```
program Attenuator;
{
Steuerung des Weinschel Attenuators 3200-1 von 0dB bis 127dB.

Autor:                Hans-Peter Prast, DL2KHP
Datum:                15.10.2014
Prozessor:            ATMega16
Clock:                8 MHz, intern
Version:              1.0
Letzte Änderung:      15.10.2014   Programmdatei angelegt angepasst
                      18.10.2014   Programmcode erstellt
                      19.10.2014   Programmcode fertiggestellt
                      05.11.2014   Debugging Programmcode
                      08.11.2014   Debugging Programmcode
                      10.11.2014   Handling für "Offset" geändert


-----------------------------------------------------------------
Der Dämpfungswerte des Weinschel Attenuators 3200-1 werden durch die
Kombination von 8 Einzelwerten in 1dB-Stufen zwischen 0 und 127dB
eingestellt. Das Kombinationsmuster der einzelnen Stufen ist für jedes
einzelne dB in einer Tabelle im EEPROM abgelegt. Das EEPROM wird dann mit
dem dB-Wert als Adresse ausgelesen und der Inhalt des Speicherplatzes an den
Attenuator ausgegeben.
Wenn die Taste des Drehgebers gedückt wird, wird der eingestellte Wert als
Offset übernommen. Die Anzeige zeigt dann anstelle von "Att = xx dB"
"Out = xx dBm" an, der Anzeigewert wird "0" und der eingestellte Wert wird
jetzt als Offset angezeigt. Jetzt kann auch ein Wert größer "0", bis zum
eingestellten Offset, angezeigt werden. Das ist als Hilfe gedacht, wenn Geräte
z.B. bis 20dBm kalibriert werden sollen. Dann kann ein Wert von 20 dBm angelegt
werden. Wird dann bei einer Einstellung "Att = -20dB" die Taste am Drehgeber
betätigt, wird der Wert übernommen und "Out = 0 dBm" angezeigt. Der Anzeige
entspricht jetzt dem tatsächlichen Ausgangswert. Jetzt kann dann auch bis zum
eingestellten Offset, über "0" hinaus, bis 20 dBm eine Einstellung vorgenommen
werden. Ein erneuter Druck auf den Drehgeber schaltet den Offset wieder ab.

Wenn beim Einschalten die Taste Down1x festfehalten wird, kann die Helligkeit
des Displays mit den Tasten Up10x und Down10x eingestellt werden. Ein Druck auf
die Taste Up1x beendet die Einstellung. Der Helligkeitswert wird abgespeichert
und beim Einschalten wieder eingestellt.


Belegung EEPROM
0x000 - 0x07F  Schaltmuster für die Dämpfungswerte
0x1F0          PWM-Wert für die Hintergrundbeleuchtung
0x1F1 - 0x1FF  frei

}

// Vereinbarungsteil ----------------------------------------------
const
// LCD Parameter --------------------------------------------------
   k_zeile1 = ' * Attenuator * '; // Startmeldung
   k_zeile2 = ' * Vers.: 1.0 * '; // dito
   b_rs = 6;                      // RS-Bit im Steuerport B
   b_en = 7;                      // EN-Bit im Steuerport B
// Ports ----------------------------------------------------------
   k_ddr_a = 0xff;                // Display-Daten, alles Output
   k_ddr_b = %11000000;           // B0 - B5 Eingang, B6 = RS, B7 = EN
   k_port_b = %00111111;          // Pull-Up-Widerst. Port B
   k_ddr_d = %10000000;           // PD0 = B. PD2 = A, PD3 =  PD7 = PWM Displ.
