

```

*****
' Title: AD9833-DDS-ATmega8-Firmware zu Platine TN20140328
' 1 Hz bis 12,5MHz, Schrittweite 1Hz bis 1MHz, Sweep Zeit/Frequenzen
' und Kurvenform einstellbar. Compiler : Getestet mit LunaAVR 2014.r2.4
' Revisions: V1.00, 20/01/2015 - Initiale Version, (c) T. Neveling, 62,8% Flash
'             V1.10, 21/01/2015 - Code-Optimierungen aus User-Forum, 54,5% Flash
'             V1.20, 23/01/2015 - Eigene LCD-Zeichen für Kurvenform, 56,0% Flash
'             V1.30, 26/06/2015 - Langer Rotarydruck Fr => EEPROM      57,3% Flash
*****

```

```

#include "Library/Lcd4.interface"      ' LCD Interface
#include "Library/Math64.module"       ' zur Integer Umrechnung Frequenz => AD9833-Register
#define bool_t as byte

const F_CPU = 8000000
avr.device = atmega8
avr.clock = F_CPU
avr.stack = 48

const TIMERWERT = 6                  ' für 2ms-Interrupts bei Timer0: 256-8MHz * 2ms / 64 = 6
const Loe = "                        " ' 16 Blanks zum Löschen einer LCD-Zeile
const LiX = "Limits tauschen!"       ' Hinweis auf falsche Eingabe, z.B. LowLimit > HighLimit
Const Steps = 40                     ' Stufen für einen Sweep-Zyklus (auf oder ab)

Dim Fr As Long                       ' Frequenz
Dim Fr_u As Long                     ' Wobbelfrequenz: upper
Dim Fr_l As Long                     ' Wobbelfrequenz: lower
Dim F_lim As Long                    ' Hilfsgröße beim Wobbeln
Dim Fr_step As Long                  ' Wobbelstep in Hz
Dim Fr_merk As Long                  ' letzte Frequenz vor dem Wobbeln
Dim Freq_lsb, Freq_msb as Word       ' Hilfsgrößen für AD9833 Frequenzregister
Dim Ctrl as Word                     ' AD9833 Steuerwort
Dim Latency As Byte                  ' Sweep Geschwindigkeit
Dim Keycount As Byte                 ' Zähler für langen Rotary-Tastendruck
Dim Ad As Long                       ' Daten für AD9833
Dim Sw As Long                       ' Schrittweite 1Hz-1MHz, bzw 0 zum Wobbeln
Dim Up As bool_t                     ' Richtung auf/ab beim Wobbeln
Dim Cmd As Byte                      ' Drehgeber-Befehl: 1=rechts, 2=links, 0=nix
Dim Wave As Byte                     ' Kurvenform: 0=Sinus, 1=Dreieck, 2=Rechteck, 3=Rechteck f/2
eeDim EEFr As Long                   ' gespeicherte Frequenz im EEPROM

Dim Merk As Byte                     ' Drehgeber Hilfsvariable
Dim Rotary As Byte                   ' Signalzustände Enkoder
Dim Rotary_old As Byte

' -----
' Encoder Settings
' -----

#define ENC_A_PHASE      as portc.0
#define ENC_B_PHASE      as portc.1
#define ENC_KEY           as portd.0

#define ENC_PIN           as avr.PINC
#define ENC_A_BIT         as 0
#define ENC_A_MASK        as (1<<ENC_A_BIT)
#define ENC_B_BIT         as 1
#define ENC_B_MASK        as (1<<ENC_B_BIT)

ENC_A_PHASE.mode = input, Pullup      ' Drehgeber Phase A
ENC_B_PHASE.mode = input, Pullup      ' Drehgeber Phase B
ENC_KEY.mode = input, Pullup          ' Drehgeber Taste

