```
1   ;*****************************************************************************************
2   ;* Test: Ansteuerung eines DDS-Chips AD9952 von Analog Devices                           *
3   ;* Arduino 2009 mit ATmega328P, 16 MHz Quarz                                             *
4   ;* Portbelegungen: LCD:PC0-3=DB4-7,PC4=RS,PC5=R/W PD7=E_LCD                              *
5   ;* DDS: PB0=RESET, PB1=/CS, PB2=I/O-UPDATE, PB3=SDIO, PB4=SDO, PB5=SCLK                  *
6   ;* PD0/1 frei für UART, PD2/3=Drehgeber Interrupt, PD4-6 noch frei                       *
7   ;* Christoph Kessler Juni 2016                                                           *
8   ;*****************************************************************************************
9   .nolist
10  .include "m328Pdef.inc"
11  .list
12  ;*****************************************************************************************
13  ;* Register-Variablen 1-15 ( keine "immediate"-Befehle möglich ):
14  ;*****************************************************************************************
15  .def    LpmReg  = r0    ; pointer for SRAM read/write
16  .def    SavSrg  = r1    ; save status register during interrupt
17  .def    reg02   = r2    ;
18  .def    reg03   = r3    ;
19  .def    reg04   = r4    ;
20  .def    reg05   = r5    ;
21  .def    reg06   = r6    ;
22  .def    reg07   = r7    ;
23  .def    reg08   = r8    ;
24  .def    reg09   = r9    ;
25  .def    reg10   = r10   ;
26  .def    reg11   = r11   ;
27  .def    reg12   = r12   ;
28  .def    reg13   = r13   ;
29  .def    reg14   = r14   ;
30  .def    reg15   = r15   ;
31  ;*****************************************************************************************
32  ;* Register-Variablen 16-23 ( "immediate"-Befehle möglich ):
33  ;*****************************************************************************************
34  .def    Cnt     = r16   ; counter
35  .def    Temp    = r17   ; temporary register 1
36  .def    Temp2   = r18   ; temporary register 2
37  .def    Flag    = r19   ; flag for irq / main handshake
38  .def    hexL    = r20   ; Umrechnung hex/dez low Byte
39  .def    hexH    = r21   ; Umrechnung hex/dez high Byte
40  .def    Cnt1    = r22   ; zweiter Zähler
41  .def    Cnt2    = r23   ; dritter Zähler
42  ;*****************************************************************************************
43  ;* Register-Variablen 24-31 ( 16Bit-Register ):
44  ;*****************************************************************************************
45  .def    EncodL  = r24   ;
46  .def    EncodH  = r25   ;
47  ;.def   reg26   = r26   ;
48  ;.def   reg27   = r27   ;
49  ;.def   reg28   = r28   ;
50  ;.def   reg29   = r29   ;
51  ; ZL            = r30   ; Zeiger auf SRAM low
52  ; ZH            = r31   ; Zeiger auf SRAM high
53  ;*****************************************************************************************
54  ;* Konstanten-Definitionen:
55  ;*****************************************************************************************
56  .equ    RW      = 5     ; LCD-Display: Read/notWrite = Bit5 von PortC
57  .equ    RS      = 4     ; LCD-Display: Register Select = Bit4 von PortC
58  .equ    E_LCD   = 7     ; LCD-Display: Enable Impuls = Bit7 von PortD
59  .equ    Busy    = 3     ; LCD-Display: Busy-Flag = Bit3 von PortC
60  .equ    EncStp  = 1     ; EncoderStep
61  ;
62  .equ    RESET   = 0     ; DDS-Serial:  RESET = Bit0 von PortB
63  .equ    nCS     = 1     ; DDS-Serial:  /CS = Bit1 von PortB
64  .equ    IOupd   = 2     ; DDS-Serial:  I/O-UPDATE = Bit2 von PortB
65  .equ    SDIO    = 3     ; DDS-Serial:  SDIO = Bit3 von PortB =MOSI
66  .equ    SDO     = 4     ; DDS-Serial:  SDO = Bit4 von PortB =MISO
```

```
67  .equ    SCLK    = 5     ; DDS-Serial:  SCLK = Bit5 von PortB = SCK
68  ;
69  .cseg
70  ;***************************************************************************************
71  ;* Reset/Interrupt-Vektoren ATMega328P (2 Worte / Vektor):
