```
;*****************************************************************************************
; Radio mit dem RDA5807 Modul und ATtiny44, Platine TN20181024, Takt 1MHz intern, default
; Fusebytes 62/DF, Beschaltung: PA7=Btn left, PA6=Btn right (jeweils nach Masse),
; PA0 - PA3: LEDs zur Statusanzeige, PB0=SDA, PB1=SCL = I2C Interface per "Bitbanging"
; Bedienung: Kurze Tastendrücke => Senderwechsel, lange Tastendrücke => Lautstärke ändern
;
; 07/11/2018, erstellt mit AVRStudio 4.17, (c) T. Neveling
;*****************************************************************************************
;
.include "tn44Adef.inc"          ; ATtiny44 Definitionsdatei einbinden
;
.def    IntReg      = r14        ; Zur SREG-Sicherung während des Timer0 Interrupts
.def    NULL        = r15        ; immer = 0
.def    temp1       = r16        ; Universalvariablen
.def    temp2       = r17
.def    temp3       = r18
.def    fm Index    = r19        ; Nummer des eingestellten Senders
.def    Keys        = r20        ; Zur Tasten Entprellung
.def    Counter     = r21        ; Steuert die LED Ausgabe (Sender bzw. Lautstärke)
;
; RDA5807 Steuerbytes
.def    fm Hi       = r2         ; Empfangsfrequenz   => R3hi/lo
.def    fm Lo       = r3
.def    Volume      = r22        ; Lautstärke (0..15) => R5lo (Bits 0..3)
;
.cseg                            ; Codesegment
;
; Interrupt Vektoren
.org 0          rjmp start       ; Reset Handler, Startadresse 0
.org OVF0Addr   rjmp t0int       ; 16,384ms Overflow Interrupt: Tastenabfrage
;
start:
    ldi temp1, LOW(RAMEND)       ; Stackpointer setzen
    out SPL, temp1
    ldi temp1, HIGH(RAMEND)
    out SPH, temp1
;
    ldi temp1, 0x0F              ; Ports initialisieren
    out DDRA, temp1              ; PA0-PA3 sind Ausgänge für die LEDs
    ldi temp1, 0xFF              ; Pullup Widerstände an PA6 und PA7 (Tasten)
    out PORTA, temp1             ; Alle 4 LED zunächst einschalten
    rcall i2c_init               ; I2C Portsignale PB0/PB1 initialisieren
;
    clr Keys                     ; Variablen initialisieren
    clr NULL                     ; immer = 0
    ldi Volume, 0x0F             ; Maximale Lautstärke einstellen (Range 0 - 15)
;
    ldi temp1, 170               ; RDA5807 Hardware Reset abwarten! (680ms)
    rcall delay_4ms
;
    ldi temp2, Last FM + 4       ; Startfrequenz aus EEPROM lesen: temp2 = Adresse
    rcall EE Read                ; Byte nach fm index lesen
    rcall fm_get_channel         ; RDA5807 konfigurieren & Startsender einstellen
;
; Timer0: 16,384ms Overflow Interrupt zur Tastenenabfrage und Entprellung
    ldi temp1, (1<<CS01) | (1<<CS00)
    out tccr0b, temp1            ; Prescaler 1/64 => 64 * 256 / 1MHz = 16,384ms
    ldi temp1, 1<<TOIE0
    out timsk0, temp1            ; Timer0 Interrupt freigeben
    sei                          ; globale Interrupt Freigabe
;
; Leere Main Schleife
main: rjmp main
;
;*************************************************************
;* Timer Interrupt zur Tasten Auswertung, Iteration 16,384ms *
;*************************************************************
t0int:
    push temp1
    in intreg, sreg
;
    cpi Keys, 0
    brne KeysNotZero
                                 ; beim letzten Mal war keine Taste gedrückt
    sbis PINA, 6                 ; Tasten neu abfragen
    rcall btn left               ; btn_left wurde gedrückt
    sbis PINA, 7
    rcall btn_right              ; btn_right wurde gedrückt
```

```
    rjmp t0intx
;
KeysNotZero:                    ; Keys waren beim letzten Mal <> 0: jetzt neu prüfen
    clr Keys
    sbis PINA, 6
    sbr Keys, 1
    sbis PINA, 7                ; wenn keine Taste mehr gedrückt ist: Keys = 0
    sbr Keys, 1
;
t0intx:
    rcall LED Anzeige
    out sreg, intreg
    pop temp1
    reti
;
LED Anzeige:                    ; Lautstärke und Senderindex im Wechsel anzeigen
    dec Counter
    cpi Counter, 170            ; Senderindex für ca. 3s zeigen
    brlo LED Anzeige 1
    mov temp1, Volume           ; Lautstärke für ca. 1s zeigen
    rjmp LED Anzeige_2
LED Anzeige 1:
    mov temp1, fm_Index
LED Anzeige 2:
    ori temp1, 0b11110000       ; Pull-up Widerstände an den Tasten PA6 & PA7 beibehalten
    out PORTA, temp1
    ret
;
btn left:                       ; Taste links
    ldi temp1, 100
    rcall delay 4ms             ; 400ms
    sbic PINA, 6