```

```

'-----
' Key Settings
'-----
portc.2.mode = input, Pullup      ' Taster "left"
portc.3.mode = input, Pullup      ' Taster "down"
portc.4.mode = input, Pullup      ' Taster "up"
portc.5.mode = input, Pullup      ' Taster "right"

'-----
' AD9833 Port Settings
'-----
#define AD9833_LOAD      as portb.2  ' SS    => FSYNC (AD9833) für SPI
#define AD9833_CLOCK     as portb.5  ' SCK   => SCLK  (AD9833)
#define AD9833_DATA      as portb.3  ' MOSI  => SDATA (AD9833)

AD9833_LOAD.mode = output, high    ' Pegel setzen
AD9833_CLOCK.mode = output, high
AD9833_DATA.mode = output, low

' Timer0 für periodische Drehgeber-Abfrage (alle 2ms)
'-----
timer0.clock = 64                  ' Prescaler
timer0.isr = T0_ovr
timer0.value = TIMERWERT
timer0.enable

'-----
' LCD Port Settings
'-----
portd.6.mode = Output, low         ' LCD R/W Signal hier immer = 0 für Schreibmodus

lcd4.PinDB4 = portd.4              ' Anschlussbelegung des Displays an Platine TN20140328
lcd4.PinDB5 = portd.3
lcd4.PinDB6 = portd.2
lcd4.PinDB7 = portd.1
lcd4.PinEN = portd.5
lcd4.PinRS = portd.7
lcd4.init

lcd4.DefChar(mychar0.Addr,0)       ' selbst definierte LCD-Zeichen
lcd4.DefChar(mychar1.Addr,1)       ' zur Darstellung der Kurvenform
lcd4.DefChar(mychar2.Addr,2)
lcd4.DefChar(mychar3.Addr,3)

'-----
' Welcome
'-----
lcd4.textd "DDS 1Hz..12,5MHz"      ' Power-on Meldung
lcd4.cursor.set 2,1               ' Cursor auf Anfang der 2. Zeile
lcd4.textd "26/06/2015 LF1.3"     ' Datum & LunaAVR-Firmware x.y
Fr = EEFr                         ' Startfrequenz aus EEPROM holen
wait 2

SW = 1000                         ' Schrittweite: 1kHz
FR_l = 50000                     ' untere Sweepfrequenz
FR_u = 100000                    ' obere Sweepfrequenz
Latency = 20                     ' 20ms / Sweepstep
Cmd = 0                          ' Drehgeber = idle
Wave = 2                          ' mit Rechteck starten (hat TTL-Pegel)

```

```

LCD_Freq()          ' Startfrequenz & Kurvenform auf LCD Zeile 1
LCD_Step()          ' dto initiale Schrittweite Zeile 2
Freq()              ' Initiallisierung des AD9833 per Soft-SPI

'-----

avr.interrupts.enable

'-----
' Hauptschleife
'-----

DO
  IF (ENC_KEY = 0) THEN          ' Drehgebertaste gedrückt?
    Incr KeyCount                ' langen Tastendruck an zählen

    select case Cmd
    Case 0                      ' langer Tastendruck ohne Drehung
      if KeyCount = 15 then      ' 15 * 150ms = 2,25s
        if Sw <> 0 then
          Show_Settings()        ' Sweep-Parameter anzeigen, EEPROM-Speicherung
        endif
      endif
    Case 1                      ' Rechtsdrehung => Schrittweite ändern
      Step_Change()
    Case 2                      ' Linksdrehung => Kurvenform ändern
      Wave_Change()
    end select

  else                          ' Drehgebertaste nicht gedrückt
    KeyCount = 0
    when Cmd = 1 do F_up()        ' Rechtsdrehung => Frequenz erhöhen
    when Cmd = 2 do F_Down()      ' Linksdrehung => Frequenz verringern
  Endif

  'während des Wobbelns (Sw = 0) die zugehörigen Parameter NICHT ändern!
  if Sw <> 0 Then
    IF PortC.3 = 0 THEN          ' Taster ab
      LCD4.Cursor.Set 2, 1
      IF Fr <= Fr_u THEN          ' wenn < vorhandenem upper Sweepplimit
        Fr_l = Fr                ' aktuelle Frequenz = lower Sweepplimit
        LCD4.Textd "=Startfrequenz!"
      else
        LCD4.Textd LiX            ' Hinweis: Limit exchange
      endif
    endif

    IF PortC.4 = 0 THEN          ' Taster auf
      LCD4.Cursor.Set 2, 1
      IF Fr >= Fr_l THEN          ' wenn >= vorhandenem lower Sweepplimit
        Fr_u = fr                ' aktuelle Frequenz = upper Sweepplimit
        LCD4.Textd "=Stoppfrequenz!"
      else
        LCD4.Textd LiX            ' Hinweis: Limit exchange
      endif
    endif

    IF PortC.2 = 0 THEN          ' Sweep-Pause verkleinern
      when Latency > 2 do Latency = Latency - 2
      Delay_Msg()
    endif
  end if
end do

