```
;*****************************************************************************************
; R2R DDS Generator-Firmware für Platine TN20140318 Fusebits: Low, Ext, Lock=FFH, High=DFH
; benutzt 32 bit Phasenakku bei 20MHz Takt. Initiale Version adaptiert von Jesper Hansen
;
; V1.00: 28/03/2014 - Initiale Version, T. Neveling migriert auf AVR-Studio 4.17
; V1.01: 21/04/2014 - Startfrequenz wie beim Control-uC 1kHz statt bisher 400Hz
; V1.02: 31/01/2015 - Auf 250000 geänderte Baudrate ohne Verwendung des x2 Bits
; V2.00: 03/02/2015 - Rechteck & Sägezahn werden direkt berechnet (ohne Tabelle)
;                     Variable Zyklenzahl je nach Algorithmus, definiert im Control uC
;                     Dutycycle bei Rechteck variabel, Bezeichner für alle Register
;*****************************************************************************************
; PB0..7 =  DA Wandler Data out
; PD0    =  RXD
;
; Die Ausgangsfrequenz mit 32 Bit Akku ist: f = deltaPhase * fClock/2^32
; fClock = CPU clock [20MHz] / Zahl der CPU-Takte zur Frequenzberechnung = Abtastfrequenz
; diese variiert je nach Kurvenform und wird vom Control uC entsprechend berechnet
; Für die Erzeugung eines 100kHz-Signals bei fClock = 2MHz stehen 20 Samples zur Verfügung
; fMax (theoretisch) = 0.5 * fClock, z.B. bei 10 CPU-Zyklen = 1MHz
;
.include "tn4313def.inc"            ; für ATtiny4313
;
.equ XTAL       = 20000000          ; Systemclock [Hz]
.equ BAUD       = 250000            ; USART Baudrate
.equ UBRRL_VAL  = (XTAL/(BAUD*16)-1); Wert hierzu = 4, Doublespeed-Bit NICHT gesetzt!
;
.def temp       = r16               ; Universalregister
.def Wave       = r17               ; Kurvenform: 1=Sinus...7=Rechteck fast
.def NULL       = r18               ; immer = 0
.def FULL       = r19               ; immer = 255
.def DUTY       = r20               ; Tastverhältnis Rechteck
;
.def Adder1     = r22               ; 4 Byte Addervalue
.def Adder2     = r23
.def Adder3     = r24
.def Adder4     = r25
;
.def Phase1     = r27               ; 4 Byte Phasenakku
.def Phase2     = r28               ; Warnung von AVR Studio über Doppelbelegung ignorieren!
.def Phase3     = r29
.def Phase4     = r30
;
.org 0  rjmp reset
.org 7  rjmp rx6                    ; RX complete interrupt, empfängt jeweils 6 Byte
;
;*************************************************************
; Signaltabellen: Fortlaufend 256 Werte über 1 volle Periode
;*************************************************************
.org 0x100  ; Start auf einer durch 256 teilbaren Adresse, dann ist ZL=r30 immer =0
;
sine:       ; Sinustabelle
.db $80, $83, $86, $89, $8C, $90, $93, $96, $99, $9C, $9F, $A2, $A5, $A8, $AB, $AE
.db $B1, $B3, $B6, $B9, $BC, $BF, $C1, $C4, $C7, $C9, $CC, $CE, $D1, $D3, $D5, $D8
.db $DA, $DC, $DE, $E0, $E2, $E4, $E6, $E8, $EA, $EB, $ED, $EF, $F0, $F1, $F3, $F4
