D1 DDS

Kurze Einführung

Bei der Direkten Digitalen Synthese (*Direct Digital Synthesis*) wird ein periodisches bandbegrenztes, analoges Signal (zum Beispiel ein Sinussignal) mit Hilfe eines D/A-Wandlers oder einer PWM aus einem zeitlich änderndem digitalen Signal erzeugt.

Der große Vorteil des DDS-Verfahrens ist die praktisch beliebig feine Frequenzauflösung. Weitere Vorteile sind das schnelle Umschalten zwischen den Frequenzen, die große erreichbare Bandbreite und hohe Frequenzstabilität.

Die weite Verbreitung von ICs, die die komplette Hardware eines Synthesizers nach dem DDS-Verfahren realisieren, hat wesentlich zum Erfolg des Verfahrens beigetragen.

Warum DDS?

Die Berechnung zum Beispiel einer Sinusschwingung (8 Bit: $u=255*\sin(\alpha)$) mit dem Mikrocontroller würde viel zu lange dauern. Es ist einfacher die Amplitudenwerte einer Schwingung in einer Tabelle abzulegen und dann auf diese zuzugreifen. Die Frequenz ist dann allerdings durch die Zugriffsgeschwindigkeit des Programms festgelegt.

Beispiel: Sinustabelle mit 256 Werten, Timerinterrupt, der die Routine zur Ausgabe der Werte mit 1 MHz aufruft: Maximale Frequenz: 1 MHz/256 = 3,9 kHz. Die Änderung der Frequenz ist nur durch die Änderung der Timerinterrupt-Frequenz möglich.

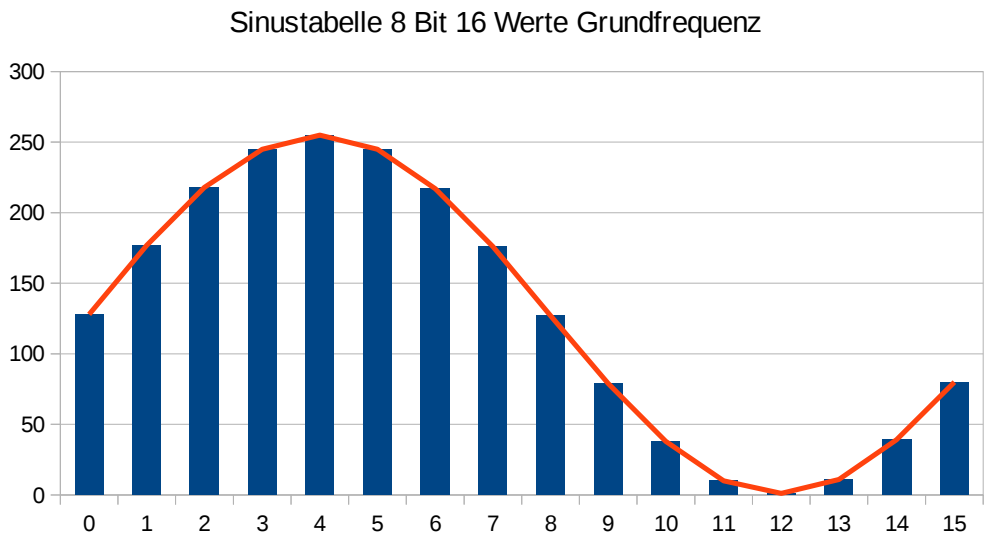
Mit einem Trick lassen sich auch höhere Frequenzen ausgeben. Gibt man nur jeden zweiten Tabellenwert aus, so verdoppelt sich die Frequenz. Bei jedem dritten Wert verdreifacht die Frequenz usw..

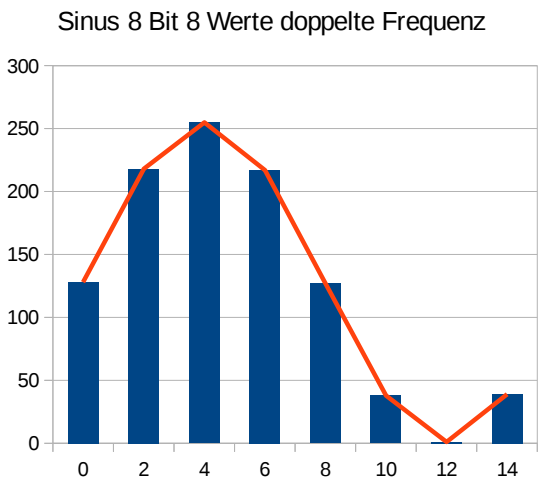
Pro Periode müssen mindestens noch 2 Werte übertragen werden (Nyquist-Kriterium). Man ist also auch nach oben hin in der Frequenz begrenzt.

