial text on display
;***************************************************************************************************
***
UpLine:
        ldi     TmpL,$80                ; (1) set input cursor to
        rcall   LCDCmd                  ; (3) display-RAM, 1st byte upper line
        ldi     ZL,low(UpLTbl<<1)       ; (1) textstring "f_Rx= 438,425MHz"
        ldi     ZH,high(UpLTbl<<1)      ; (1) 16bit address mode
        ldi     Cnt,15                  ; (1) 16 characters
UpLin1:
        lpm     TmpL,Z+                 ; (3) fetch character from table
        rcall   LCDDat                  ; (3) to display, upper line
        dec     Cnt                     ; (1) 16 reached ?
        brge    UpLin1                  ; (2/1)
;***************************************************************************************************
***
; init ADC0
;***************************************************************************************************
***
        ldi     TmpL,$E0                ; (1) Ref=1.1Vint, left-adj, ADC0=input
        sts     ADMUX,TmpL              ; (1)
        ldi     TmpL,$EE                ; (1) Start,Auto, enable Int, Clk/64
        sts     ADCSRA,TmpL             ; (1)
        ldi     TmpL,$00                ; (1) free running
        sts     ADCSRB,TmpL             ; (1)
        ldi     TmpL,$01                ; (1) disable ADC0 dig.inp (optional)
        sts     DIDR0,TmpL              ; (1)
;***************************************************************************************************
```

```
***
;* set PLL start frequency
;*****************************************************************************************************************
***
        ldi     Frq0,low(F_CPU)         ; (1) receiver frequency for packet radio digipeater DB0CPU
        ldi     Frq1,high(F_CPU)        ; (1)
        mov     PLL_L,Frq0              ; (1)
        mov     PLL_H,Frq1              ; (1)
        clr     Frq2                    ; (1)
        clr     Frq1                    ; (1)
        clr     Frq0                    ; (1)
        clr     EncVal                  ; (1)
;*****************************************************************************************************************
***
;* create 5 special characters for bargraph
;*****************************************************************************************************************
***
        ldi     TmpL,$04                ; (1) cursor decrement, no shift
        rcall   LCDCmd                  ; (3)
        ldi     TmpL,$0C                ; (1) display on, cursor off, not blinking
        rcall   LCDCmd                  ; (3)
        ldi     TmpL,$6F                ; (1) charRAM, 6.byte of lower line 47d=$2F
        rcall   LCDCmd                  ; (3)
        ldi     Cnt,5                   ; (1) create special characters 05...00
        ldi     TmpH,$1F                ; (1) 5 black pixel hor.
BGrph1:
        clr     TmpL                    ; (1) lowest 2 lines white for cursor
        rcall   LCDDat                  ; (3)
        rcall   LCDDat                  ; (3)
        mov     TmpL,TmpH               ; (3)
        rcall   LCDDat                  ; (3) fill 6 char lines vert.
        mov     TmpL,TmpH               ; (3)
        rcall   LCDDat                  ; (3)
        mov     TmpL,TmpH               ; (3)
        rcall   LCDDat                  ; (3)
        mov     TmpL,TmpH               ; (3)
        rcall   LCDDat                  ; (3)
        mov     TmpL,TmpH               ; (3)
        rcall   LCDDat                  ; (3)
        mov     TmpL,TmpH               ; (3)
        rcall   LCDDat                  ; (3)
        lsl     TmpH                    ; (1) one more white pixel at right side of char
        dec     Cnt                     ; (1) next character
        brge    BGrph1                  ; (1)
        ldi     TmpL,$06                ; (1) cursor increment, no shift
        rcall   LCDCmd                  ; (3)
;*****************************************************************************************************************
***
; init 8 Bit Timer 2 for 19200 Hz interrupts rising and falling edges
;*****************************************************************************************************************
***
        ldi     TmpL,$02                ; (1)
        sts     TCCR2A,TmpL             ; (1) CTC mode, no OCR output
;       ldi     TmpL,$02                ; (1) (same number as before)
        sts     TCCR2B,TmpL             ; (1) CTC mode, dont force compare, clk= f_Q /8
        ldi     TmpL,92                 ; (1)
        sts     OCR2A,TmpL              ; (1) 14,31818MHz / 19200Hz = 8 * 93,217
        ldi     TmpL,47                 ; (1)
        sts     OCR2B,TmpL              ; (1) low to high at 47 (93,217/2=46,6)
        ldi     TmpL,$06                ; (1)
        sts     TIMSK2,TmpL             ; (1) enable compare interrupts A and B
/*
;*****************************************************************************************************************
***
; init 16 Bit Timer 1 for 4800 Hz output at OC1A pin 15 for tx-test or oscilloscope sync with baudrate 960
0Bd
;*****************************************************************************************************************
***
        ldi     TmpL,$C2                ; (1) fast PWM, Top=ICR, OC1A _/~ compare
        sts     TCCR1A,TmpL             ; (1)
        ldi     TmpL,$19                ; (1) fast PWM, Top=ICR, clk/1 (no prescaling)
        sts     TCCR1B,TmpL             ; (1)
        ldi     TmpL,high(2982)         ; (1) count from 0 to 2982 (14,31818MHz / 4800Hz = 2982.95)
        sts     ICR1H,TmpL              ; (1)
        ldi     TmpL,low(2982)          ; (1)
        sts     ICR1L,TmpL              ; (1)
        ldi     TmpL,high(1492)         ; (1) low to high at 1492 (=2982/2)
        sts     OCR1AH,TmpL             ; (1)
        ldi     TmpL,low(1492)          ; (1)
        sts     OCR1AL,TmpL             ; (1)
*/
;*****************************************************************************************************************
***
;* initialize SPI port for RFM12
;*****************************************************************************************************************
***
        sbi     PortB,2                 ; (2) SlaveSelect _/~
        ldi     TmpL,$50                ; (1) SPI-ControlRegister = % 0101 0000
        out     SPCR,TmpL               ; (1) (SPE,MSTR,SPR0)=1 (SPIE,DORD,CPOL,CPHA,SPR1,SPR0)=0
;*****************************************************************************************************************
***
; wait 250 msec for RFM12 to get ready for data
```