72  ;***************************************************************************************
73          rjmp    Initial nop     ; 0x0000 nach RESET zum Hauptprogramm
74          rjmp    ExtInt0 nop     ; 0x0002 Interrupt Impulsgeber Kanal A
75          rjmp    ExtInt1 nop     ; 0x0004 Interrupt Impulsgeber Kanal B
76          reti nop                ; 0x0006 jmp PCINT0     PCINT0 Handler
77          reti nop                ; 0x0008 jmp PCINT1     PCINT1 Handler
78          reti nop                ; 0x000A jmp PCINT2     PCINT2 Handler
79          reti nop                ; 0x000C jmp WDT        Watchdog Timer Handler
80          reti nop                ; 0x000E jmp TIM2_COMPA Timer2 Compare A Handler
81          reti nop                ; 0x0010 jmp TIM2_COMPB Timer2 Compare B Handler
82          reti nop                ; 0x0012 jmp TIM2_OVF   Timer2 Overflow Handler
83          reti nop                ; 0x0014 jmp TIM1_CAPT  Timer1 Capture Handler
84          reti nop                ; 0x0016 jmp TIM1_COMPA Timer1 Compare A Handler
85          reti nop                ; 0x0018 jmp TIM1_COMPB Timer1 Compare B Handler
86          reti nop                ; 0x001A jmp TIM1_OVF   Timer1 Overflow Handler
87          reti nop                ; 0x001C jmp TIM0_COMPA Timer0 Compare A Handler
88          reti nop                ; 0x001E jmp TIM0_COMPB Timer0 Compare B Handler
89          reti nop                ; 0x0020 jmp TIM0_OVF   Timer0 Overflow Handler
90          reti nop                ; 0x0022 jmp SPI_STC    SPI Transfer Complete Handler
91          reti nop                ; 0x0024 jmp USART_RXC  USART, RX Complete Handler
92          reti nop                ; 0x0026 jmp USART_UDRE USART, UDR Empty Handler
93          reti nop                ; 0x0028 jmp USART_TXC  USART, TX Complete Handler
94          reti nop                ; 0x002A jmp ADC        ADC Conversion Complete Handler
95          reti nop                ; 0x002C jmp EE_RDY     EEPROM Ready Handler
96          reti nop                ; 0x002E jmp ANA_COMP   Analog Comparator Handler
97          reti nop                ; 0x0030 jmp TWI        2-wire Serial Interface Handler
98          reti nop                ; 0x0032 jmp SPM_RDY    Store Program Memory Ready Handler
99  ;***************************************************************************************
100 ;* nach Reset: Stack,Ports initialisieren
101 ;***************************************************************************************
102 Initial:
103         ldi     Temp,Low(RamEnd)
104         out     spl,Temp        ;
105         ldi     Temp,High(RamEnd)
106         out     sph,Temp        ; Stack initialisiert
107         ldi     Temp,0b00000010 ; $02
108         out     PortB,Temp      ; PB Ausgang DDS /CS auf high, Rest low
109         clr     Temp            ; $00
110         out     PortC,Temp      ; PC alle Ausgänge auf low
111         out     PortD,Temp      ; PD alle Ausgänge auf low
112         ldi     Temp,0b00101111 ; $2F
113         out     DDRB,Temp       ; DDS: PB4 input, Rest output, PB6/7 nicht vorhanden
114         ldi     Temp,0b00111111 ; $3F
115         out     DDRC,Temp       ; LCD: PC0-5 outputs, PC6/7 nicht vorhanden
116         ldi     Temp,0b10000000 ; $80
117         out     DDRD,Temp       ; LCD: PD7 output, Rest inputs
118         clr     EncodL          ;
119         clr     EncodH          ;
120 ;***************************************************************************************
121 ;* LC-Display initialisieren
122 ;***************************************************************************************
123         ldi     Temp,150        ; 150*100usec = ca.15msec
124         rcall   Wait100         ; wait for display ready
125         ldi     Temp,0b00000011 ;
126         rcall   LCDOut          ; 0011 to LCD
127         ldi     Temp,41         ; 41*100usec = ca.4,1msec
128         rcall   Wait100         ;
129         ldi     Temp,0b00000011 ;
130         rcall   LCDOut          ; again 0011 to LCD