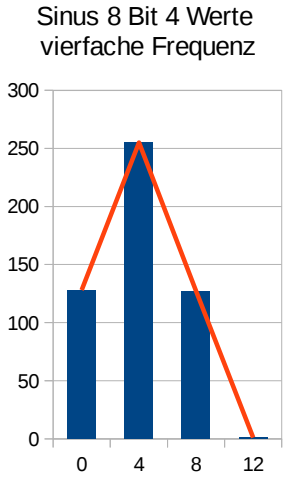
Beispiel: Sinustabelle mit 256 Werten, Timerinterrupt, der die Routine zur Ausgabe der Werte mit 36,363 kHz aufruft:

Grundfrequenz:	36,36 kHz/256 =	140 Hz
Jeder zweite Wert:	36,36 kHz/128 =	280 Hz
Jeder dritte Wert:	36,36 kHz/85 =	426 Hz
Jeder vierte Wert:	36,36 kHz/64 =	568 Hz
...		
Maximale Frequenz:	36,36 kHz/2 =	18,18k Hz.

Beispiel: Tabelle (Wavetable, B = 8 Bit, Amplitude 0-255) mit 16 Werten (Adresszeiger P = 4 Bit):







Nachteilig ist die bei diesem Verfahren die immer gröbere Abstufung des Signals. Außerdem sind so nur ganzzahlige Vielfache der Grundfrequenz erreichbar.

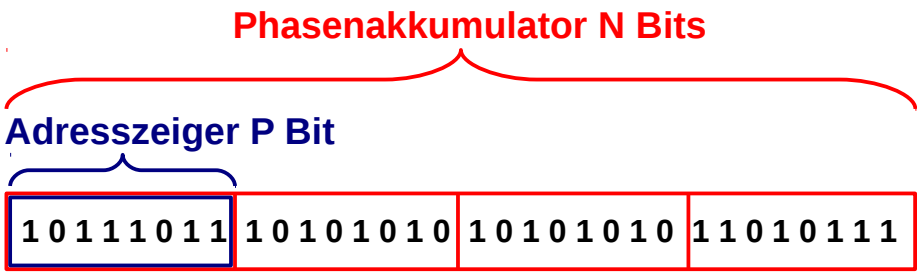
Mit Hilfe der DDS lassen sich diese beiden Nachteile umgehen.

DDS mit dem Mikrocontroller

Statt eines einfachen binären Zählers, der die Tabellenwerte der Reihe nach aufruft, verwenden wir einen sogenannter Phasenakkumulator mit N Bit. Dieses Register ist viel größer als zum Adressieren der Tabelle notwendig wäre. Meist werden Register mit 4 oder 6 Byte (N = 32 bzw. 48 Bit) verwendet. Nur die oberen P-Bit (zum Beispiel 8 Bit für eine

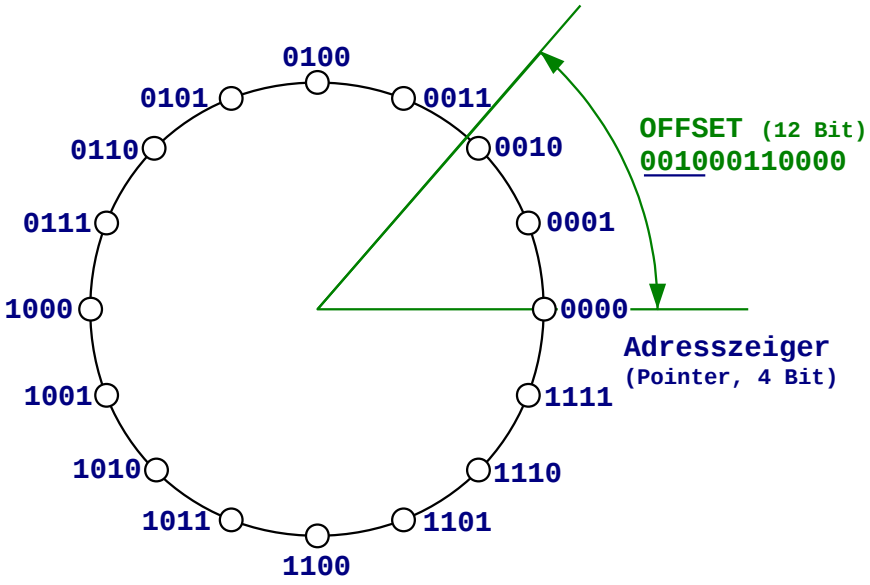
Tabelle mit 256 Werten) des Phasenregister werden dann zur Adressierung der Tabelle verwendet.