```
;*****************************************************************************************
***
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
        ldi     TmpL,255                ; (1) 255*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
;*****************************************************************************************
***
; send 12 configuration words to RFM12
;*****************************************************************************************
***
        ldi     ZL,low(RfmTbl<<1)       ; (1) init table pointer
        ldi     ZH,high(RfmTbl<<1)      ; (1) 16bit address mode
        ldi     Cnt,12                  ; (1) 12 words
RfmCnf:
        lpm     TmpL,Z+                 ; (3) fetch high byte from table
        cbi     PortB,2                 ; (2) SlaveSelect ~\_
        out     SPDR,TmpL               ; (2) send high byte, MSB first
Wait_H1:
        in      TmpL,SPSR               ; (2) wait for SPI ready
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_H1                 ; (2)
        lpm     TmpL,Z+                 ; (3) fetch low byte from table
        out     SPDR,TmpL               ; (2) send low byte, MSB first
Wait_L1:
        in      TmpL,SPSR               ; (2) wait for SPI ready
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_L1                 ; (2)
        sbi     PortB,2                 ; (2) SlaveSelect _/~
        dec     Cnt                     ; (1) 12 reached ?
        brne    RfmCnf                  ; (2/1)
;*****************************************************************************************
***
;* start rotary encoder interrupt
;*****************************************************************************************
***
        ldi     TmpL,$0F                ; (1) interrupt on rising edge of INT0 or INT1
        sts     EICRA,TmpL              ; (2) in External Interupt Control Register A
        ldi     TmpL,$03                ; (1) set bit 1 and 0
        out     EIMSK,TmpL              ; (1) in External Interupt MaSK register
        sei                             ; (1) enable interrupts
;*****************************************************************************************
***
; main loop
;*****************************************************************************************
***
Main:
;*****************************************************************************************
***
; wait 25 msec
;*****************************************************************************************
***
        ldi     TmpL,255                ; (1) 2*100usec = ca.25msec
        rcall   Wait100                 ; (3) wait
;*****************************************************************************************
***
; read 8 bit RSSI value from ADC, display as bargraph 0...80 units
;*****************************************************************************************
***
        ldi     TmpL,$06                ; (1) cursor increment, no shift
        rcall   LCDCmd                  ; (3)
        ldi     TmpL,$0C                ; (1) display on, cursor off, not blinking
        rcall   LCDCmd                  ; (3)
        ldi     TmpL,$C0                ; (1) display-RAM, 1. byte, lower line
        rcall   LCDCmd                  ; (3)
        ldi     Cnt,16                  ; (1) fill 16 display characters
        ldi     TmpL,$05                ; (1) set special character to 5 pixel
        lds     TmpH,ADCH               ; (1) fetch RSSI value from ADC0
BGrph2:
        subi    TmpH,16                 ; (1) subtract 16
        brcs    BGrph3                  ; (1) branch if < 0 ( TmpH was 0...15)
        ldi     TmpL,$05                ; (1) set special character to 5 pixel
        rcall   LCDDat                  ; (3) output 5 pixel char
        dec     Cnt                     ; (1) 16 characters reached ?
        brne    BGrph2                  ; (1) sonst zurück
        rjmp    BGrph5                  ; (1) ready
```

```
BGrph3:
        subi    TmpH,-17                ; (1) undo last subtraction and add 1 to round
        mov     TmpL,TmpH               ; (1) (* 5/16 =)
        lsl     TmpL                    ; (1)
        lsl     TmpL                    ; (1) * 4 = 0...60
        add     TmpL,TmpH               ; (1) + 1 = 0...75
        lsr     TmpL                    ; (1) /16 = 0...4
        lsr     TmpL                    ; (1)
        lsr     TmpL                    ; (1)
        lsr     TmpL                    ; (1)
BGrph4:
        andi    TmpL,$0F                ; (1) output one special char 0...4
        rcall   LCDDat                  ; (3)
        clr     TmpL                    ; (1) fill rest with blank char 0
        dec     Cnt                     ; (1)
        brge    BGrph4                  ; (1)
BGrph5:
/*
;***************************************************************************************************************
***
; send AFC strobe to RFM12
;***************************************************************************************************************
***
        cbi     PortB,2                 ; (2) SlaveSelect ~\_
        ldi     TmpL,$c4                ; (1) $c4d9 AFC Command high byte, enable _/~
        out     SPDR,TmpL               ; (1) output high byte to RFM12
Wait_H2:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_H2                 ; (1) no, wait
        in      TmpH,SPDR               ; (1) read status register high byte
        ldi     TmpL,$d1                ; (1) $c4d9 AFC Command low byte
        out     SPDR,TmpL               ; (1) output low byte to RFM12
Wait_L2:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_L2                 ; (1) no, wait
        in      TmpL,SPDR               ; (1) read status register low byte
        sbi     PortB,2                 ; (2) SlaveSelect _/~

;***************************************************************************************************************
***
; read status register from RFM12, wait for ATGL
;***************************************************************************************************************
***
        cbi     PortB,2                 ; (2) SlaveSelect ~\_
        clr     TmpL                    ; (1) $0000 = read status register
        out     SPDR,TmpL               ; (1) output high byte to RFM12
Wait_H3:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_H3                 ; (1) no, wait
        in      TmpH,SPDR               ; (1) read status register high byte
        clr     TmpL                    ; (1) $0000 = read status register
        out     SPDR,TmpL               ; (1) output low byte to RFM12
Wait_L3:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_L3                 ; (1) no, wait
        in      TmpL,SPDR               ; (1) read status register low byte
        sbi     PortB,2                 ; (2) SlaveSelect _/~