   k_port_d = %01111010;          // Pull-Up-Widerstände Port D
```

```
63:    k_ddr_c = 0xff;                 // Port C Ansteuerung Dämpfungsglied
64: // Taster, Digital-Bits ------------------------------------------------
65:    b_sw1 = 0;                      // Eingang Taster SW1, up 1x     port B
66:    b_sw2 = 1;                      // Eingang Taster SW2, down 1x   port B
67:    b_sw3 = 2;                      // Eingang Taster SW3, up 10x    port B
68:    b_sw4 = 3;                       // Eingang Taster SW4, down 10x  port B
69:    b_pwm = 7;                      // PWM Ausgang LCD-Beleuchtung   port D
70:    b_inc_a = 2;                    // Drehgeber Ausgang A    port D
71:    b_inc_b = 0;                    // Drehgeber Ausgang B    port D
72:    b_push = 3;                     // Drehgeber Taster       port D
73: // Timer 2, PWM Steuerung ----------------------------------------------
74:    k_tccr2 = %01101100;            // Fast PWM, OC-Output, 1:64
75:    k_pwm = 200;                    // Vorgabe PWM-Steuerung Hell
76:    k_eep_pwm = 0x1f0;              // Adresse EEPROM Speicherplatz PWM-Wert
77: // User Flag Register --------------------------------------------------
78:    b_db_neu = 1;                   // dBm-Wert wurde geändert
79:    b_anz_neu = 2;                  // LCD-Anzeige aktualisieren
80:    b_taste_up1 = 3;                // Taste Up 1x wurde betätigt
81:    b_taste_down1 = 4;              // Taste Down 1x wurde betätigt
82:    b_taste_up2 = 5;                // Taste Up 10x wurde betätigt
83:    b_taste_down2 = 6 ;             // Taste Down 10x wurde betätigt
84:    b_taste_push = 7;               // Taster Drehgeber
85:    b_offset = 8;                   // Offset aktiv
86:
87: // Variablen -----------------------------------------------------------
88: var
89:    v_stat_int1,                    // für Statusabfrage im Interrupt 1
90:    v_stat_ovint,                   // für Statusabfrage im Overflow-Interrupt
91:    v_stat_net : boolean;           // für Statusabfrage in der Steuerung
92:
93:    v_pwm,                          // PWM-Wert für die Hintergrundbeleuchtung
94:    v_pwm_zwi,                      // Zwischenspeicher für PWM-Wert
95:    v_db,                           // Enthält den Wert der geschalteten dB
96:    v_db_off,                       // Enthält den Offset für die Anzeige
97:    v_sw,                           // Enthält den Schalterzustand
98:    v_temp   : byte;                // Global
99:
100:   v_db_anz : integer;             // Variable für den Anzeigewert
101:
102:   v_ufr    : array [1..10] of boolean; // User Flag Register
103:   v_displ  : array [1..23] of char; // 1. Vorz., 2. Zehner, 3. Einer, Komma
104:   v_zeile : string[16];
105:
106: // LCD-Variablen für die Library ---------------------------------------
107: var LCD_RS : sbit at PORTB6_bit;
108: var LCD_EN : sbit at PORTB7_bit;
109: var LCD_D4 : sbit at PORTA4_bit;
110: var LCD_D5 : sbit at PORTA5_bit;
111: var LCD_D6 : sbit at PORTA6_bit;
112: var LCD_D7 : sbit at PORTA7_bit;
113: var LCD_RS_Direction : sbit at DDB6_bit;
114: var LCD_EN_Direction : sbit at DDB7_bit;
115: var LCD_D4_Direction : sbit at DDA4_bit;
116: var LCD_D5_Direction : sbit at DDA5_bit;
117: var LCD_D6_Direction : sbit at DDA6_bit;
118: var LCD_D7_Direction : sbit at DDA7_bit;
119:
120:
121: // Ende Vereinbarungsteil ----------------------------------------------
122:
123:
124: // Interrupt Service Routinen ------------------------------------------
```

```
125:
126: // Drehgeber betätigt ---------------------------------------------
127: procedure int0int(); org 0x002;  // Drehgeber wurde betätigt
128: begin
129:
130:   if (pind.b_inc_a = pind.b_inc_b) then
131:       begin
132:         if v_db <= 126 then
133:           begin
134:             v_db := v_db + 1;          // Wert für die Dämpfung incrementieren
135:           end;
136:       end;
137:    if (pind.b_inc_a <> pind.b_inc_b) then
138:        begin
139:          if v_db > 0  then
140:            begin
141:              v_db := v_db - 1;          // Wert für die Dämpfung decrementieren
142:            end;
143:        end;
144:    v_ufr[b_db_neu]:= 1;
145: end;
146:
147:
148: // Taster Drehgeber betätigt -------------------------------------
149: // Bei jeder Betätigung wird das Tasten Flag geändert
150: procedure int1int(); org 0x004;  // Taster Drehgeber wurde betätigt
151: begin
152:    if v_stat_int1 = 0 then
153:      begin