Beispiel: Phasenakku mit 4 Byte

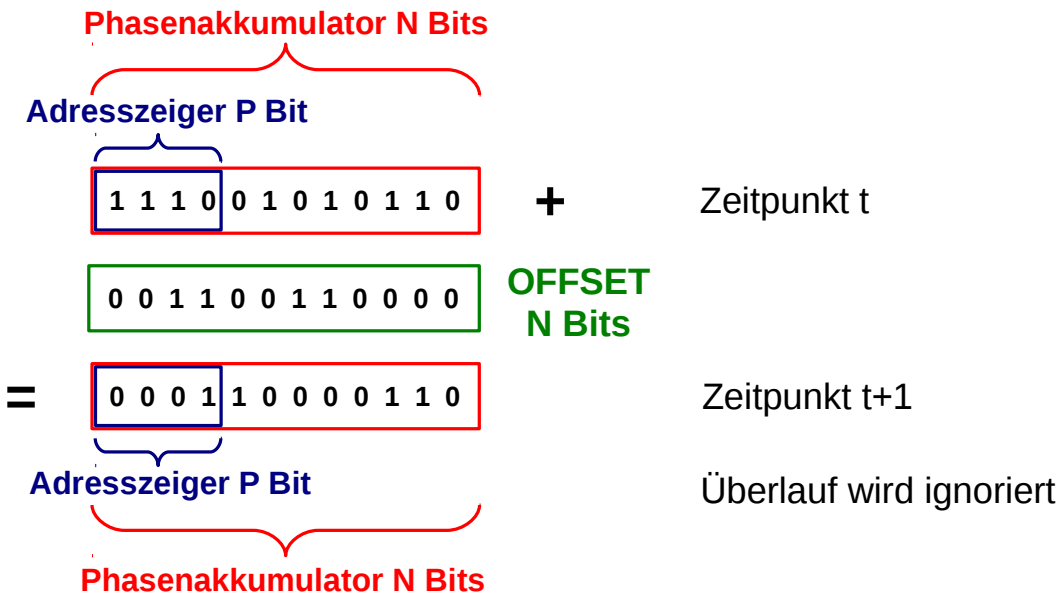


Das Phasenregister wird um einen festen Phasenoffset (Phaseninkrement), hier Offset genannt erhöht. Einsteht dabei ein Überlauf, so wird dieser ignoriert. Dieses Phasenoffset-Register hat dieselbe Größe wie das Phasenregister.

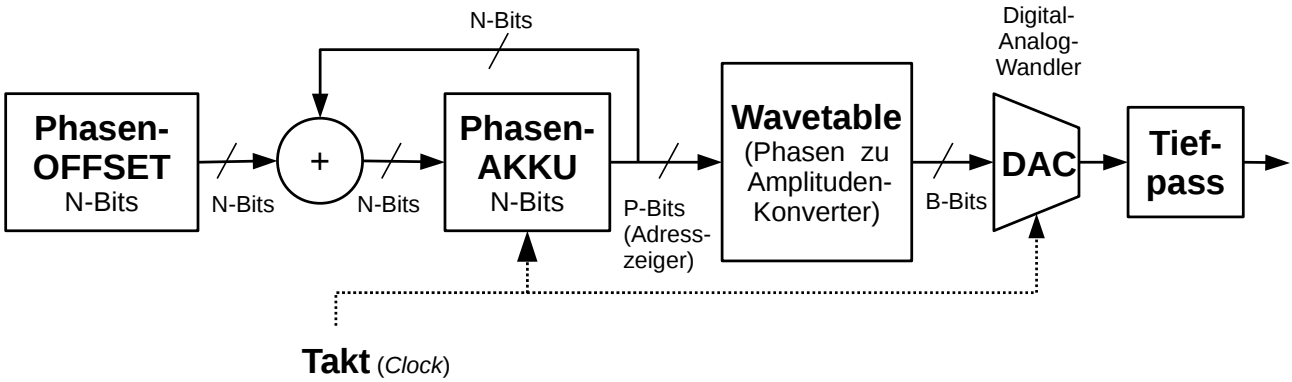
Für die folgenden Beispiele wird ein Phasenakku und ein Offset mit je N = 12 Bit verwendet. Der Adresszeiger hat P = 4 Bit um eine Tabelle mit 16 Werten zu adressieren. Die Amplitude beträgt B = 8 Bit (0-255).