;***************************************************************************************************************
***
; clear AFC strobe in RFM12
;***************************************************************************************************************
***
        cbi     PortB,2                 ; (2) SlaveSelect ~\_
        ldi     TmpL,$c4                ; (1) $c4d1 AFC Command high byte, strobe ~\_
        out     SPDR,TmpL               ; (1) output high byte to RFM12
Wait_H4:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_H4                 ; (1) no, wait
        in      TmpH,SPDR               ; (1) read status register high byte
        ldi     TmpL,$d1                ; (1) $c4d1 AFC Command low byte
        out     SPDR,TmpL               ; (1) output low byte to RFM12
Wait_L4:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_L4                 ; (1) no, wait
        in      TmpL,SPDR               ; (1) read status register low byte
        sbi     PortB,2                 ; (2) SlaveSelect _/~

        sbrc    TmpL,5                  ; (2/1)
        cbi     PortB,1                 ; (2)
        sbrs    TmpL,5                  ; (2/1)
        sbi     PortB,1                 ; (2)
*/
;       rjmp    Main                    ;
```

```
;*****************************************************************************************************
***
; test rotary encoder, if unchanged return to main loop
;*****************************************************************************************************
***
        tst     EncVal                  ; (1) rotary encoder ?
        breq    Main                    ; (2/1) no change, back to main prg.
;*****************************************************************************************************
***
; rotary encoder changed: update PLL divider word, limit value to 96...3903
;*****************************************************************************************************
***
        cli                             ; (1)
        mov     TmpL,EncVal             ; (1) no interrupt allowed
        clr     EncVal                  ; (1) between these two lines
        sei                             ; (1)
        mov     Frq0,PLL_L              ; (1)
        mov     Frq1,PLL_H              ; (1)
        tst     TmpL                    ; (1) increase or decrease PLL frequency ?
        brmi    PllLo                   ; (1)
        add     Frq0,TmpL               ; (1)
        adc     Frq1,Zero               ; (1)
        cpi     TmpL,3                  ; (1) fast forward
        brlo    PllHi1                  ; (1)
        rol     TmpL                    ; (1)
        rol     TmpL                    ; (1)
        rol     TmpL                    ; (1)
        add     Frq0,TmpL               ; (1)
        adc     Frq1,Zero               ; (1)
PllHi1:
        cpi     Frq1,high(3903)         ; (1) high(3903)=$0F
        brlo    PllSet                  ; (1) if $00...$0E -> ok
        brne    SetMax                  ; (1) if $10...$FF -> max. reached
        cpi     Frq0,low(3903)          ; (1) if $0F test low byte, low(3903)=$3F
        brlo    PllSet                  ; (1)
SetMax:
        ldi     Frq1,high(3903)         ; (1) set PLL word to max.
        ldi     Frq0,low(3903)          ; (1)
        rjmp    PllSet                  ; (1)
PllLo:
        neg     TmpL                    ; (1) change from signed to absolute value
        sub     Frq0,TmpL               ; (1) unsigned minus unsigned
        sbc     Frq1,Zero               ; (1)
        cpi     TmpL,3                  ; (1) fast backward
        brlo    PllLo1                  ; (1)
        rol     TmpL                    ; (1)
        rol     TmpL                    ; (1)
        rol     TmpL                    ; (1)
        sub     Frq0,TmpL               ; (1)
        sbc     Frq1,Zero               ; (1)
PllLo1:
        tst     Frq1                    ; (1) (high(96)=$00)
        brmi    SetMin                  ; (1) if $FF...$80 -> min. reached
        brne    PllSet                  ; (1) if $01...$7F -> ok
        cpi     Frq0,low(96)            ; (1) if $00 test low byte, low(96)=$60
        brsh    PllSet                  ; (1)
SetMin:
        ldi     Frq1,high(96)           ; (1) set PLL word to min.
        ldi     Frq0,low(96)            ; (1)
PllSet:
        mov     PLL_L,Frq0              ; (1)
        mov     PLL_H,Frq1              ; (1)
;*****************************************************************************************************
***
; output new frequency control word to RFM12
;*****************************************************************************************************
***
        cbi     PortB,2                 ; (2) SlaveSelect ~\_
        mov     TmpL,PLL_H              ; (1) fetch PLL divider high byte
        ori     TmpL,$a0                ; (1) add frequency control command
        out     SPDR,TmpL               ; (1) output high byte to RFM12
Wait_H:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_H                  ; (1) no, wait
        out     SPDR,PLL_L              ; (1) output low byte to RFM12
Wait_L:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_L                  ; (1) no, wait
        sbi     PortB,2                 ; (2) SlaveSelect _/~
;*****************************************************************************************************
***
; compute frequency from PLL divider
;*****************************************************************************************************
***
        ldi     TmpL,low(F_Step)        ; (1) compute PLL-word * F_Step (= 2.50 kHz, fq=10 MHz, 433MHz-Ban
d)
        ldi     TmpH,high(F_Step)       ; (1) 16*16 multiply -> 32 bit result Frq3...Frq0
        mul     PLL_H,TmpH              ; (1) PLL_H * high(F_Step)
        movw    Frq3:Frq2, MultH:MultL  ; (1)
        mul     PLL_L,TmpL              ; (1) PLL_L * low(F_Step)
```