131         ldi     Temp,1          ; 100usec
132         rcall   Wait100         ;
```

```
133           ldi     Temp,0b00000011 ;
134           rcall   LCDOut          ; 3 times 0011 to LCD
135           ldi     Temp,1          ; 100usec
136           rcall   Wait100         ;
137           ldi     Temp,0b00000010 ;
138           rcall   LCDOut          ; 0010 = set 4 bit mode
139           ldi     Temp,1          ; 100usec
140           rcall   Wait100         ;
141
142           ldi     Temp,0b00101000 ; 2 lines 5*7dots
143           rcall   LCDCmd          ;
144           ldi     Temp,0b00001100 ; display off
145           rcall   LCDCmd          ;
146           ldi     Temp,0b00000001 ; display clear
147           rcall   LCDCmd          ;
148           ldi     Temp,0b00000110 ; increment, no shift
149           rcall   LCDCmd          ;
150   ;*****************************************************************************************
151   ;* Starttext auf Display ausgeben
152   ;*****************************************************************************************
153   UpLine:
154           ldi     Temp,0b10000000 ; display-RAM, 1st Byte upper line
155           rcall   LCDCmd
156           ldi     ZH,high(UpLTbl) ;
157           ldi     ZL,low(UpLTbl)  ; textstring "AD9952  DDS Test"
158           lsl     ZL              ; Z*2 (16bit-addressing)
159           rol     ZH
160           ldi     Cnt,15          ; 16 characters
161   UpLin1:
162           lpm
163           mov     Temp,LpmReg     ; to display, upper line
164           rcall   LCDDat
165           adiw    ZL,1            ; next one
166           dec     Cnt             ; 16 reached ?
167           brge    UpLin1
168   LoLine:
169           ldi     Temp,0b11000000 ; display-RAM, 1st Byte lower line
170           rcall   LCDCmd
171           ldi     ZH,high(LoLTbl) ;
172           ldi     ZL,low(LoLTbl)  ; textstring "Drehgeber: XXXXX"
173           lsl     ZL              ; Z*2 (16bit-addressing)
174           rol     ZH
175           ldi     Cnt,15          ; 16 characters
176   LoLin1:
177           lpm
178           mov     Temp,LpmReg     ; to display, lower line
179           rcall   LCDDat
180           adiw    ZL,1            ; next one
181           dec     Cnt             ; 16 reached ?
182           brge    LoLin1
183
184   ;*****************************************************************************************
185   ;* DDS und SPI initialisieren
186   ;*****************************************************************************************
187   DDSinit:
188           sbi     PortB,RESET     ; DDS RESET _/~  DDS reset auf default-Werte
189
190           ldi     Temp,10         ; 10*50usec = ca.0,5msec
191           rcall   Wait100         ; wait
192
193           cbi     PortB,RESET     ; DDS RESET ~\_
194           ldi     Temp,$00        ; SPI vorbereiten
195           out     SPSR,Temp       ; SPI-StatusRegister  = % 0000 0000
196           ldi     Temp,$50        ; SPI-ControlRegister = % 0101 0000
197           out     SPCR,Temp       ; (SPE,MSTR)=1 (SPIE,DORD,CPOL,CPHA,SPR1,SPR0)=0
198   ;*****************************************************************************************
```

```
199  ;* alle DDS-Register mit Startwerten füllen
200  ;*****************************************************************************************
201          ldi     ZL,low(DDSTbl<<1)       ; Tabellenzeiger initialisieren
202          ldi     ZH,high(DDSTbl<<1)      ; links schieben wegen 16bit-Adressen
203          ldi     Cnt,13          ; Anzahl der Tabellen-Bytes
204          cbi     PortB,nCS       ; DDS /CS ~\_ SPI belegen
205  DDSCnf:
206          lpm     Temp,Z+         ; ein Byte aus der Tabelle holen
207          out     SPDR,Temp       ; Byte via SPI senden, MSB zuerst
208  WaitSPI:
209          in      Temp,SPSR       ; warten auf SPI
210          sbrs    Temp,SPIF       ;
211          rjmp    WaitSPI         ;
212          dec     Cnt             ; Anzahl Cnt erreicht?