Jetzt sind nicht mehr ganzzahlige Vielfache der Frequenz möglich, sondern auch Zwischenwerte (Der Effekt ist wie wenn man mit Kommastellen arbeiten würde, also zum Beispiel auch jeden 1,3ten Wert nehmen könnte).



Blockschaltbild der DDS:

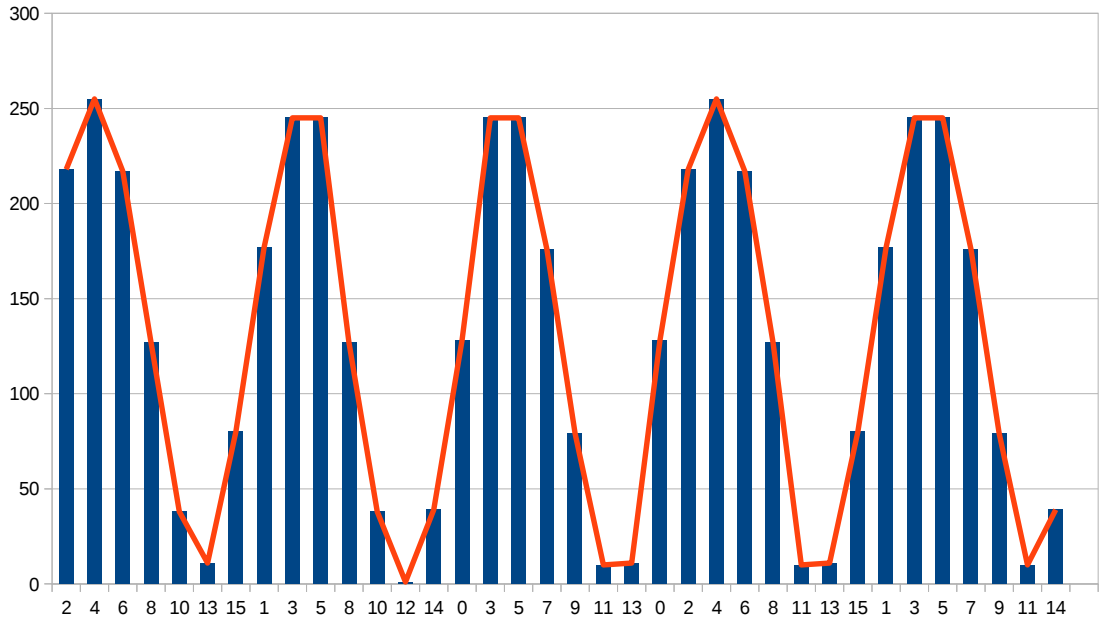


Beispiel: N = 12 Bit → Phasenakku und Offset haben 12 Bit-Breite
Für den Adresszeiger (*Pointer*) werden nur die oberen 4 Bit verwendet (P = 4Bit)
Der Überlauf bei der Addition wird nicht berücksichtigt (Modulo 2^N).

	Phasenakku		Offset	Adresszeiger
	000000000000	+	001000110000	0000 = 0
=	001000110000	+	001000110000	0010 = 2
=	010001100000	+	001000110000	0100 = 4
=	011010010000	+	001000110000	0110 = 6

=	1000 11000000	+	001000110000	1000 = 8
=	1010 11110000	+	001000110000	1010 = 10
=	1101 00100000	+	001000110000	1101 = 13
=	1111 01010000	+	001000110000	1111 = 15
=	0001 10000000	+	001000110000	0001 = 1
=	0011 10110000	+	001000110000	0011 = 3
=	0101 11100000	+	001000110000	0101 = 5
=	1000 00010000	+	001000110000	1000 = 8

usw.