```
        movw    Frq1:Frq0, MultH:MultL  ; (1)
        mul     PLL_H,TmpL              ; (1) PLL_H * low(F_Step)
        add     Frq1,MultL              ; (1)
        adc     Frq2,MultH              ; (1)
        adc     Frq3,Zero               ; (1)
        mul     PLL_L,TmpH              ; (1) PLL_L * high(F_Step)
        add     Frq1,MultL              ; (1)
        adc     Frq2,MultH              ; (1)
        adc     Frq3,Zero               ; (1)
        lsl     Frq0                    ; (1)
        rol     Frq1                    ; (1) 2^14 -> 2^15
        rol     Frq2                    ; (1)
        rol     Frq3                    ; (1)
        lsl     Frq0                    ; (1)
        rol     Frq1                    ; (1) 2^15 -> 2^16
        rol     Frq2                    ; (1)
        rol     Frq3                    ; (1)
        mov     Frq0,Frq2               ; (1) :2^16
        mov     Frq1,Frq3               ; (1)
        clr     Frq2                    ; (1)
        clr     Frq3                    ; (1)
        subi    Frq0,byte1(F_Strt)      ; (1) add F_Strt (=430 MHz, fq=10 MHz, 433MHz-Band)
        sbci    Frq1,byte2(F_Strt)      ; (1) no "addi" available, for "subi" F_Strt must be a negative nu
mber !
        sbci    Frq2,byte3(F_Strt)      ; (1)
;***************************************************************************************************
***
; convert frequency from binary to bcd, output to display
;***************************************************************************************************
***
FrqLcd:
        ldi     TmpL,$86                ; (1) set cursor to display-RAM, 6th byte, upper line
        rcall   LCDCmd                  ; (3)
;
        ldi     TmpL,250                ; (1) TmpL=250 (-> 10 next line)
FrqLcd1:
        subi    TmpL,-16                ; (1) TmpL=TmpL+16 ->
        subi    Frq0,byte1(100000)      ; (1) increment high nibble as BCD digit
        sbci    Frq1,byte2(100000)      ; (1) Frq=Frq-100000
        sbci    Frq2,byte3(100000)      ; (1)
        brcc    FrqLcd1                 ; (1) until Frq is negative
FrqLcd2:
        dec     TmpL                    ; (1) TmpL=TmpL-1 ->
        subi    Frq0,byte1(-10000)      ; (1) decrement low nibble down from 10
        sbci    Frq1,byte2(-10000)      ; (1) Frq=Frq+10000
        sbci    Frq2,byte3(-10000)      ; (1)
        brcs    FrqLcd2                 ; (1) until Frq is positive
        mov     SavTmp,TmpL             ; (1) save low nibble
        swap    TmpL                    ; (1) high nibble
        andi    TmpL,$0F                ; (1)
        ori     TmpL,$30                ; (1) as ASCII char
        rcall   LCDDat                  ; (1) to LCD
        mov     TmpL,SavTmp             ; (1) fetch low nibble
        andi    TmpL,$0F                ; (1)
        ori     TmpL,$30                ; (1) as ASCII char
        rcall   LCDDat                  ; (1) to LCD
;
        ldi     TmpL,250                ; (1) TmpL=250 (-> 10 next line)
FrqLcd3:
        subi    TmpL,-16                ; (1) TmpL=TmpL+16 -> increment high nibble as BCD digit
        subi    Frq0,byte1(1000)        ; (1) Frq=Frq-1000
        sbci    Frq1,byte2(1000)        ; (1)
        brcc    FrqLcd3                 ; (1) until Frq is negative
FrqLcd4:
        dec     TmpL                    ; (1) TmpL=TmpL-1 -> decrement low nibble down from 10
        subi    Frq0,byte1(-100)        ; (1) Frq=Frq+100
        sbci    Frq1,byte2(-100)        ; (1) =0
        brcs    FrqLcd4                 ; (1) until Frq is positive
        mov     SavTmp,TmpL             ; (1) save low nibble
        swap    TmpL                    ; (1) high nibble
        andi    TmpL,$0F                ; (1) output low nibble
        ori     TmpL,$30                ; (1) as ASCII char
        rcall   LCDDat                  ; (1) to LCD
        ldi     TmpL,','                ; (1) decimal point
        rcall   LCDDat                  ; (1) to LCD
        mov     TmpL,SavTmp             ; (1) fetch low nibble
        andi    TmpL,$0F                ; (1) output low nibble
        ori     TmpL,$30                ; (1) as ASCII char
        rcall   LCDDat                  ; (1) to LCD
;
        ldi     TmpL,250                ; (1) TmpL=250 (-> 10 next line)
FrqLcd5:
        subi    TmpL,-16                ; (1) TmpL=TmpL+16 -> increment high nibble as BCD digit
        subi    Frq0,10                 ; (1) Frq=Frq-10
        brcc    FrqLcd5                 ; (1) until Frq is negative
        add     TmpL,Frq0               ; (1) TmpL=Frq0

        mov     SavTmp,TmpL             ; (1) save low nibble
        swap    TmpL                    ; (1) high nibble
        andi    TmpL,$0F                ; (1) output low nibble
        ori     TmpL,$30                ; (1) as ASCII char
        rcall   LCDDat                  ; (1) to LCD
```

```
        mov     TmpL,SavTmp             ; (1) fetch low nibble
        andi    TmpL,$0F                ; (1) output low nibble
        ori     TmpL,$30                ; (1) as ASCII char
        rcall   LCDDat                  ; (1) to LCD

        rjmp    Main                    ; (2)