.db $F5, $F6, $F8, $F9, $FA, $FA, $FB, $FC, $FD, $FD, $FE, $FE, $FE, $FF, $FF, $FF
.db $FF, $FF, $FF, $FF, $FE, $FE, $FE, $FD, $FD, $FC, $FB, $FA, $FA, $F9, $F8, $F6
.db $F5, $F4, $F3, $F1, $F0, $EF, $ED, $EB, $EA, $E8, $E6, $E4, $E2, $E0, $DE, $DC
.db $DA, $D8, $D5, $D3, $D1, $CE, $CC, $C9, $C7, $C4, $C1, $BF, $BC, $B9, $B6, $B3
.db $B1, $AE, $AB, $A8, $A5, $A2, $9F, $9C, $99, $96, $93, $90, $8C, $89, $86, $83
.db $80, $7D, $7A, $77, $74, $70, $6D, $6A, $67, $64, $61, $5E, $5B, $58, $55, $52
.db $4F, $4D, $4A, $47, $44, $41, $3F, $3C, $39, $37, $34, $32, $2F, $2D, $2B, $28
.db $26, $24, $22, $20, $1E, $1C, $1A, $18, $16, $15, $13, $11, $10, $0F, $0D, $0C
.db $0B, $0A, $08, $07, $06, $06, $05, $04, $03, $03, $02, $02, $02, $01, $01, $01
.db $01, $01, $01, $01, $02, $02, $02, $03, $03, $04, $05, $06, $06, $07, $08, $0A
.db $0B, $0C, $0D, $0F, $10, $11, $13, $15, $16, $18, $1A, $1C, $1E, $20, $22, $24
.db $26, $28, $2B, $2D, $2F, $32, $34, $37, $39, $3C, $3F, $41, $44, $47, $4A, $4D
.db $4F, $52, $55, $58, $5B, $5E, $61, $64, $67, $6A, $6D, $70, $74, $77, $7A, $7D
;
triangle:   ; Dreiecktabelle
.db $00, $02, $04, $06, $08, $0A, $0C, $0E, $10, $12, $14, $16, $18, $1A, $1C, $1E
.db $20, $22, $24, $26, $28, $2A, $2C, $2E, $30, $32, $34, $36, $38, $3A, $3C, $3E
.db $40, $42, $44, $46, $48, $4A, $4C, $4E, $50, $52, $54, $56, $58, $5A, $5C, $5E
.db $60, $62, $64, $66, $68, $6A, $6C, $6E, $70, $72, $74, $76, $78, $7A, $7C, $7E
.db $80, $82, $84, $86, $88, $8A, $8C, $8E, $90, $92, $94, $96, $98, $9A, $9C, $9E
.db $A0, $A2, $A4, $A6, $A8, $AA, $AC, $AE, $B0, $B2, $B4, $B6, $B8, $BA, $BC, $BE
.db $C0, $C2, $C4, $C6, $C8, $CA, $CC, $CE, $D0, $D2, $D4, $D6, $D8, $DA, $DC, $DE
.db $E0, $E2, $E4, $E6, $E8, $EA, $EC, $EE, $F0, $F2, $F4, $F6, $F8, $FA, $FC, $FE
.db $FF, $FE, $FC, $FA, $F8, $F6, $F4, $F2, $F0, $EE, $EC, $EA, $E8, $E6, $E4, $E2
.db $E0, $DE, $DC, $DA, $D8, $D6, $D4, $D2, $D0, $CE, $CC, $CA, $C8, $C6, $C4, $C2
.db $C0, $BE, $BC, $BA, $B8, $B6, $B4, $B2, $B0, $AE, $AC, $AA, $A8, $A6, $A4, $A2
.db $A0, $9E, $9C, $9A, $98, $96, $94, $92, $90, $8E, $8C, $8A, $88, $86, $84, $82
.db $80, $7E, $7C, $7A, $78, $76, $74, $72, $70, $6E, $6C, $6A, $68, $66, $64, $62
.db $60, $5E, $5C, $5A, $58, $56, $54, $52, $50, $4E, $4C, $4A, $48, $46, $44, $42
.db $40, $3E, $3C, $3A, $38, $36, $34, $32, $30, $2E, $2C, $2A, $28, $26, $24, $22
.db $20, $1E, $1C, $1A, $18, $16, $14, $12, $10, $0E, $0C, $0A, $08, $06, $04, $02
;
```

```
EKG:        ; "EKG"-Tabelle
.db $49, $4A, $4B, $4B, $4A, $49, $49, $49, $49, $48, $47, $45, $44, $43, $43, $43
.db $44, $44, $43, $41, $3E, $3D, $3B, $39, $38, $37, $37, $36, $36, $36, $37, $37
.db $37, $37, $37, $37, $36, $35, $33, $32, $31, $31, $34, $3D, $4D, $65, $84, $A9
.db $CF, $EE, $FF, $FE, $EA, $C6, $9A, $6D, $44, $25, $11, $05, $00, $01, $06, $0D
.db $14, $1C, $24, $2D, $34, $39, $3D, $40, $41, $42, $43, $44, $44, $45, $46, $47