    rjmp btn left 1             ; kurzer Tastendruck
    sbr Keys, 1                 ; Taste ist weiterhin gedrückt
    rcall fm vol decr           ; Lautstärke verringern
    rjmp btn_left_2
btn left 1:
    rcall fm_prev               ; vorheriger Sender aus der Tabelle
btn left_2:
    ret
;
btn right:                      ; Taste rechts
    ldi temp1, 100
    rcall delay 4ms             ; 400ms
    sbic PINA, 7
    rjmp btn right 1            ; kurzer Tastendruck
    sbr Keys, 1                 ; Taste ist weiterhin gedrückt
    rcall fm vol incr           ; Lautstärke erhöhen
    rjmp btn_right_2
btn right 1:
    rcall fm_next               ; nächster Sender aus der Tabelle
btn right_2:
    ret
;
; Frequenztabelle für Festfrequenzen (Raum Bremen, Stand 10/2018)
; Berechnung: fhi/flo = (Frequenz * 10 - 870) * 64 + 16
; Beispiel 95,0 MHz: fhi/flo = (950 - 870) * 64 + 16 = 5136 = 0x1410
fm Stations:
;       High  Low                                           Index
    .db 0x03, 0x50              ; Bremen 2       88,3MHz    0
    .db 0x07, 0x10              ; Energy Bremen  89,8MHz    1
    .db 0x0A, 0x50              ; NDR1           91,1MHz    2
    .db 0x11, 0x10              ; Bremen 1       93,8MHz    3
    .db 0x12, 0x90              ; NDR Kultur     94,4MHz    4
    .db 0x14, 0x10              ; NDR Info       95,0MHz    5
    .db 0x20, 0x10              ; NDR2           99,8MHz    6
    .db 0x21, 0x50              ; DKultur       100,3MHz    7
    .db 0x23, 0x90              ; Bremen 4      101,2MHz    8
    .db 0x26, 0x50              ; ffn           102,3MHz    9
    .db 0x2E, 0xD0              ; Antenne NDS   105,7MHz    A
    .db 0x32, 0x50              ; DLF           107,1MHz    B
;
fm get channel:
    ldi ZL, LOW(fm Stations*2)  ; Zeiger auf die Frequenztabelle
    ldi ZH, HIGH(fm Stations*2)
    mov temp1, fm Index         ; Nr des einzustellenden Senders
    lsl temp1                   ; *2, da 2 Byte je Eintrag
    add ZL, temp1
```

```
    adc ZH, NULL                ; eigentlich unnötig, da Addr(fm Stations < 256 Byte)
    lpm fm Hi, Z+               ; Bytes in Frequenzregister kopieren
    lpm fm lo, Z
    rcall fm command            ; gewählte Empfangsfrequenz einstellen
    ldi temp2, Last_FM + 4      ; Sender Index im EEPROM speichern
    rcall EE_Write
    ret
;
fm prev:                        ; Wechsel zwischen gespeicherten Festfrequenzen
    dec fm Index
    cpi fm Index, 255
    brne fm prev 01
    ldi fm Index, 10            ; wieder oben in Tabelle anfangen
fm prev 01:
    rcall fm_get_channel
    ret
;
fm next:                        ; Wechsel zwischen gespeicherten Festfrequenzen
    inc fm Index
    cpi fm Index, 12            ; 12 Einträge sind vorhanden, Index = 0..11
    brne fm next 01
    clr fm Index                ; wieder am Tabellenanfang beginnen
fm next 01:
    rjmp fm_prev_01             ; Der Rest ist wie oben
;
fm vol incr:                    ; Lautstärke erhöhen
    inc Volume
    cpi Volume, 16
    brne fm vol incr 01
    dec Volume                  ; max 15
fm vol incr 01:
    clr fm lo                   ; nicht neu tunen (Bit 4 ist gelöscht)
    rcall fm_command            ; RDA5807 Update
    ret
;
fm vol decr:                    ; Lautstärke verringern
    dec Volume
    cpi Volume, 255
    brne fm vol decr 01
    inc Volume                  ; min 0
fm vol decr 01:
    rjmp fm_vol_incr_01         ; Der Rest ist wie oben
;
fm command:                     ; Das Beschreiben der 16 Bit Register erfolgt ab R02
    rcall i2c start             ; aufsteigend, immer erst High Byte, dann Low Byte
    ldi temp1, (0x10 << 1)      ; I2C Adresse für sequenziellen Zugriff (schreibend)
    rcall i2c write
    ldi temp1, 0b11010000       ; Mute OFF, NORMAL Out, Stereo, Seek OFF, Bassboost
    rcall i2c write             ; => R02 - High
    ldi temp1, 0b00000101       ; Enable, no reset, New Method, RDS OFF, 32,768kHz, wrap seek
    rcall i2c write             ; => R02 - Low
    mov temp1, fm Hi            ; Startfrequenz: Obere 8 Bit
    rcall i2c write             ; => R03 - High
    mov temp1, fm Lo            ; Startfrequenz: Untere 2 Bit, Band 0, 100kHz Raster, Tunebit
    rcall i2c write             ; => R03 - Low
    ldi temp1, 0x04             ; Bei R04 sind im Datenblatt nicht alle Bits beschrieben!