213          brne    DDSCnf          ;
214          sbi     PortB,nCS       ; DDS /CS _/~ SPI freigeben
215          sbi     PortB,IOupd     ; DDS IOupd _/~  DDS auf neue Werte umstellen
216          nop
217          nop
218          nop
219          nop
220          nop                     ; at least 2 SYNC_CLK cycles = 1/(25 MHz)= 40 ns
221          nop                     ; 1/(16 MHz)= 62,5ns
222          nop
223          nop
224          nop
225          nop
226          cbi     PortB,IOupd     ; DDS IOupd ~\_
227  ;       sbi     PortB,nCS       ; DDS /CS _/~ SPI freigeben
228
229          ldi     Temp,150        ; 150*50usec = ca.7,5msec
230          rcall   Wait100         ; wait for display ready
231
232          rjmp    DDSinit         ; erst mal dauernd wiederholen, zur Kontrolle
233  /*
234  ;*****************************************************************************************
235  ;* Encoder-Interrupt starten:
236  ;*****************************************************************************************
237          clr     Flag
238          ldi     Temp,$0F        ; ExtInt0/1 beide auf _/~
239          sts     EICRA,Temp      ; ExternalInterruptControlReg.A(=069h>063h)
240          ldi     Temp,$03        ; Bit0/1, Rest egal
241          out     EIMSK,Temp      ; ExternalInterruptMaskRegister
242          sei                     ; Interrupt Enable Flag
243
244
245  ;*****************************************************************************************
246  ;* Hauptprogramm:
247  ;*****************************************************************************************
248  Haupt:
249          rjmp    Haupt           ; erstmal nix tun
250
251  ;*****************************************************************************************
252  ;* Encoder-Anzeige aktualisieren
253  ;*****************************************************************************************
254          tst     Flag            ; Null = Änderung durch Interrupt
255          brne    Haupt           ; sonst return
256          rcall   Disp            ; auf Display ausgeben
257          rjmp    Haupt           ; Endlosschleife
258  */
259  ;*****************************************************************************************
260  ;* Unterprogramm: Anzeige aktualisieren falls neuer Inkrementalgeberwert
261  ;*****************************************************************************************
262  Disp:
263          com     Flag            ; Flag auf $FF
264          ldi     Temp,$0C        ; Display on, Cursor off, Blinken off
```

```
265           rcall    LCDCmd          ; Befehl senden
266           ldi      Temp,$CB        ; Display nicht löschen, Cursor Home
267           rcall    LCDCmd          ; Befehl senden
268
269           mov      hexL,EncodL     ; Encoder-Wert in Dez.Umrechnen
270           mov      hexH,EncodH     ;
271           rcall    HexBCD          ;
272
273           ldi      ZH,high(TxtBuf)
274           ldi      ZL,low(TxtBuf)  ; Z-Ptr zeigt auf Textanfang
275           ldi      Cnt1,5          ; Text hat 5 Zeichen
276           rcall    OutText         ; gibt Text im Buffer aus
277           ret
278   ;*****************************************************************************************
279   ;* Unterprogramm: Gibt <Cnt1> Zeichen an das Display aus
280   ;*****************************************************************************************
281   OutText:
282           ld       Temp,Z          ; Zeichen-Code nach Temp laden
283           rcall    LCDDat          ; Zeichen senden, Autoinkrement der Adresse
284           adiw     ZL,1            ; Z-Ptr inkrementieren
285           dec      Cnt1            ; Zähler - 1
286           brne     OutText         ; alle Zeichen an das LCD ausgeben
287           ret
288   ;*****************************************************************************************
289   ;* Unterprogramm: LC-Display: Befehls- oder Datenbyte in Temp ausgeben
290   ;*****************************************************************************************
291   LCDCmd:
292           clt                      ; command to LCD , T-Flag=RS=0
293           rjmp     LCD1
294   LCDDat:
295           set                      ; data to LCD, T-Flag=RS=1
296   LCD1:
297           push     Temp            ; auf busy-flag warten
298           ldi      Temp,$30        ; R/W,RS=output,DB4..7=input
299           out      DDRC,Temp       ;
300           ldi      Temp,$20        ; RS=0,R/W=1=read
301           out      PortC,Temp      ;
302   LCD2:
303           sbi      PortD,E_LCD     ; Enable _/~
304           nop                      ;
305           nop                      ; Enable min. 450 ns
306           nop                      ;
307           nop                      ;
308           nop                      ;
309           nop                      ;
310           nop                      ;
311           nop                      ;