154:        v_ufr[b_taste_push] := 1;  // Taster Flag setzen
155:        v_stat_int1 := 1;
156:      end
157:      else
158:        begin
159:          v_ufr[b_taste_push] := 0;  // Taster Flag wieder löschen
160:          v_stat_int1 := 0;
161:        end;
162:     v_ufr[b_db_neu]:= 1;
163: end;
164:
165: // PWM Steuerung der LCD Hintergrundbeleuchtung ------------------
166: // Timer-Overflow Interrupt mit Abfrage der Up-Down-Tasten
167: procedure ovf2int(); org 0x008; // Timer 2 Overflow Interrupt
168: var
169:   v_zwi : byte;
170: begin
171:   ocr2 := v_pwm;        // Output Compare Register mit PWM-Wert neu laden
172: // Tasten im Timerinterrupt abfragen --------------------------
173:   v_zwi := 0;
174:   v_zwi := PINB;                // Tastereingänge aus Port B laden
175:   v_zwi := not v_zwi;           // und invertieren (Low aktiv)
176:   v_zwi := (v_zwi and 0x0f);    // Taster maskieren (Bit 0 bis 3)
177:   if (v_stat_ovint = 0) and (v_zwi > 0) then
178:     begin                       // Nur einmal ausführen solange Taster aktiv
179:       v_sw := v_zwi;            // Tasterbits in Variable übernehmen
180:       v_stat_ovint := 1;        // Statusbit aktiv setzen
181:       v_ufr[b_db_neu] := 1;     // Berechnung neustarten
182:     end;
183:   if (v_zwi = 0) then v_stat_ovint := 0; // Statusbit inaktiv setzen
184:                                          // wenn kein Taster mehr betätigt ist
185: end;
186:
```

```
187: // Messwerte vom 10 Bit Analog/Digitalwandler lesen ---------------------
188: procedure adcint(); org 0x01C;
189: begin
190:                                 // wird momentan nicht benötigt
191: end;
192:
193: procedure twiint(); org 0x0022;
194: begin
195:                                 // wird momentan nicht benötigt
196: end;
197: // Ende der Interruptservice-Routinen --------------------------------
198:
199: // Unterprogramme ----------------------------------------------------
200: procedure p_init;                  // Prozessor initialisieren
201: Var i : byte;
202: Begin
203:    DDRA := k_ddr_a;                // Port A Display, alles Output
204:    DDRB := k_ddr_b;                // Disp. PB7=RS, PB6=E, PB0-PB4 = Sw1 - Sw4
205:    PORTB := k_port_b;              // Pullup-Widerstände für Sw1 - Sw4 ein
206:    DDRC := k_ddr_c;                // Ansteuerung Attenuator, alles Output
207:    DDRD := k_ddr_d;                // Datenrichtung Taster + PWM Port
208:    PORTD := k_port_d;              // Pull-Up-Widerstände Port D, 7 PWM-Port
209:    portd.b7 := 1;                  // Hintergrundbeleuchtung ein
210: // LCD-Anzeige initialisieren ----------------------------------------
211:    Lcd_Init();                     // Display initialisieren
212:    lcd_cmd(_lcd_cursor_off);       // und Cursor ausschalten
213:    v_zeile :=  k_zeile1;
214:    lcd_out(1,1,v_zeile);           // Startmeldung ausgeben
215:    v_zeile :=  k_zeile2;
216:    lcd_out(2,1,v_zeile);
217: // Werte aus EEPROM laden -------------------------------------------
218:    v_pwm := EEPROM_Read(k_eep_pwm);   // PWM-Wert aus EEPROM laden
219: // PWM Steuerung initialisieren --------------------------------------
220:    ocr2 := v_pwm;                     // Output Compare Register vorbesetzen
221:    tccr2 := k_tccr2;                  // Timer 2 initialisieren
222:    timsk.toie2 := 1;                  // Timer 2, Overflow Interrupt ein
223: // Globale Variablen initialisieren ----------------------------------
224:    v_db := 0;                         // db-Wert löschen
225:    v_db_off := 0;                     // Offset löschen
226:    v_sw := 0;                         // Schalter löschen
227: // Die Statusbits müssen global vereinbart werden, da lokale Variablen
228: // bei jedem Prozedur Aufruf gelöscht werden!
229:    v_stat_net := 0;                   // Status in der Steuerung
230:    v_stat_int1 := 0;                  // Status im Interrupt 1
231:    v_stat_ovint := 0;                 // Status im Overflow-Interrupt
232: // User Flag-Register für die Programmsteuerung
233:    for i := 1 to 10 do v_ufr[i] := 0; // User-Flag Register löschen
234:    v_ufr[b_db_neu] := 1;              // Berechnung neustarten
235:    v_ufr[b_anz_neu]:= 1;              // Anzeige aktivieren
236:
237:    SREG_I_bit := 1;                   // Global Interrupt enable
238: end;
239:
240: // Hintergrundbeleuchtung einstellen ---------------------------------
241: // Wird ausgeführt, wenn beim Einschalten die Down1x-Taste gedrückt wird.