    rcall i2c write             ; => R04 - High
    ldi temp1, 0x00
    rcall i2c write             ; => R04 - Low
    ldi temp1, 0x88             ; Interrupt = default, Seek Threshold = 8
    rcall i2c write             ; => R05 - High
    ldi temp1, 0x80             ; Bit 7 ist im Datenblatt nicht beschrieben.
    or temp1, Volume            ; untere 4 Bits für die Lautstärke
    rcall i2c write             ; => R05 - Low
    rcall i2c_stop              ; I2C Bus wieder freigeben
    ret
;
i2c init:                       ; ### Basisfunktionen zur I2C Ansteuerung ###
    cbi PORTB, 0
    cbi PORTB, 1
    rcall scl hi
    rcall sda_hi
    ret
;
i2c start:
    rcall sda lo
    rcall scl_lo
    ret
```

```
;
i2c stop:
    rcall scl hi
    rcall sda_hi
    ret
;
i2c write:                      ; Byte aus temp1 auf I2C ausgeben
    ldi temp2, 8                ; 8 Bit seriell schreiben
i2c wr 0:
    sbrc temp1, 7               ; oberes Bit zuerst
    rcall sda hi
    sbrs temp1, 7
    rcall sda lo
    rcall scl hi
    rcall scl_lo
    lsl temp1
    dec temp2
    brne i2c wr 0
    rcall sda hi
    rcall scl hi
    sbis PINB, 0
    ldi temp1, 1
    rcall scl_lo
    ret
;
i2c delay:
    push temp1                  ; wird in i2c_write weiter benutzt
    ldi temp1, 7
i2c delay 0:
    dec temp1
    brne i2c delay_0
    pop temp1
    ret
;
scl lo:
    sbi DDRB, 1
    rcall i2c_delay
    ret
;
scl hi:
    cbi DDRB, 1
    rcall i2c_delay
    ret
;
sda lo:
    sbi DDRB, 0
    rcall i2c_delay
    ret
;
sda hi:
    cbi DDRB, 0
    rcall i2c_delay
    ret
;
delay 4ms:                      ; temp1 = Multiplikator wird vorher gesetzt
delay 0:                        ; Konstanten hier für 1MHz µC Takt
    ldi temp2, 50
delay 1:
    ldi temp3, 26
delay 2:
    dec temp3
    brne delay_2
    dec temp2
    brne delay_1
    dec temp1
    brne delay_0
    ret
;
;       ### EEPROM lesen und schreiben (hierzu ggf. Interrupts sperren) ###
;
EE Read:                        ; 1 Byte aus Adresse temp2 nach fm index einlesen
    sbic EECR, EEPE             ; prüfe ob vorheriger Schreibzugriff beendet ist
    rjmp EE Read                ; wenn nicht, weiter prüfen
    out EEARH, NULL             ; Adresse in EEPROM Adressregister laden, High hier immer = 0
    out EEARL, temp2            ; Low Adressbyte gemäß Vorgabe
    sbi EECR, EERE              ; Lesen aktivieren EEPROM-Read-Bit im EEPROM Controlreg.
    in fm_index, EEDR           ; Daten aus EEPROM Dataregister in µC Register kopieren
    ret
```

```
;
EE Write:                       ; 1 Byte aus fm index nach Adresse temp2 schreiben
    sbic EECR, EEPE             ; prüfe ob der letzte Schreibzugriff beendet ist
    rjmp EE Write               ; wenn nicht, weiter prüfen
    ldi temp3, (0<<EEPM1)|(0<<EEPM0)    ; Modus: Löschen und (über)schreiben
    out EECR, temp3
    out EEARH, NULL             ; Adresse in EEPROM Adressregister laden, High hier immer = 0
    out EEARL, temp2            ; Low Adressbyte gemäß Vorgabe
    out EEDR, fm index          ; Datenbyte in EEPROM Datenregister schreiben
    sbi EECR, EEMPE             ; Schreiben vorbereiten
    sbi EECR, EEPE              ; Und los !
    ret
;
.eseg                           ; Daten im EEPROM definieren
Last_FM:    .db 0, 0, 0, 0, 0   ; EEPROM Speicherung von fm_index im 5. Byte
```