;***************************************************************************************************************
***
;* subroutine LC-Display: output data byte in TmpL
;***************************************************************************************************************
***
LCDCmd:
        clt                             ; (1) command to LCD , T-Flag=RS=0
        rjmp    LCD1                    ; (2)
LCDDat:
        set                             ; (1) data to LCD, T-Flag=RS=1
LCD1:
        push    TmpL                    ; (2) wait for busy-flag
        ldi     TmpL,$02                ; (1) 4 data lines now inputs
        out     DDRD,TmpL               ; (1)
        sbi     PortC,RW                ; (2) R/W=H, read busy flag from LCD
        cbi     PortC,RS                ; (2) RS =L, instruction
LCD2:
        sbi     PortC,E_LCD             ; (2) Enable _/~
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1) Enable min. 450 ns = 6.5 / 14,31818 MHz
        nop                             ; (1)
        nop                             ; (1)
        in      TmpL,PinD               ; (1) read BusyFlag
        cbi     PortC,E_LCD             ; (2) Enable ~\_
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1) Enable min. 450 ns = 6.5 / 14,31818 MHz
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1)
        sbi     PortC,E_LCD             ; (2) Enable _/~ , dummy for 2nd nibble
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1) Enable min. 450 ns = 6.5 / 14,31818 MHz
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1)
        cbi     PortC,E_LCD             ; (2) Enable ~\_
        sbrc    TmpL,Busy               ; (2/1) Skip, if LCD ready (Busy-Flag=0)
        rjmp    LCD2                    ; (2)
        ldi     TmpL,$F2                ; (1) 4 data lines now outputs
        out     DDRD,TmpL               ; (2)
        cbi     PortC,RW                ; (2) RW=0, write to LCD
        brts    RShigh                  ; (2/1) instruction or data ?
        cbi     PortC,RS                ; (2) RS=1, instruction
        rjmp    RS_low                  ; (2/1)
RShigh:
        sbi     PortC,RS                ; (2) RS=0, data
RS_low:
        pop     TmpL                    ; (2) fetch databyte
        push    TmpL                    ; (2) save low nibble
        andi    TmpL,$F0                ; (1) high nibble only, RW=0
        out     PortD,TmpL              ; (2) output high nibble
        sbi     PortC,E_LCD             ; (2) Enable _/~ (min 450ns)
        pop     TmpL                    ; (2) fetch low nibble
        swap    TmpL                    ; (1)
        andi    TmpL,$F0                ; (1) high nibble only, RW=0
        cbi     PortC,E_LCD             ; (2) Enable ~\_
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1)
LCDout:
        out     PortD,TmpL              ; (2) output low nibble
        sbi     PortC,E_LCD             ; (2) Enable _/~
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1) Enable min. 450 ns = 6.5 / 14,31818 MHz
        nop                             ; (1)
        nop                             ; (1)
        nop                             ; (1)
        cbi     PortC,E_LCD             ; (2) Enable ~\_
        ret                             ; (4)
;***************************************************************************************************************
***
;* subroutine: wait ca. (TmpL*100)usec (f_clock=14.31818 MHz)
;***************************************************************************************************************
***
Wait100:
        clr     TmpH                    ; (1)
Wait1:
        nop                             ; (1) 1*NOP->  71,5 µsec inner loop
        nop                             ; (1) 2*NOP->  89,4 µsec inner loop
        nop                             ; (1) 3*NOP-> 107,3 µsec inner loop
```

```
        dec     TmpH                    ; (1)
        brne    Wait1                   ; (2/1)
        dec     TmpL                    ; (1)
        brne    Wait1                   ; (2) 2nd loop
        ret                             ; (4)
/*
;***************************************************************************************************************
***
;* AD2 als Mittenfrequenzanzeige auf Cursor ausgeben
;***************************************************************************************************************
***
MitFrq:
        ldi     TmpL,$C0                ; (1) Cursor auf untere Zeile
        add     TmpL,AD2                ; (1) 4Bit AD2 (Mittenanzeige) addieren
        rcall   SendCom                 ; (1) Befehl senden
        ldi     TmpL,$0E                ; (1) Display on, Cursor on, Blinken off
        rcall   SendCom                 ; (1) Befehl senden
;***************************************************************************************************************
***
;* Spektrum-Anzeige, Display-Initialisierung:
;***************************************************************************************************************
***
Spektr:
        ldi     TmpL,$06                ; (1) Cursor increment, no shift
        rcall   SendCom
        ldi     TmpL,$0C                ; (1) Display on, Cursor off, (nicht blinkend)
        rcall   SendCom
        ldi     TmpL,$C0                ; (1) DisplayRAM, 1. Byte untere Zeile
        rcall   SendCom
; hier Port-Schalterabfrage, SwpStp stellen, rjmp Spekt1 wenn Breitband-mode,
        ldi     ZH,high(SpkTbl)         ; (1) Textstring "2320xxxxxxxx2450"
        ldi     ZL,low(SpkTbl)          ; (1) x sind Sonderzeichen "0" bis "7"
        lsl     ZL                      ; (1) Z*2 wg. 16Bit-Speicher
        rol     ZH
        ldi     Cnt1,16                 ; (1) 16 Zeichen
        rjmp    AmaSpk
BrdSpk:
        ldi     ZH,high(BrdTbl)         ; (1) Textstring "2000xxxxxxxx3000"
        ldi     ZL,low(BrdTbl)          ; (1) x sind Sonderzeichen "0" bis "7"
        lsl     ZL                      ; (1) Z*2 wg. 16Bit-Speicher
        rol     ZH
        ldi     Cnt1,16                 ; (1) 16 Zeichen
AmaSpk:
        lpm
        mov     TmpL,LpmReg             ; (1) ins Display, untere Zeile
        rcall   SendDat
        adiw    ZL,1                    ; (1) eins weiterzählen
        dec     Cnt1                    ; (1) 16 erreicht ?
        brne    AmaSpk
;***************************************************************************************************************
***
;* Spektrum-Anzeige, Endlosschleife zu je 2 * 40 Messungen pro Dreieck:
;***************************************************************************************************************
***
BrdLoop:
        ldi     SwpOscL,$E8             ; (1) f-Start = 2012,500 MHz (-479,5 /0,125)
        ldi     SwpOscH,$2F             ; (1) = 12264 dez = 2FE8 hex
        ldi     TmpL,200                ; (1) Festfrequenz-Stufen, (80 pro Umdr.)
        mov     FrqStp,TmpL             ; (1) 80*125kHz = 10 MHz
        ldi     TmpL,200                ; (1) Sweep-Step ( 39 Stufen )
        mov     SwpStp,TmpL             ; (1) 200*125kHz = 25 MHz
        rjmp    SpkLp1
AmaLoop:
        ldi     SwpOscL,$91             ; (1) f-Start = 2321,625 MHz (-479,5 /0,125)
        ldi     SwpOscH,$39             ; (1) = 14737 dez = 3991 hex
        ldi     TmpL,8                  ; (1) Festfrequenz-Stufen, (80 pro Umdr.)
        mov     FrqStp,TmpL             ; (1) 8*125kHz = 1 MHz
        ldi     TmpL,26                 ; (1) Sweep-Step ( 39 Stufen )
        mov     SwpStp,TmpL             ; (1) 26*125kHz =3,25 MHz
SpkLp1:
        rcall   SetPLL                  ; (1) Startfrequenz Tuner-PLL einstellen
        rcall   DacInc                  ; (1) ersten Messwert verwerfen
        clr     DAC                     ; (1) Treppenspannung für ext. Oszilloskop
;***************************************************************************************************************
***
;* Spektrum-Anzeige, Schleife vorwärts = 5*8 Messungen
;***************************************************************************************************************
***
        clr     Cnt1                    ; (1) Schleifenanfang für 8 Sonderzeichen
SpkLp2:
        ldi     Cnt2,5                  ; (1) Schleifenanfang für 5 Messungen
        clr     ZH                      ; (1) highByte Registeradresse r01 = 0
        ldi     ZL,$01                  ; (1) Zeiger auf AD1 = Register r01
SpkLp3:
        add     SwpOscL,SwpStp          ; (1) um 26 * 0,125MHz = 3,25 MHz weiter
        brcc    SpkLp4
        inc     SwpOscH
SpkLp4:
        rcall   DacInc                  ; (1) Feldstärke messen, (vorheriger Wert)
        st      Z+,TmpL                 ; (1) 5 Messwerte in AD1..5 ablegen
        rcall   SetPLL                  ; (1) Tuner-PLL einstellen
        dec     Cnt2
```