Die Frequenz Ausgangsfrequenz f_{out} ergibt sich durch folgende Formel:

$$f_{out} = \frac{Offset \cdot f_{CLK}}{2^N}$$

Die Auflösung beträgt dabei: $f_{CLK}/2^N$.

f_{CLK} ist die Frequenz mit der die Werte an den D/A-Wandler bzw. mit der PWM ausgegeben werden. Um sie zu erzeugen kann man zum Beispiel einen Timerinterrupt verwenden.
Die Formel lässt erkennen, dass man umso höhere Frequenzen erzeugen kann, je höher die Frequenz f_{CLK} ist. Ist sie beim Mikrocontroller jedoch zu hoch, bleibt dem Hauptprogramm keine Zeit mehr. Beim AVR-Syntesizer meeblib (meeblip.com) wurde zum Beispiel eine Frequenz von 36,36 kHz gewählt. So verbleiben beim verwendeten 16 MHz Quarz rund 440 Taktzyklen zum Bearbeiten der Daten (ISR + Hauptprogramm).

Je höher die zu erzeugende Frequenz f_{OUT} im Vergleich zu f_{CLK} wird, umso mehr machen sich „Unschönheiten“ des DDS-Prinzips durch Jitter, Rauschen und Nebenwellen im Spektrum bemerkbar. Mit einer Taktrate von 36,36 kHz kommt man nicht höher als rund 3 kHz in der Programmiersprache C. Mit Assembler kann man durch effizientere Programmierung viel Zeit einsparen und so höherer Frequenzen erreichen.

Beispiel: Nehmen wir in unserem obigen Beispiel eine Taktfrequenz f_{CLK} von 36363 Hz an, so erhalten wir:

$$f_{out} = \frac{OFFSET \cdot f_{CLK}}{2^N} = \frac{560 \cdot 36363}{4096} = 4971{,}5\,Hz \qquad (0b0010001100 = 560)$$

Die kleinste Frequenz (Grundfrequenz der Tabelle, alle 16 Werte) ohne DDS ergibt sich mit $f_{OUTGF} = f_{CLK}/16 = 2272{,}7$ Hz.

Die Frequenz wurde also mit diesem Offset um den Faktor $f_{OUT}/f_{OUTGF} = 4971{,}5/2272{,}7 = 2{,}187$ erhöht.
Wird der Offset um 1 erhöht, so steigt die Frequenz auf 4980,38Hz. Die Differenz von 8,878 Hz errechnet sich aus $f_{CLK}/2^N$.

Die Grundfrequenz wird übrigens mit dem Offset 0b000100000000 = 256 erreicht. Es sind mit der DDS auch kleinere Frequenzen als die Grundfrequenz möglich!

Hier einige Beispiele aus denen man ersieht wie sich die Auflösung mit Hilfe der Verbreiterung des Phasenakku erhöhen lässt:

Breite des Phasenakku N	12 Bit	16 Bit	16 Bit	24 Bit	32 Bit
Taktfrequenz f_{CLK}	36363 Hz	36363 Hz	100kHz	36363 Hz	36363 Hz
maximaler Offset	4096	65636	65636	16777216	4294967296
OFFSET	700	11200	11200	2867200	734003200
Ausgangsfrequenz f_{OUT}	6,21 kHz	6,21 kHz	17,089 kHz	6,21 kHz	6,21 kHz
Auflösung	8,877Hz	0,554 Hz	1,525 Hz	0,002167 Hz	8,466μHz!

Der Offset für eine beliebige Frequenz errechnet sich aus:

$$Offset = \frac{f_{out} \cdot 2^N}{f_{CLK}}$$

Assembler-Beispielcode für ein Programm, das ein Sinussignal mit 1 kHz ausgibt:

F_{CLK} = 36363 Hz, N = 24 Bit, Offset = $1000Hz*2^{24}/36363Hz$ = 461373 = 0x070A3D

```
;*****************************************************************************
;*
;*      Titel:  D1_dds_1kHz.asm (DDS feste Frequenz)
;*      Datum:  05/07/14        Version: 0.2 (18/12/14)
;*      Autor:  WEIGU
;*
;*      Informationen zur Beschaltung:
;*
;*      Prozessor:      ATmega32A           Quarzfrequenz: 16MHz
;*      Eingaenge:
;*      Ausgaenge:      PORTB verbunden mit einem R2R DAC +
;*                      RC-Tiefpass 47nF, 1,2k
;*
;*      Informationen zur Funktionsweise:
;*
;*      Frequenzgenerator mit DDS (f = 1kHz)
;*
;*      Eine Interruptroutine wird mit 36364 Hz aufgerufen (Routine kurz halten
;*      da nur 440 Taktzyklen fuer Routine + Hauptprogramm zur Verfuegung
;*      stehen).
;*      Innerhalb dieser Routine wird ein Sinussignalsignal mit Hilfe der
;*      direkten Frequenz Synthese (DDS = Direct Digital Synthesis) erzeugt.
;*      Dazu wird ein 24-Bit Phasenakku mit einem Wert inkrementiert (OFFSET)
;*      der proportional zur erwuenschten Frequenz ist. Die oberen 8 Bit des
;*      Adresszaehler bilden dann die Adresse fuer eine eine ROM Tabelle mit 256
;*      Werten (Lockup-Tabelle, Wavetable).
;*
;*      OFFSET = 2 ^ 24 * Freq / SamplingFreq (36364Hz)
;*      OFFSET = 461,37344*f
;*      fuer f = 1kHz: OFFSET = 461373 = 0x070A3D
;*
;*****************************************************************************

;-----------------------------------------------------------------------------
;       Einbinden der controllerspezifischen Definitionsdatei
;-----------------------------------------------------------------------------
.NOLIST                         ;List-Output ausschalten
.INCLUDE "m32def.inc"           ;AVR-Definitionsdatei einbinden
.LIST                           ;List-Output wieder einschalten

;-----------------------------------------------------------------------------
;       Organisation des Datenspeichers (SRAM)
;-----------------------------------------------------------------------------
.DSEG                          ;was ab hier folgt kommt in den SRAM-Speicher

; Phaseninkrement OFFSET des DCO
OFFSET0: .byte 1
OFFSET1: .byte 1
OFFSET2: .byte 1

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;       Programmspeicher (FLASH)     Programmstart nach RESET ab Adr. 0x0000
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.CSEG                           ;was ab hier folgt kommt in den FLASH-Speicher
.ORG    0x0000                  ;Programm beginnt an der FLASH-Adresse 0x0000
RESET1: rjmp    INIT            ;springe nach INIT (ueberspringe ISR Vektoren)

;-----------------------------------------------------------------------------
;       Sprungadressen fuer die Interrupts organisieren (ISR VECTORS)
;-----------------------------------------------------------------------------
        ;Vektortabelle (im Flash-Speicher)
.ORG    OC2addr                 ;Timer2 Compare
        rjmp    ISRTC

;-----------------------------------------------------------------------------
;       Initialisierungen und eigene Definitionen
;-----------------------------------------------------------------------------
.ORG    INT_VECTORS_SIZE        ;Platz fuer ISR Vektoren lassen
```

```
INIT:

.DEF    PHASE0 = r2             ;Phasenakku des DCO
.DEF    PHASE1 = r3
.DEF    PHASE2 = r4
.DEF    Zero    = r15           ;Register 1 wird zum Rechnen benoetigt
        clr     r15             ;und mit Null belegt
.DEF    Tmp1  = r16             ;Register 16 dient als erster Zwischenspeicher

        ;Stapel initialisieren (fuer Unterprogramme bzw. Interrupts)
        ldi     Tmp1,LOW(RAMEND) ;RAMEND (SRAM) ist in der Definitions-
        out     SPL,Tmp1         ;datei festgelegt
        ldi     Tmp1,HIGH(RAMEND)
        out     SPH,Tmp1

        ;Initialisiere den Phasenakku