```

```

IF PortC.5 = 0 THEN          ' Sweep-Pause vergrößern
    when Latency < 254 do Latency = Latency + 2
    Delay_Msg()
endif
waitms 150

else                          ' Sw = 0: kontinuierlich auf/ab wobbeln
    if Up = TRUE THEN        ' inkrementieren
        F_lim = Fr_u - Fr_step
        if Fr < F_lim then    ' Fr + Fr_Step < Fr_u?
            Fr = Fr + Fr_step ' dann Fr_step addieren
        else
            Fr = Fr_u         ' sonst oberes Limit erreicht
        endif
        when Fr >= Fr_u do Up = FALSE

    else                      ' dekrementieren
        F_Lim = Fr_l + Fr_step
        if Fr > F_lim then
            Fr = Fr - Fr_Step
        else
            Fr = Fr_l
        endif
        when Fr <= Fr_l do UP = TRUE
    endif

    Freq()                   ' per SPI in den AD9833 damit
    waitms Latency

endif

Loop

'-----
' ISR Timer0 Overflow
'-----
ISR T0_ovr fastauto          ' Abfrage des Panasonic Drehgebers
    Timer0.value = TIMERWERT ' für nächstes 2ms Intervall

    ' Drehgeber einlesen
    Rotary = (ENC_PIN and (ENC_A_MASK or ENC_B_MASK))
    Merk = Rotary             ' merken für Rotary_old
    Rotary_old = Rotary_old << 2 ' alten Wert hochschieben
    Rotary = Rotary or Rotary_old ' mit neuem Wert ver-ODERN
    Rotary_old = Merk         ' neuen (alten) Wert speichern

    IF Rotary.3 = 0 THEN      ' untere Werte spiegeln
        Rotary = Rotary XOR &b00001111 ' damit bleiben von der ursprünglichen
    endif                    ' Look-up-Table nur 2 Einzelwerte übrig!

    when Rotary = 14 do Cmd = 1 ' Drehung rechts
    when Rotary = 11 do Cmd = 2 ' Drehung links
endISR

'-----
' Prozeduren für Änderungen durch den Drehgeber
'-----
procedure Step_Change()      ' Rechtsdrehung & Taste gedrückt
    If Sw = 0 THEN
        Sw = 1               ' Sweep-Ende, alte Festfrequenz einstellen
        Fr = Fr_Merk
    
```

```

    LCD_Freq()
    Freq()
else
    Sw = Sw * 10                ' Schrittweite erhöhen
    if Sw = 10000000 THEN      ' grösster Wert war 1MHz
        Start_Sweep()         ' hier wird Sw = 0 gesetzt
    endif
endif
LCD_Step()
Cmd = 0                        ' Befehl abgearbeitet
Endproc

procedure Wave_Change()       ' Linksdrehung & Taste gedrückt
    Incr Wave                  ' Kurvenform ändern
    when Wave = 4 do Wave = 0
    LCD_Freq()                 ' hier wird auch das Control-Word gesetzt!
    Freq()
    Cmd = 0
Endproc

Procedure F_up()              ' Rechtssdrehung: Frequenz erhöhen
    IF Sw > 0 THEN
        Fr = Fr + Sw
        IF Fr > 12500000 THEN  ' Max Ausgangsfrequenz = 12,5MHz
            Fr = 0
        endif
        LCD_Freq()
        Freq()
    endif
    cmd = 0
endproc

procedure F_down()            ' Linksdrehung: Frequenz verkleinern
    if Sw > 0 Then
        if fr > sw then
            Fr = Fr - sw
        else
            Fr = 0              ' nur bis 0 Hz ;)
        endif
        LCD_Freq()
        Freq()
    endif
    cmd = 0
endproc

procedure Start_Sweep()
    Sw = 0                     ' Wobbeln starten
    Fr_merk = Fr               ' letzte Frequenz merken
    Fr_Step = Fr_u - Fr_l
    if Fr_step < Steps then
        Fr_step = 1
    else
        Fr_step = Fr_step / Steps ' Frequenzsprung für jeden Step
    endif
    Fr = Fr_u                  ' mit oberer Grenzfrequenz beginnen
    UP = FALSE                  ' zuerst dekrementieren
endproc

' -----
' LCD Ausgaben
' -----

```

```

procedure Clear_LCD_LN1()          ' Zeile 1 löschen
    LCD4.Cursor.Home
    LCD4.textd Loe
    LCD4.Cursor.Home
endproc

procedure Clear_LCD_LN2()          ' Zeile 2 löschen
    LCD4.Cursor.set 2, 1
    LCD4.textd Loe
    LCD4.Cursor.set 2, 1
endproc

procedure LCD_Step()                ' Schrittweite in Zeile 2 ausgeben
    Clear_LCD_LN2()
    IF Sw > 0 then
        LCD4.Text("Step: " + Str(Sw))
    else
        LCD4.Text("Sweep " + Str(Fr_Step))
        Clear_LCD_LN1()
        LCD4.Textd "Ende: Neuer Step"
    endif
endproc

Procedure Delay_Msg()               ' Sweep Verzögerung ausgeben
    Clear_LCD_LN2()
    LCD4.Text("Delay [ms]: " + Str(Latency))
endproc

procedure LCD_Freq()                ' Frequenz & Kurvenform in Zeile 1 ausgeben
    Clear_LCD_LN1()
    LCD4.Text(Str(Fr) + " Hz")
    LCD4.Cursor.set 1, 14
    select case Wave
    case 0
        LCD4.Textd (chr(1) + chr(2) + chr(3))    ' Sinus
        Ctrl = &B0010000000000000
    case 1
        LCD4.Textd ("/" + Chr(0) + "/")          ' Dreieck
        Ctrl = &B00100000000000010
    case 2
        LCD4.Textd "_-_"                          ' Rechteck
        Ctrl = &B00100000000101000
    case 3
        LCD4.Textd ("_" + chr(253) + "_")          ' Rechteck / 2 _÷_
        Ctrl = &B00100000000100000
    end select
endproc

procedure Show_Settings()           ' langer Druck auf Rotary-Taste
    LCD4.Screen.Clear
    LCD4.Text("u=" + Str(Fr_u))
    LCD4.Cursor.set 2, 1
    LCD4.Text("l=" + Str(Fr_l) + " t=" + Str(Latency))
    EEFr = Fr                                ' aktuelle Frequenz im EEPROM speichern
    Wait 2
    LCD_Step()
    LCD_Freq()
endproc

' -----
' Frequenz auf AD9833 ausgeben