312           in       Temp,PinC       ; read BusyFlag
313           cbi      PortD,E_LCD     ; Enable ~\_
314           nop                      ;
315           nop                      ; Enable min. 450 ns
316           nop                      ;
317           nop                      ;
318           nop                      ;
319           nop                      ;
320           nop                      ;
321           nop                      ;
322           sbi      PortD,E_LCD     ; Enable _/~ , dummy for 2nd nibble
323           nop                      ;
324           nop                      ; Enable min. 450 ns
325           nop                      ;
326           nop                      ;
327           nop                      ;
328           nop                      ;
329           nop                      ;
330           nop                      ;
```

```
331           cbi     PortD,E_LCD     ; Enable ~\_
332           nop                     ;
333           nop                     ;
334           nop                     ;
335           nop                     ;
336           nop                     ;
337           nop                     ;
338           nop                     ;
339           nop                     ;
340           sbrc    Temp,Busy       ; Skip, if LCD ready (Busy-Flag=0)
341           rjmp    LCD2            ;
342           ldi     Temp,$3F        ; (R/W,RS,DB4..7)=outputs
343           out     DDRC,Temp       ;
344           pop     Temp            ; fetch databyte
345           push    Temp            ; save low nibble
346           swap    Temp            ; swap high nibble to low
347           andi    Temp,$0F        ; DB4..7,RW=0=write
348           bld     Temp,RS         ; RS=T-flag - instruction/data
349           out     PortC,Temp      ; output high nibble
350           sbi     PortD,E_LCD     ; Enable _/~ (min 450ns)
351           nop                     ;
352           nop                     ;
353           nop                     ;
354           nop                     ;
355           nop                     ;
356           nop                     ;
357           pop     Temp            ; fetch low nibble
358           andi    Temp,$0F        ; DB4..7,RW=0=write
359           cbi     PortD,E_LCD     ; Enable ~\_
360           bld     Temp,RS         ; RS=T-flag - instruction/data
361   LCDout:
362           out     PortC,Temp      ; output low nibble
363           sbi     PortD,E_LCD     ; Enable _/~
364           nop                     ;
365           nop                     ; Enable min. 450 ns
366           nop                     ;
367           nop                     ;
368           nop                     ;
369           nop                     ;
370           nop                     ;
371           nop                     ;
372           cbi     PortD,E_LCD     ; Enable ~\_
373           ret
374
375   ;***************************************************************************************
376   ;* Unterprogramm: warte ca. TEMP*50usec
377   ;***************************************************************************************
378   Wait100:
379           clr     Temp2           ; (1)
380   Wait1:
381           dec     Temp2           ; (1) 1/(16MHz)=62,5ns -> *256=48µs
382           brne    Wait1           ; (2) 1st loop 48 usec
383           dec     Temp            ; (1)
384           brne    Wait1           ; (2) 2nd loop
385           ret
386
387   ;***************************************************************************************
388   ;* Interruptprogramm für Inkrementalgeber
389   ;***************************************************************************************
390   ExtInt0:
391           in      SavSrg,SREG     ; save status register
392           lds     Flag,EICRA      ; Int0  _/~ oder ~\_ ?
393           sbrc    Flag,0          ; Überspringe nächsten Befehl falls ~\_
394           rjmp    UpA             ; es war _/~ also Sprung zu UpA
395           ori     Flag,$01        ; EICRA-Bit0 = 1
396           sts     EICRA,Flag      ; Int0 für nächstes Mal umschalten auf _/~
```

```
397             sbic    PinD,3          ; Überspringe nächsten Befehl falls Kanal B=0
398             rjmp    IncEnc          ; wenn B=1 und A=~\_ -> inkrementieren
399             rjmp    DecEnc          ; wenn B=0 und A=~\_ -> dekrementieren
400     UpA:
401             andi    Flag,$FE        ; EICRA-Bit0 = 0
402             sts     EICRA,Flag      ; Int0 für nächstes Mal umschalten auf ~\_
403             sbic    PinD,3          ; Überspringe nächsten Befehl falls Kanal B=0
404             rjmp    DecEnc          ; wenn B=1 und A=_/~ -> dekrementieren
405             rjmp    IncEnc          ; wenn B=0 und A=_/~ -> inkrementieren
406
407     ExtInt1:
408             in      SavSrg,SREG     ; save status register
409             lds     Flag,EICRA      ; Int1  _/~ oder ~\_ ?