        clr     PHASE0
        clr     PHASE1
        clr     PHASE2

        ;OFFSET fuer 1 kHz initialisieren (0x070A3D)
        ldi     Tmp1,0x3D
        sts     OFFSET0,Tmp1
        ldi     Tmp1,0x0A
        sts     OFFSET1,Tmp1
        ldi     Tmp1,0x07
        sts     OFFSET2,Tmp1

        ;Initialisiere Timer2 (CTC Interrupt mit 36,36 KHz):
        ldi     r16, 54         ;OCR2 = 54 gives 16MHz/8/55 = 36363.63636 Hz
        out     OCR2, r16
        ldi     r16, 0x0A       ;CTC mode (OC2 off), prescaler = CK/8
        out     TCCR2, r16
        ldi     r16, 0x80       ;OCIE2=1
        out     TIMSK, r16

        ser     Tmp1            ;PortC Ausgang (R2R DAC)
        out     DDRB,Tmp1
        sei                     ;Interrupts global erlauben
;-----------------------------------------------------------------------------
;       Hauptprogramm
;-----------------------------------------------------------------------------
MAIN:   rjmp    MAIN    ;Endlosschleife

;-----------------------------------------------------------------------------
;       Unterprogramme und Interrupt-Behandlungsroutinen
;-----------------------------------------------------------------------------
ISRTC:  push    r16
        in      r16,SREG
        push    r16
        push    ZL
        push    ZH

        ldi     ZL,LOW(SINET*2)         ;Adressiere Wavetable
        ldi     ZH,HIGH(SINET*2)
        add     ZL,PHASE2
        adc     ZH,Zero
        lpm     r16,Z
        out     PORTB,r16               ;Ausgabe Sample (Wavetable)

        ;Erhoehe die Phase des DCO um Delta
        lds     r16,OFFSET0
        add     PHASE0,r16
        lds     r16,OFFSET1
        adc     PHASE1,r16
        lds     r16,OFFSET2
        adc     PHASE2,r16

ISRTCR: pop     ZH
        pop     ZL
        pop     r16
```

```
        out     SREG,r16
        pop     r16
        reti

;-----------------------------------------------------------------------------
;       Tabellen im Programmspeicher (Flash)
;-----------------------------------------------------------------------------
;Sinustabelle mit 256 Werten
SINET:  .DB     128, 131, 134, 137, 140, 143, 146, 149
        .DB     152, 155, 158, 162, 165, 167, 170, 173
        .DB     176, 179, 182, 185, 188, 190, 193, 196
        .DB     198, 201, 203, 206, 208, 211, 213, 215
        .DB     218, 220, 222, 224, 226, 228, 230, 232
        .DB     234, 235, 237, 238, 240, 241, 243, 244
        .DB     245, 246, 248, 249, 250, 250, 251, 252
        .DB     253, 253, 254, 254, 254, 255, 255, 255
        .DB     255, 255, 255, 255, 254, 254, 254, 253
        .DB     253, 252, 251, 250, 250, 249, 248, 246
        .DB     245, 244, 243, 241, 240, 238, 237, 235
        .DB     234, 232, 230, 228, 226, 224, 222, 220
        .DB     218, 215, 213, 211, 208, 206, 203, 201
        .DB     198, 196, 193, 190, 188, 185, 182, 179
        .DB     176, 173, 170, 167, 165, 162, 158, 155
        .DB     152, 149, 146, 143, 140, 137, 134, 131
        .DB     128, 124, 121, 118, 115, 112, 109, 106
        .DB     103, 100, 97, 93, 90, 88, 85, 82
        .DB     79, 76, 73, 70, 67, 65, 62, 59
        .DB     57, 54, 52, 49, 47, 44, 42, 40
        .DB     37, 35, 33, 31, 29, 27, 25, 23
        .DB     21, 20, 18, 17, 15, 14, 12, 11
        .DB     10, 9, 7, 6, 5, 5, 4, 3
        .DB     2, 2, 1, 1, 1, 0, 0, 0
        .DB     0, 0, 0, 0, 1, 1, 1, 2
        .DB     2, 3, 4, 5, 5, 6, 7, 9
        .DB     10, 11, 12, 14, 15, 17, 18, 20
        .DB     21, 23, 25, 27, 29, 31, 33, 35
        .DB     37, 40, 42, 44, 47, 49, 52, 54
        .DB     57, 59, 62, 65, 67, 70, 73, 76
        .DB     79, 82, 85, 88, 90, 93, 97, 100
        .DB     103, 106, 109, 112, 115, 118, 121, 124

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.EXIT                           ;Ende des Quelltextes
```