```

```

'-----
struct uint64_t                                ' eigener Datentyp 64 Bit Integer
    long  lo64
    long  hi64
endstruct

procedure Freq()                                ' Frequenzregister = Fr * 2^28 / 25MHz
    const DDS_CLOCK_SCALE = 16                  ' * 16, dann wird Fr * 2^32 zum hi-Word der 64 Bit-Variable
    const DDS_CLOCK_FREQ = (25000000 * DDS_CLOCK_SCALE) ' long

    dim ftw as uint64_t

    ftw.lo64 = 0
    ftw.hi64 = Fr                                ' das ist dann schon die Multiplikation mit 2^32

    Math64.Div6432(ftw, ftw, DDS_CLOCK_FREQ)
    Ad = ftw.lo64                                ' fertig - in ftw.lo64 steht das Ergebnis
    ' die unteren Bits 0 - 13 = Freq_lsb
    ' Bit 15 = 0, Bit 14 = 1 => Frequenzregister 0
    Freq_lsb = (Ad.LowWord AND 0x3fff) OR (1<<14)

    Ad = Ad << 2                                ' Bitpattern hochschieben altes Bit 14 nach Bit 16
    ' die folgenden 14 Bits sind Freq_msb (28 Bit gesamt)
    ' Bit 15 = 0, Bit 14 = 1 => Frequenzregister 0
    Freq_msb = (ad.HighWord AND 0x3fff) OR (1<<14)

    ' jeweils 3x2 Byte an AD9833 senden
    DDS_Send_Word(Ctrl)                          ' Steuerwort mit Kurvenform
    DDS_Send_Word(Freq_lsb)                      ' 28 Bit für Frequenzregister 0
    DDS_Send_Word(Freq_msb)
endproc

'-----
' Software SPI
'-----

procedure DDS_Send_Word( dat as word )
    dim mask as static word

    AD9833_CLOCK = 1
    AD9833_LOAD = 1
    nop
    AD9833_LOAD = 0                                ' FSYNC high > low = Chip-Select (siehe Datenblatt)

    mask = &B1000000000000000

    while (mask)                                ' 16 Bit herausschieben, mit Bit 15 beginnen
        if ( dat and mask ) then
            AD9833_DATA = 1
        else
            ' entsprechendes Datenbit isolieren & ausgeben
            AD9833_DATA = 0
        endif

        AD9833_CLOCK = 1
        nop
        AD9833_CLOCK = 0                        ' mit fallender Taktflanke an AD9833 übergeben

        mask = mask >> 1                        ' nächstes Bit rechts verarbeiten
    wend                                         ' nach 16 Bit ist mask=0

    AD9833_LOAD = 1
    AD9833_CLOCK = 1

```

```
AD9833_DATA = 0
```

```
endproc
```

```
data myChar0                                ' LCD-Zeichen zur Darstellung der Kurvenform
.db &b000000                                ' "\" für Dreieck, Slash ist vorhanden: 0x2F
.db &b10000
.db &b01000
.db &b00100
.db &b00010
.db &b00001
.db &b00000
.db &b00000
enddata
```

```
data myChar1                                ' myChar 1-3 für Sinus
.db &b00011
.db &b00100
.db &b01000
.db &b01000
.db &b10000
.db &b10000
.db &b00000
.db &b00000
enddata
```

```
data myChar2
.db &b10000
.db &b01000
.db &b01000
.db &b00100
.db &b00100
.db &b00010
.db &b00010
.db &b00001
enddata
```

```
data myChar3
.db &b00000
.db &b00000
.db &b00001
.db &b00001
.db &b00010
.db &b00010
.db &b00100
.db &b11000
enddata
```