242: procedure p_pwm;
243: begin
244:   v_zeile := '  Beleuchtung  ';
245:   lcd_out(1,1,v_zeile);
246:   v_zeile := '  einstellen   ';
247:   lcd_out(2,1,v_zeile);
248:   while pinb.b_sw2 = 0 do;        // Warten bis Down1x-Taste losgelassen wird
```

```
249:   repeat
250:     if ((pinb.b_sw3 = 0) and (pinb.b_sw4 = 1)) then
251:       if (v_pwm < 0xff) then      // prüfen ob Maxwert erreicht
252:          v_pwm := v_pwm + 1;      // Wenn nein, v_pwm incrementieren
253:     if ((pinb.b_sw4 = 0) and (pinb.b_sw3 = 1)) then
254:       if (v_pwm > 3) then         // prüfen ob Minwert erreicht
255:          v_pwm := v_pwm - 1;      // Wenn nein, v_pwm decrementieren
256:    delay_ms(10);                 // Verzögerung für Tastenbedienung
257:    until (pinb.b_sw1 = 0);       // Einstellung mit Up1x beenden
258:    v_zeile := '  Beleuchtung  ';
259:    lcd_out(1,1,v_zeile);
260:    v_zeile := '  eingestellt  ';
261:    lcd_out(2,1,v_zeile);
262:    EEPROM_write(k_eep_pwm, v_pwm); // PWM-Wert im EEPROM ablegen
263:    delay_ms(500);                 // Verzögerung für Anzeige
264:    v_ufr[b_taste_up1] := 0;       // Tasten-Flag wieder löschen
265:    v_ufr[b_taste_down1] := 0;     // Tasten-Flag wieder löschen
266: end;
267:
268: // Startprozedur, wird nur einmalig beim Einschalten durchgeführt -----------
269: procedure p_start;          // Gerät initialisieren
270: begin
271: // 1 s Verzögerung für Startbildschirm
272:   delay_ms(1000);
273:   if (pinb.b_sw2 = 0) then p_pwm; // Falls Down1x gedückt, Beleuchtung anpassen
274:   repeat until (pinb.b_sw1 = 1);   // Warten bis Up1x-Taste losgelassen wird
275: //Tasteninterrupts einschalten ---------------------------------------------
276:   mcucr.isc10 := 0;
277:   mcucr.isc11 := 1;          // Interrupt 1 bei Low-Flanke Taste Drehgeber INT1
278:   mcucr.isc00 := 0;
279:   mcucr.isc01 := 1;          // Interrupt 0 bei Low-Flanke Drehgeber "A"  INT0
280:   gicr.int0 := 1;            // Interrupt 0 Taste Drehgeber "A" einschalten
281:   gicr.int1 := 1;            // Interrupt 1 Taste Drehgeber einschalten
282:   v_sw := 0;                 // Tasten-Variable löschen
283:   v_db := 0;
284:   v_ufr[b_anz_neu] := 1;     // Flag für LCD-Anzeige setzen
285: end;
286:
287: // Schalter auswerten -------------------------------------------------
288: // Hier wird das untere Nibble der Schaltervariable ausgewertet. Momentan
289: // werden nur die Einzelbetätigungen der Schalter ausgewertet. Mögliche
290: // Doppelbetätigungen können bei Bedarf noch eingefügt werden.