```
        brne    SpkLp3                  ; (1) Schleifenende für 5 Messungen
        rcall   Char8x8                 ; (1) in 1 Sonderzeichen zu 8 Zeilen umrechnen
        inc     Cnt1                    ; (1) bis 5*8=40 Messungen erreicht sind
        cpi     Cnt1,8
        brne    SpkLp2                  ; (1) Schleifenende für 8 Sonderzeichen
        rcall   Chr8                    ; (1) Sonderzeichen ins Display,FixFreq-Update
;***************************************************************************************************
***
;* Spektrum-Anzeige, Schleife rückwärts = 5*8 Messungen
;***************************************************************************************************
***
        ldi     Cnt1,8                  ; (1) Schleifenanfang für 8 Sonderzeichen
SpkLp5:
        ldi     Cnt2,5                  ; (1) Schleifenanfang für 5 Messungen
        clr     ZH                      ; (1) highByte (r01) = 0
        ldi     ZL,$06                  ; (1) Zeiger auf (AD5 = Register r01)+1
SpkLp6:
        sub     SwpOscL,SwpStp          ; (1) um 26 * 0,125MHz = 3,25 MHz zurück
        brcc    SpkLp7
        dec     SwpOscH
SpkLp7:
        rcall   DacDec                  ; (1) Feldstärke messen, (vorheriger Wert)
        st      -Z,TmpL                 ; (1) 5 Messwerte in AD1..5 ablegen
        rcall   SetPLL                  ; (1) Tuner-PLL einstellen
        dec     Cnt2
        brne    SpkLp6                  ; (1) Schleifenende für 5 Messungen
        rcall   Char8x8                 ; (1) in 1 Sonderzeichen zu 8 Zeilen umrechnen
        dec     Cnt1                    ; (1) bis 5*8=40 Messungen erreicht sind
        brne    SpkLp5                  ; (1) Schleifenende für 8 Sonderzeichen
        rcall   Chr8                    ; (1) Sonderzeichen ins Display,FixFreq-Update
        rjmp    AmaLoop                 ; (1) Schleifenende einer Dreieckschwingung
;***************************************************************************************************
***
;* Unterprogramm zur Spektrum-Anzeige,  5 Messergebnisse AD1..AD5
;* in 8 Sonderzeichen - Zeilen umrechnen und in Puffer ablegen
;***************************************************************************************************
***
Char8x8:
        ldi     Cnt2,8                  ; (1) Schleifenanfang für 8 Sonderzeichen-Zeilen
        ldi     ZH,high(ChrBuf+8)
        ldi     ZL,low(ChrBuf+8)        ; (1) Pufferanfang, unterste Zeile
        mov     TmpL,Cnt1               ; (1) + 8*(Sonderzeichen Nr. 0-7)
        lsl     TmpL
        lsl     TmpL                    ; (1) ergibt Zeiger auf
        lsl     TmpL
        add     ZL,TmpL                 ; (1) 8 Byte - Sonderzeichenpuffer Nr.0-7
        brcc    Char8x81
        inc     ZH
Char8x81:
        clr     Line                    ; (1) Zeilenpuffer löschen = 8 * "clc"
        clc
        dec     AD1                     ; (1) 3 Bit zu 8 decodieren
        sbrs    AD1,7                   ; (1) schon negativ? dann skip next
        sec                             ; (1) Bildpunkt an
        rol     Line                    ; (1) von rechts nach links füllen
        dec     AD2
        sbrs    AD2,7                   ; (1) das ganze fünf mal
        sec
        rol     Line
        dec     AD3
        sbrs    AD3,7
        sec
        rol     Line
        dec     AD4
        sbrs    AD4,7
        sec
        rol     Line
        dec     AD5
        sbrs    AD5,7
        sec
        rol     Line                    ; (1) Sonderzeichen-Zeile fertig, jetzt
        st      -Z,line                 ; (1) von unten nach oben in den Puffer
        dec     Cnt2                    ; (1) nächste Zeile darüber
        brne    Char8x81                ; (1) Schleifenende für 8 Sonderzeichen-Zeilen
        ret
;***************************************************************************************************
***
;* Spektrum-Anzeige, Dreieckspannungsausgabe/Feldstärkemessung
;***************************************************************************************************
***
DacInc:
        ldi     TmpL,DACAdr             ; (1) Adr. des PCF8591, Richtung=Write
        rcall   ReStart                 ; (1) Startbed. + Slave-Adr. senden
        ldi     TmpL,$40                ; (1) Control-word des PCF8591
        rcall   WriteByte               ; (1) senden
        mov     TmpL,DAC                ; (1) ein Byte zum DAC
        inc     DAC                     ; (1) Treppenspannung um 6 höher
        inc     DAC                     ; (1) = 240 Stufen für 40 Messungen
        inc     DAC
        inc     DAC
        inc     DAC
        inc     DAC
```