Bascom-Beispielcode von Jean Daubenfeld für ein Programm, das ein Sinussignal mit 1 kHz ausgibt:

FCLK = 36363 Hz, N = 32 Bit, Offset = 1000Hz*2^{32}/36363Hz = 118111601 = 0x070A3D71

!Das Bascom Programm arbeitet mit 4 Byte (32 Bit).

```
' 1kHz Sinus DDS (PortB verbunden mit R2R DAC + RC-Tiefpass 47nF, 1,2k
' Offset = fout*2^N/fCLK = 1000Hz*2^32/36363Hz = 118111601 = &H070A3D71
' Jean Daubenfeld / WEIGU

$regfile = "m32def.dat"
$crystal = 16000000

$hwstack = 50
$swstack = 50
$framesize = 60

Config Portb = Output


Enable Timer2
Config Timer2 = Timer , Prescale = 8 , Clear_timer = 1
On Compare2 Tim2_isr
Enable Compare2
Compare2 = 54                    'OCR2 = 54 gives 16MHz/8/55 = 36,36kHz
Enable Interrupts

Dim Offset As Dword
Dim Phasenakku As Dword
Offset = &H070A3D71
Phasenakku = &H00000000

Dim Adresszeiger As Dword
Dim Sin_wert As Byte

'------------            Hauptprogramm:          ------------------------------
Do
Loop
End

'------------            Unterprogramme:          ------------------------------
Tim2_isr:
   Phasenakku = Phasenakku + Offset
   Adresszeiger = Phasenakku
   Shift Adresszeiger , Right , 24
Sin_wert = Lookup(Adresszeiger , Sinus)
Portb = Sin_wert
Return

'------------            Sinustabelle:          ------------------------------
Sinus:
Data 128 , 131 , 134 , 137 , 140 , 143 , 146 , 149
Data 152 , 155 , 158 , 162 , 165 , 167 , 170 , 173
Data 176 , 179 , 182 , 185 , 188 , 190 , 193 , 196
Data 198 , 201 , 203 , 206 , 208 , 211 , 213 , 215
Data 218 , 220 , 222 , 224 , 226 , 228 , 230 , 232
Data 234 , 235 , 237 , 238 , 240 , 241 , 243 , 244
Data 245 , 246 , 248 , 249 , 250 , 250 , 251 , 252
Data 253 , 253 , 254 , 254 , 254 , 255 , 255 , 255
Data 255 , 255 , 255 , 255 , 254 , 254 , 254 , 253
Data 253 , 252 , 251 , 250 , 250 , 249 , 248 , 246
Data 245 , 244 , 243 , 241 , 240 , 238 , 237 , 235
Data 234 , 232 , 230 , 228 , 226 , 224 , 222 , 220
Data 218 , 215 , 213 , 211 , 208 , 206 , 203 , 201
Data 198 , 196 , 193 , 190 , 188 , 185 , 182 , 179
Data 176 , 173 , 170 , 167 , 165 , 162 , 158 , 155
```

```
Data 152 , 149 , 146 , 143 , 140 , 137 , 134 , 131
Data 128 , 124 , 121 , 118 , 115 , 112 , 109 , 106
Data 103 , 100 , 97 , 93 , 90 , 88 , 85 , 82
Data 79 , 76 , 73 , 70 , 67 , 65 , 62 , 59
Data 57 , 54 , 52 , 49 , 47 , 44 , 42 , 40
Data 37 , 35 , 33 , 31 , 29 , 27 , 25 , 23
Data 21 , 20 , 18 , 17 , 15 , 14 , 12 , 11
Data 10 , 9 , 7 , 6 , 5 , 5 , 4 , 3
Data 2 , 2 , 1 , 1 , 1 , 0 , 0 , 0
Data 0 , 0 , 0 , 0 , 1 , 1 , 1 , 2
Data 2 , 3 , 4 , 5 , 5 , 6 , 7 , 9
Data 10 , 11 , 12 , 14 , 15 , 17 , 18 , 20
Data 21 , 23 , 25 , 27 , 29 , 31 , 33 , 35
Data 37 , 40 , 42 , 44 , 47 , 49 , 52 , 54
Data 57 , 59 , 62 , 65 , 67 , 70 , 73 , 76
Data 79 , 82 , 85 , 88 , 90 , 93 , 97 , 100
Data 103 , 106 , 109 , 112 , 115 , 118 , 121 , 124
```