410             sbrc    Flag,2          ; Überspringe nächsten Befehl falls ~\_
411             rjmp    UpB             ; es war _/~ also Sprung zu UpB
412             ori     Flag,$04        ; EICRA-Bit2 = 1
413             sts     EICRA,Flag      ; Int1 für nächstes Mal umschalten auf _/~
414             sbic    PinD,2          ; Überspringe nächsten Befehl falls Kanal A=0
415             rjmp    DecEnc          ; wenn A=1 und B=~\_ -> dekrementieren
416             rjmp    IncEnc          ; wenn A=0 und B=~\_ -> inkrementieren
417     UpB:
418             andi    Flag,$FB        ; EICRA-Bit2 = 0
419             sts     EICRA,Flag      ; Int1 für nächstes Mal umschalten auf ~\_
420             sbic    PinD,2          ; Überspringe nächsten Befehl falls Kanal A=0
421             rjmp    IncEnc          ; wenn A=1 und B=_/~ -> inkrementieren
422             rjmp    DecEnc          ; wenn A=0 und B=_/~ -> dekrementieren
423
424     IncEnc:
425             inc     EncodL          ; increment low byte
426             brne    IncEn1          ; FFh -> 00h ?
427             inc     EncodH          ; increment high byte
428     IncEn1:
429             clr     Flag            ; set flag = 00
430             out     SREG,SavSrg     ; restore status register
431             reti
432     DecEnc:
433             dec     EncodL          ; decrement low byte
434             cpi     EncodL,$FF
435             brne    DecEn1          ; 00h -> FFh ?
436             dec     EncodH          ; decrement high byte
437     DecEn1:
438             clr     Flag            ; set flag = 00
439             out     SREG,SavSrg     ; restore status register
440             reti
441
442
443     ;***************************************************************************************
444     ;* HexBCD 16-bit binär (ganzzahlig in hexH:hexL ) -> BCD.
445     ;* Das Unterprogramm wandelt eine aus 16 Bit bestehende Hexadezimalzahl
446     ;* in eine 5-stellige BCD-Zahl im TextBuffer um.
447     ;* xxxx (Hex) -> abcde (BCD) 00000...65535
448     ;***************************************************************************************
449     HexBCD:
450             rcall   Div10           ; hexH:hexL / 10, Rest in Temp
451             ori     Temp,$30        ; in ASCII umwandeln
452             sts     TxtBuf+4,Temp   ; Einer in TextBuffer
453             rcall   Div10           ; hexH:hexL / 10, Rest in Temp
454             ori     Temp,$30        ; in ASCII umwandeln
455             sts     TxtBuf+3,Temp   ; Zehner in TextBuffer
456             rcall   Div10           ; hexH:hexL / 10, Rest in Temp
457             ori     Temp,$30        ; in ASCII umwandeln
458             sts     TxtBuf+2,Temp   ; Hunderter in TextBuffer
459             rcall   Div10           ; hexH:hexL / 10, Rest in Temp
460             ori     Temp,$30        ; in ASCII umwandeln
461             sts     TxtBuf+1,Temp   ; Tausender in TextBuffer
462             rcall   Div10           ; hexH:hexL / 10, Rest in Temp
```

```
463             ori     Temp,$30        ; in    ASCII umwandeln
464             sts     TxtBuf,Temp     ; Zehntausender in TextBuffer
465             ret
466
467     ;***************************************************************************************
468     ;* dividiert hexH:hexL durch 10, Erg. in hexH:hexL, Rest in Temp
469     ;***************************************************************************************
470     Div10:
471             clr     Temp
472             lsl     hexL
473             rol     hexH
474             rol     Temp            ; die Division der höchstwertigsten
475             lsl     hexL            ; drei Bits durch 10 kann nur Null
476             rol     hexH            ; ergeben
477             rol     Temp
478             lsl     hexL
479             rol     hexH
480             rol     Temp
481             ldi     Cnt2,13         ; Zähler für restl. 13 Bits
482     Loop10:
483             lsl     hexL            ; Divident um 1 nach links
484             rol     hexH
485             rol     Temp            ; Überlauf in Temp
486             subi    Temp,10         ; Test: Überlauf > 10 ?
487             brlo    HB1             ; Sprung, wenn nicht
488             inc     hexL            ; Divisionszähler um 1 inkrem.