```
        rjmp    Adc1
DacDec:
        ldi     TmpL,DACAdr             ; (1) Adr. des PCF8591, Richtung=Write
        rcall   ReStart                 ; (1) Startbed. + Slave-Adr. senden
        ldi     TmpL,$40                ; (1) Control-word des PCF8591
        rcall   WriteByte               ; (1) senden
        mov     TmpL,DAC                ; (1) ein Byte zum DAC
        dec     DAC                     ; (1) Treppenspannung um 6 tiefer
        dec     DAC                     ; (1) = 240 Stufen für 40 Messungen
        dec     DAC
        dec     DAC
        dec     DAC
        dec     DAC
Adc1:
        rcall   WriteByte               ; (1) senden
        rcall   Stopp
        ldi     TmpL,DACAdr+1           ; (1) Adr. des PCF8591, Richtung=Read
        rcall   Start                   ; (1) Startbed. + Slave-Adr. senden
        ser     Flg                     ; (1) "no Acknowledge" ausgeben
        rcall   ReadByte                ; (1) einen Messwert lesen
        swap    TmpL                    ; (1)
        andi    TmpL,$0F                ; (1) erst auf 4 Bit reduzieren
        inc     TmpL                    ; (1) aufrunden
        lsr     TmpL                    ; (1) von 4 auf 3 Bit reduzieren
        rcall   Stopp
        ret
;***************************************************************************************************************
***
;* Spektrum-Anzeige, Unterprogramm Display-Ausgabe
;***************************************************************************************************************
***
Chr8:
        ldi     TmpL,$40                ; (1) CharRAM, 1. Byte = RAM-Anfang
        rcall   SendCom
        ldi     ZH,high(ChrBuf)
        ldi     ZL,low(ChrBuf)
        ldi     Cnt1,64                 ; (1) 64 Bytes ins LCD-Character-RAM
Chr81:
        ld      TmpL,Z+
        rcall   SendDat
        dec     Cnt1
        brne    Chr81
        tst     Flag                    ; (1) Null = Änderung durch Interrupt
        brne    Cursor                  ; (1) sonst return
        rcall   FrqDisp                 ; (1) Frequenz aktualisieren
        mov     ZL,FixOscL              ; (1) Cursorberechnung
        mov     ZH,FixOscH              ; (1) PLL-Wert (2320-2450MHz) = $3991-$3D87
        subi    ZL,$8C
        sbci    ZH,$39                  ; (1) minus $398C -> 0005-1019 dez
        brcs    Curs1                   ; (1) unterhalb 2320MHz - Cursor links
        cpi     ZH,$04
        brge    Curs2                   ; (1) oberhalb 2450MHz - Cursor rechts
        lsl     ZL                      ; (1) oberstens Bit des lowbyte
        rol     ZH                      ; (1) und zwei unterste Bits highbyte
        andi    ZH,$07                  ; (1) ergeben drei Bit Cursor-Position
        mov     CurPos,ZH
        rjmp    Cursor
Curs1:
        clr     CurPos
        com     CurPos                  ; (1) $FF = links vom Frequenzfenster
        rjmp    Cursor
Curs2:
        ldi     TmpL,$08
        mov     CurPos,TmpL             ; (1) $08 = rechts vom Frequenzfenster
Cursor:
        ldi     TmpL,$84                ; (1) Cursor auf obere Zeile, 5.von links
        add     TmpL,CurPos             ; (1) 3Bit Cursorposition addieren
        rcall   SendCom
        ldi     TmpL,$0E                ; (1) Display on, Cursor on, Blinken off
        rcall   SendCom
        ret
;***************************************************************************************************************
***
;* Gibt <Cnt1> Zeichen an das Display aus
;***************************************************************************************************************
***
OutText:
        ld      TmpL,Z                  ; (1) Zeichen-Code nach TmpL laden
        rcall   SendDat                 ; (1) Zeichen senden, Autoinkrement der Adresse
        adiw    ZL,1                    ; (1) Z-Ptr inkrementieren
        dec     Cnt1                    ; (1) Zähler - 1
        brne    OutText                 ; (1) alle Zeichen an das LCD ausgeben
        ret
*/
;***************************************************************************************************************
***
;* Interrupt routine for rotary encoder
;***************************************************************************************************************
***
ExtInt0:
        in      SavSrg,SREG             ; (1) save status register
        push    TmpL                    ; (1) save TmpL
```