.db $47, $47, $47, $47, $47, $47, $47, $48, $48, $48, $49, $49, $4A, $4B, $4B, $4C
.db $4D, $4E, $4F, $50, $51, $52, $53, $54, $56, $58, $5B, $5D, $60, $62, $64, $66
.db $68, $6B, $6D, $70, $73, $76, $79, $7B, $7D, $7E, $7F, $7F, $7F, $7F, $7F, $7E
.db $7D, $7C, $79, $77, $74, $71, $6D, $69, $66, $62, $5F, $5C, $59, $57, $54, $51
.db $4F, $4D, $4C, $4B, $4A, $49, $48, $46, $45, $44, $43, $43, $43, $44, $44, $44
.db $45, $45, $45, $45, $45, $45, $45, $46, $47, $48, $49, $49, $4A, $4A, $4B, $4B
.db $4B, $4B, $4B, $4B, $4A, $4A, $49, $49, $49, $49, $48, $48, $48, $47, $47, $47
.db $47, $47, $47, $47, $46, $46, $46, $45, $45, $45, $45, $45, $46, $46, $46, $45
.db $44, $44, $43, $43, $43, $43, $42, $42, $42, $41, $41, $41, $41, $41, $41, $41
.db $41, $40, $40, $3F, $3F, $40, $40, $41, $41, $41, $41, $41, $41, $41, $40, $40
.db $40, $40, $40, $40, $40, $40, $41, $41, $41, $42, $43, $44, $45, $47, $48, $49
;
reset:  ldi temp, LOW (RAMEND)  ; Stack pointer initialisieren
        out SPL, temp
;
        ldi temp,HIGH(UBRRL_VAL); Baudrate = 250000Bd
        out UBRRH, temp
        ldi temp,LOW(UBRRL_VAL)
        out UBRRL, temp
;
        ldi temp,6              ; Datenprotokoll: 8N1, eigentlich schon vorbesetzt
        out UCSRC,temp
        ldi temp,0b10010000     ; enable Receiver & RXint
        out UCSRB,temp
;
        sei                     ; global enable interrupts
        ser temp
        out DDRB,temp           ; PORTB: Alle Bits auf Ausgabe für R2R-DA-Wandlung
;
        ser FULL
        clr NULL
        ldi DUTY, 128           ; Symmetrische Rechteckimpulse (Bereich ist 0..255)
        ldi Wave, 1             ; mit Sinus starten
;
; PhaseAkku löschen
        ldi Phase1,0
        ldi Phase2,0
        ldi Phase3,0
        ldi Phase4,0
;
; AdderValue für 1kHz Startfrequenz (10 Zyklen) initialisieren
        ldi Adder1,byte1(2147484)   ; = 1000Hz * 2^32 / 2MHz
        ldi Adder2,byte2(2147484)
        ldi Adder3,byte3(2147484)
        ldi Adder4,byte4(2147484)
        rjmp main
;
; Hier werden alle Tabellen-basierten Signale erzeugt:
; 1.) Adderwert zum PhasenAkku addieren
; 2.) zugehöriges Byte aus der gewählten Tabelle holen
; 3.) dieses auf PortB ausgeben
; 4.) Wiederholen bis Transferflag von RX6 gesetzt wird (neuer Job)
;
tables: CLT                 ; Transfer-Flag löschen: Job gestartet
tab lp: add Phase1,Adder1   ; 1
        add Phase2,Adder2   ; 1
        adc Phase3,Adder3   ; 1
        adc Phase4,Adder4   ; 1
        lpm                 ; 3
        out PortB,r0        ; 1
        brtc tab_lp         ; 2 => Summe = 10 CPU-Zyklen: Sinus, Dreieck, EKG
;
; Main-Loop: Hier wird nach dem Empfang eines neuen Jobs via RS232
; je nach Kurvenform zur entsprechenden Adder-Loop verzweigt.
main:   cpi wave, 1         ; Sinus ?
        brne m1
        ldi ZH,high(sine*2)
        ldi ZL,low(sine*2)
        rjmp tables
m1:     cpi wave, 2         ; Rechteck ?