489             rjmp    HB2
490     HB1:
491             subi    Temp,-10        ; Test rückgängig machen
492     HB2:
493             dec     Cnt2            ; Bitzähler dekrementieren
494             brne    Loop10          ; Sprung, wenn noch nicht alle Bits
495             ret
496             nop                     ; segmentation fault in den nachfolgenden Tabellen
497     ;***************************************************************************************
498     ;* Text-Tabelle obere Anzeigezeile
499     ;***************************************************************************************
500     UpLTbl:
501     .db     "AD9952  DDS Test"
502     LoLTbl:
503     .db     "Drehgeber: XXXXX"
504     ;***************************************************************************************
505     ;* DDS-Init-Tabelle, MSBit UND MSByte first! (Tabelle im Datenblatt ist LSByte first)
506     ;***************************************************************************************
507     DDSTbl:
508             .db     0x00,0x02       ;0x00 CFR1      write ContrFunctReg1 (32bit).
509                                     ;0x02 CFR1[31:24] OSK manual enabled
510             .db     0x00,0x02       ;0x00 CFR1[23:16]
511                                     ;0x02 CFR1[15:8]
512             .db     0x40,0x04       ;0x40 CFR1[7:0] (0x40 SyncClk enabled / 0x42 disabled)
513                                     ;0x04 FTW0      write Frequency Tuning Word 0 (32bit)
514             .db     0x0C,0xCC       ;0x00 FTW0[31:24]       100 kHz -> 0020C49B
515                                     ;0x20 FTW0[23:16]        10 MHz -> 0CCCCCCC
516             .db     0xCC,0xCC       ;0x49 FTW0[15:8]         10 kHz -> 000346DC
517                                     ;0x9B FTW0[7:0]
518
519     /*      .db     0x01,0x18       ;0x01 CFR2      write Control Function Register 2 (24 bit)
520                                     ;0x18 CFR2[23:16]
521             .db     0x00,0x00       ;0x00 CFR2[15:8]
522                                     ;0x00 CFR2[7:0] */
523
524             .db     0x02,0x3F       ;0x02 ASF       write Amplitude Scale Factor (16 bit)
525             .db     0xFF,0x00       ;0x3F ASF[15:8]
526                                     ;0xFF ASF[7:0]  und Dummybyte 0x00
527
528     /*
```

```
529           .db     0x03,0x00       ;0x03 ARR       write Amplitude Ramp Rate (8 bit)
530
531           .db     0x05,0x00       ;0x05 POW       write Phase Offset Word (16 bit)
532                                   ;0x00 POW[15:8]
533           .db     0x00            ;0x00 POW[7:0]*/
534
535   /* FrequencyTuningWord FTW-Tabelle
536
537   Freq/Hz FTW/hex
538   10              D6
539   20              1AD
540   30              284
541   40              35A
542   50              431
543   60              508
544   70              5DF
545   80              6B5
546   90              78C
547   100             863
548   200             10C6
549   300             192A
550   400             218D
551   500             29F1
552   600             3254
553   700             3AB8
554   800             431B
555   900             4B7F
556   1.000           53E2
557   2.000           A7C5
558   3.000           FBA8
559   4.000           14F8B
560   5.000           1A36E
561   6.000           1F751
562   7.000           24B33
563   8.000           29F16
564   9.000           2F2F9
565   10.000          346DC
566   20.000          68DB8
567   30.000          9D495
568   40.000          D1B71
569   50.000          10624D
570   60.000          13A92A
571   70.000          16F006
572   80.000          1A36E2
573   90.000          1D7DBF
574   100.000         20C49B
575   200.000         418937
576   300.000         624DD2
577   400.000         83126E
578   500.000         A3D70A
579   600.000         C49BA5
580   700.000         E56041
581   800.000         10624DD
582   900.000         126E978
583   1.000.000       147AE14
584   2.000.000       28F5C28
585   3.000.000       3D70A3D
586   4.000.000       51EB851
587   5.000.000       6666666
588   6.000.000       7AE147A
589   7.000.000       8F5C28F
590   8.000.000       A3D70A3
591   9.000.000       B851EB8
592   10.000.000      CCCCCCC
593   20.000.000      19999999
594   30.000.000      26666666
```

```
40.000.000      33333333
50.000.000      40000000
60.000.000      4CCCCCCC
*/
;
;*****************************************************************************************
;* Textpuffer im SRAM
;*****************************************************************************************
.dseg
.org $100
TxtBuf:
.BYTE 16
ChrBuf:
.BYTE 64
;*****************************************************************************************
```