```
        lds     TmpL,EICRA              ; (1) Int0  _/~ or ~\_ ?
        sbrc    TmpL,0                  ; (1) skip next line if  ~\_
        rjmp    UpA                     ; (1) was _/~ so jump to UpA
        ori     TmpL,$01                ; (1) EICRA-bit0 = 1
        sts     EICRA,TmpL              ; (1) Int0 next change will be _/~
        sbic    PinD,3                  ; (1) skip next line if channel B=0
        rjmp    DecRot                  ; (1) if B=1 and A=~\_ -> decrement
        rjmp    IncRot                  ; (1) if B=0 and A=~\_ -> increment
UpA:
        andi    TmpL,$FE                ; (1) EICRA-bit0 = 0
        sts     EICRA,TmpL              ; (1) Int0 next change will be ~\_
        sbic    PinD,3                  ; (1) skip next line if channel B=0
        rjmp    IncRot                  ; (1) if B=1 and A=_/~ -> increment
        rjmp    DecRot                  ; (1) if B=0 and A=_/~ -> decrement
ExtInt1:
        in      SavSrg,SREG             ; (1) save status register
        push    TmpL                    ; (1) save TmpL
        lds     TmpL,EICRA              ; (1) Int1  _/~ or ~\_ ?
        sbrc    TmpL,2                  ; (1) skip next line when ~\_
        rjmp    UpB                     ; (1) was _/~ so jump to UpB
        ori     TmpL,$04                ; (1) EICRA-bit2 = 1
        sts     EICRA,TmpL              ; (1) Int1 next change will be _/~
        sbic    PinD,2                  ; (1) skip next line if channel A=0
        rjmp    IncRot                  ; (1) if A=1 and B=~\_ -> increment
        rjmp    DecRot                  ; (1) if A=0 and B=~\_ -> decrement
UpB:
        andi    TmpL,$FB                ; (1) EICRA-bit2 = 0
        sts     EICRA,TmpL              ; (1) Int1 next change will be ~\_
        sbic    PinD,2                  ; (1) skip next line if channel A=0
        rjmp    DecRot                  ; (1) if A=1 and B=_/~ -> decrement
        rjmp    IncRot                  ; (1) if A=0 and B=_/~ -> increment
IncRot:
        mov     TmpL,EncVal             ; (1) max limit = +127 reached ?
        cpi     TmpL,$7F                ; (1)
        breq    Exit                    ; (1) yes, so don't increment
        inc     EncVal                  ; (1) no, increment
        rjmp    Exit                    ; (1)
DecRot:
        mov     TmpL,EncVal             ; (1) min limit = -128 reached ?
        cpi     TmpL,$80                ; (1)
        breq    Exit                    ; (1) yes, so don't decrement
        dec     EncVal                  ; (1) no, decrement
Exit:
        pop     TmpL                    ; (1) restore TmpL
        out     SREG,SavSrg             ; (1) restore status register
        reti                            ; (?)
;***************************************************************************************************
***
;* Interrupt routine Timer2 free running 9600 Hz
;***************************************************************************************************
***
T2Int1:
        in      SavSrg2,SREG            ; (1) save status register
        push    TmpL                    ; (1) save TmpL
;       cbi     PortB,1                 ; (2) test only
        cbi     PortB,2                 ; (2) SlaveSelect ~\_
        ldi     TmpL,$80                ; (1) $8014 Configuration Setting Command high byte
        out     SPDR,TmpL               ; (1) output high byte to RFM12
Wait_H2:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_H2                 ; (1) no, wait
        ldi     TmpL,$14                ; (1) $8014 Configuration Setting Command low byte
        out     SPDR,TmpL               ; (1) output low byte to RFM12
Wait_L2:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_L2                 ; (1) no, wait
        sbi     PortB,2                 ; (2) SlaveSelect _/~
        pop     TmpL                    ; (1) restore TmpL
        out     SREG,SavSrg2            ; (1) restore status register
        reti                            ; (?)

T2Int2:
        in      SavSrg2,SREG            ; (1) save status register
        push    TmpL                    ; (1) save TmpL
;       sbi     PortB,1                 ; (2) test only
        cbi     PortB,2                 ; (2) SlaveSelect ~\_
        ldi     TmpL,$80                ; (1) $8018 Configuration Setting Command high byte
        out     SPDR,TmpL               ; (1) output high byte to RFM12
Wait_H3:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_H3                 ; (1) no, wait
        ldi     TmpL,$18                ; (1) $8018 Configuration Setting Command low byte
        out     SPDR,TmpL               ; (1) output low byte to RFM12
Wait_L3:
        in      TmpL,SPSR               ; (1) SPI ready ?
        sbrs    TmpL,SPIF               ; (2/1)
        rjmp    Wait_L3                 ; (1) no, wait
        sbi     PortB,2                 ; (2) SlaveSelect _/~
        pop     TmpL                    ; (1) restore TmpL
```

```
        out     SREG,SavSrg2            ; (1) restore status register
        reti                            ; (?)

;***************************************************************************************************************
***
;* Text-Table upper line
;***************************************************************************************************************
***
UpLTbl:
.db     "f_Rx= 438,425MHz"
;***************************************************************************************************************
***
;* RFM12 Initialization Table - low byte first !
;***************************************************************************************************************
***
RfmTbl:
.dw     $1480   ; $8014 Configuration Setting Command
;.dw    $3882   ; $8238 Power Management Command transmit
.dw     $d882   ; $823d Power Management Command receive
.dw     $2aad   ; $ad2a Frequency Setting Command
.dw     $23c6   ; $c623 Data Rate Command
;.dw    $c094   ; $94c0 Receiver Control Command VDI fast
.dw     $c097   ; $97c0 Receiver Control Command VDI always on
.dw     $fcc2   ; $c2fc Data Filter Command
.dw     $85ca   ; $ca85 FIFO and Reset Mode Command
;.dw    $30c4   ; $c430 AFC Command
.dw     $d1c4   ; $c4d1 AFC Command
.dw     $0099   ; $9900 TX Configuration Control Command
.dw     $00e0   ; $e000 Wake-Up Timer Command
.dw     $00c8   ; $c800 Low Duty-Cycle Command (default)
.dw     $00c0   ; $c000 Low Battery Detector and Microcontroller Clock Divider Command (default)

;.dw    $b800   ;       (Transmitter Register Write Command)
;.dw    $xxxx   ;       (Receiver FIFO Read Command) read only
;.dw    $0000   ;       (Status Read Command) read only
;***************************************************************************************************************
***
```