291: procedure p_switch;
292: begin
293:   if v_sw > 0 then
294:     begin               // Je nach gedrückter Tastenkombination wird
295:       case v_sw of      // die entsprechende Anweisung ausgeführt
296:         1: begin        // Taste Down1x betätigt
297:              if v_db > 0 then v_db := v_db - 1;
298: //             if v_db <= 126 then v_db := v_db + 1;
299:            end;
300:         2: begin        // Taste Up1x betätigt
301:              if v_db <= 126 then v_db := v_db + 1;
302: //             if v_db > 0 then v_db := v_db - 1;
303:            end;
304:         3: begin  end;  // Taste Down1x und Up1x  betätigt
305:         4: begin        // Taste Down10x betätigt
306:              if v_db > 9 then v_db := v_db - 10;
307: //             if v_db <= 117 then v_db := v_db + 10;
308:            end;
309:          5: begin  end; // Taste Up1x und Up10x  betätigt
310:          6: begin  end; // Taste Down1x und Up10x  betätigt
```

```
311:           7: begin  end; // Taste Down1x, Up1x und Up10x betätigt
312:           8: begin       // Taste Down1x betätigt
313:               if v_db <= 117 then v_db := v_db + 10;
314: //              if v_db > 9 then v_db := v_db - 10;
315:              end;
316:           9: begin  end; // Taste Down10x und Down1x  betätigt
317:          10: begin  end; // Taste Down10x und Up1x  betätigt
318:          11: begin  end; // Taste Down10x, Up1x und Down1x betätigt
319:          12: begin  end; // Taste Down10x und Up10x  betätigt
320:          13: begin  end; // Taste Down10x, Up10x und Up1x betätigt
321:          14: begin  end; // Taste Down10x, Up10x und Down1x betätigt
322:          15: begin  end; // Taste Down10x, Up10x, Down1x und Up1x betätigt
323:        end;
324:        v_sw := 0;
325:        v_ufr[b_db_neu] := 1;
326:      end;
327: end;
328:
329: // Bildung des Dämpfungswertes und Anzeigewert generieren -------
330: procedure p_db;
331: var
332:    v_ausgabe : byte;
333:    ee_adress: word;
334: begin
335: // Prüfen ob ein neuer dB-Wert vorliegt -------------------------
336:   If v_ufr[b_db_neu] = 1 then       // berechnen wenn neuer dB-Wert vorliegt
337:      begin
338:        v_db_anz := v_db;            // dB-Wert in Anzeigevariable übernehmen
339: // Offset prüfen -----------------------------------------------
340:        if (v_ufr[b_taste_push] = 1) and (v_ufr[b_offset] = 0) then
341:          begin
342:            v_db_off := v_db;        // aktueller dB-Wert als Offset übernehmen
343:            v_ufr[b_offset] := 1;    // Offset-Flag setzen
344:          end;
345:        if (v_ufr[b_taste_push] = 0) and (v_ufr[b_offset] = 1) then
346:          begin
347:            v_db_off := 0;           // Offset und Flag wieder löschen
348:            v_ufr[b_offset] := 0;
349:          end;
350: // Anzeigewert bilden ------------------------------------------
351:        v_db_anz := (v_db_anz - v_db_off);
352: // Vorzeichen setzen
353:        if (v_db_anz + v_db_off) >= 0 then v_db_anz := (v_db_anz * -1);
354: // Schaltwert aus EEPROM laden und an Attenuator ausgeben ----------
355:        ee_adress := v_db;           // dB-Wert als Tabellenadresse laden
356:        v_ausgabe := EEPROM_READ(ee_adress); // Schaltmuster aus Tabelle laden
357:        PORTC := v_ausgabe;          // Attenuator schalten
358:        v_ufr[b_db_neu] := 0;        // Flag wieder löschen
359:        v_ufr[b_anz_neu] := 1;       // Flag für LCD-Anzeige setzen
360:      end;
361: end;
362:
363: // Bildschirmausgabe -------------------------------------------
364: procedure p_lcd;
365: Var
366:   v_text : array[3] of char;    // Variable für die Ausgabewerte als Text
367: begin
368:  if (v_ufr[b_anz_neu] = 1) Then
369:   begin                        // Nur ausführen wenn neuer dB-Wert vorliegt
370:     ShortToStr(v_db_anz, v_text);
371:         begin
372:           if (v_ufr[b_offset] = 1) then
```

```
373:             begin
374:               v_zeile := 'Out =        dBm';  // wenn Offset geschaltet
375:               lcd_out(1,1,v_zeile);
376:             end
377:             else
378:             begin
379:               v_zeile := 'Att =        dB ';  // Wenn kein Offset
380:               lcd_out(1,1,v_zeile);
381:             end;
382:           lcd_out(1,8,v_text);                 // dB-Wert anzeigen
383:           v_zeile := 'Offset =     dB ';       // Offsetzeile eintragen
384:           lcd_out(2,1,v_zeile);
385:           ByteToStr(v_db_off, v_text);
386:           lcd_out(2,9,v_text);                 // Offset-Wert anzeigen
387:     end;
388:     delay_ms(100);               // Verzögerung für Anzeigestabilität
389:   end;
390:   v_ufr[b_anz_neu] := 0;
391: end;
392:
393: // Ende Unterprogramme -------------------------------------------------
394:
395: // --------------------------------------------------------------------
396: {Hauptprogramm}
397: // --------------------------------------------------------------------
398: begin
399:   p_init;                          // System initialisieren
400:   p_start;                         // Startprozedur,
401:   while true do                    // Endlosschleife Hauptprogramm
402:    begin
403:      p_switch;                     // Schalter einlesen
404:      p_db;                         // dB-Wert bilden
405:      p_lcd;                        // und auf Display ausgeben
406:    end;
407: end.
```