        brne m2
        rjmp Sqr
m2:     cpi wave, 3         ; Dreieck ?
        brne m3
        ldi ZH,high(triangle*2)
        ldi ZL,low(triangle*2)
        rjmp tables
m3:     cpi wave, 4         ; Sägezahn ?
        brne m4
        rjmp saw
```

```
m4:     cpi wave, 5         ; Sägezahn invers ?
        brne m5
        rjmp SawR
m5:     cpi wave, 6         ; EKG ?
        brne m6
        ldi ZH,high(EKG*2)
        ldi ZL,low(EKG*2)
        rjmp tables
m6:     cpi wave, 7         ; Rechteck fast ?
        brne main
;
; Schnellers Rechteck ohne Tabelle mit Pin-Toggle, Duty fix: 9 CPU-Zyklen
        CLT                 ; Transfer-Flag löschen: Job gestartet
        Out PortB, FULL     ; Port für nachfolgenden Pin-Toggle Modus init.
Sqf_lp: BRTS main           ; 1
        add Phase1,Adder1   ; 1
        add Phase2,Adder2   ; 1
        adc Phase3,Adder3   ; 1
        adc Phase4,Adder4   ; 1
        brcc Sqf1           ; 1/2
        out PinB, FULL      ; 1/0 Pin-Toggle Modus
Sqf1:   rjmp Sqf_lp         ; 2
;
; Rechteck-Signal ohne Tabelle mit variablem Tastverhältnis: 11 CPU-Zyklen
Sqr:    CLT                 ; <Duty >=Duty
Sqr lp: BRTS main           ;   1       1
        add Phase1,Adder1   ;   1       1
        add Phase2,Adder2   ;   1       1
        adc Phase3,Adder3   ;   1       1
        adc Phase4,Adder4   ;   1       1
        cp  Phase4,DUTY     ;   1       1
        brlo Sqr1           ;   2       1
        nop                 ;   0       1
        out PortB, NULL     ;   0       1
        rjmp Sqr lp         ;   0       2
Sqr1:   out PortB, FULL     ;   1       0
        rjmp Sqr_lp         ;   2       0
;
; Sägezahn ohne Tabelle: 8 CPU-Zyklen
Saw:    CLT                 ; Transfer-Flag löschen: Job gestartet
Saw_lp: BRTS main           ; 1
        add Phase1,Adder1   ; 1
        add Phase2,Adder2   ; 1
        adc Phase3,Adder3   ; 1
        adc Phase4,Adder4   ; 1
        out PortB, Phase4   ; 1
        rjmp Saw_lp         ; 2
;
; Sägezahn-invers ohne Tabelle: 8 CPU-Zyklen
SawR:   CLT                 ; Transfer-Flag löschen: Job gestartet
SawR_lp:BRTS main           ; 1
        sub Phase1,Adder1   ; 1
        sbc Phase2,Adder2   ; 1
        sbc Phase3,Adder3   ; 1
        sbc Phase4,Adder4   ; 1
        out PortB, Phase4   ; 1
        rjmp SawR_lp        ; 2
;
;*********************************************************************
; RS232-Empfangs-ISR mit 250 kBaud = 4us/Bit. Interrupt beim 1. Byte,
; die restlichen per Polling 6x9 Bit (1 Stoppbit) = 54 Bits dauern 216us
;*********************************************************************
rx6:    cli                 ; Interrupt vorübergehend sperren
        in Wave,UDR         ; 1. Byte per Interrupt = Kurvenform
        rcall get char      ; 2. Byte ist Dutycycle für Rechteck
        mov Duty,temp
        rcall get_char
        mov Adder1,temp     ; 32 Bit für den Phasenaccu Adderwert
        rcall get char
        mov Adder2,temp
        rcall get char
        mov Adder3,temp
        rcall get char
        mov Adder4,temp
        cpi Wave,8          ; erlaubt: 1 = Sinus bis 7 = Sqfast
        BRLO w_ok
        ldi Wave,1          ; unerlaubter Wert => default Sinus setzen
w ok:   SET                 ; Transfer-Flag setzen => neuer Job!
        sei
        reti
get char:
        sbis ucsra,rxc
        rjmp get_char
        in temp,UDR
        ret
```