

```

;- DL8MCG ----- HAM-Radio-CW-Morse-Code-Generator ----- auf UniDsp56-Board von DL9GFA ----

CwBufLen          equ      6
CW_FREQ           equ      800
DIRAC_VAL_CW     equ      $ffff          ;DIRAC value on t=0 (set amplitude!)
COS_OMEGA_CW     equ      @COS(2*PI*CW_FREQ/SAMPLE_FREQ) ;compute cos(omega)
SIN_OMEGA_CW     equ      @SIN(2*PI*CW_FREQ/SAMPLE_FREQ) ;compute sin(omega)

;----- CW-Ausgabe -----
org x:

CwYn_1            ds       1
CwYn_2            ds       1
CwDacVal          ds       1
CwBuf             ds       CwBufLen

CwSM              ds       1

CwCos             ds       1
CwiShape          ds       1

CwBufByteCnt      ds       1
CwBufPtr          ds       1
CwChar            ds       1

org y:

coeffsShape       bsm      nCoeffsShape ;include 'shape_coef.asm' ; e-funktion

org p:

;----- CW-Init -----
CwInit:           move     #coeffsShape,x0          ; Initialisierungen beim Programmstart
                  move     x0,x:CwiShape
                  move     #0,x0
                  move     x0,x:CwDacVal
                  move     #0,x0                    ; clear start values of
                  move     x0,x:CwYn_1              ; Y(n-1) and Y(n-2)
                  move     x0,x:CwYn_2
                  move     #DIRAC_VAL_CW,x0
                  move     #SIN_OMEGA_CW,x1
                  move     #COS_OMEGA_CW,y0
                  move     y0,x:CwCos
                  mpy      x0,x1,a                  ; Dirac to start
                  jmp      CwTon_

;----- zyklischer Aufruf CW-Ton-Oszillator -----
CwTon             clr      a                        ; Wiedereintrittspunkt
CwTon_            move     #-1,m0
                  move     x:CwiShape,r0
                  move     x:CwYn_2,x0
                  sub      x0,a                    ; a = DIRAC-Yn_2
                  move     x:CwYn_1,x0
                  move     x0,x:CwYn_2            ; Yn_1 -> Yn_2
                  move     x:CwCos,y0             ; COS_OMEGA,y0
                  mpy      x0,y0,b                ; b = COS_OMEGA * Yn_1
                  asl      b                       ; b = 2 * COS_OMEGA * Yn_1
                  add     b,a                      ; a = DIRAC + 2*COS_OMEGA*Yn_1 - Yn_2
                  nop
                  move     a,y0
                  move     a,x:CwYn_1            ; store value to Y(n-1) for next sample

                  jclr     #F_CW,x:Flags,CwTon2
                  move     y:(r0)+,x0            ; Amplitude erhöhen, e-Function-Shaping-Faktor holen
                  mpy      x0,y0,a                ; bewerten
                  move     r0,b
                  move     a,x:CwDacVal          ; ausgeben
                  cmp      #coeffsShape+nCoeffsShape,b ; Maximum erreicht?
                  jge     CwTon1
                  move     r0,x:CwiShape
CwTon1            rts

CwTon2            move     y:(r0)-,x0            ; Amplitude reduzieren, e-Function-Shaping-Faktor holen
                  mpy      x0,y0,a                ; bewerten
                  move     r0,b
                  move     a,x:CwDacVal          ; ausgeben
                  cmp      #coeffsShape-1,b        ; Minimum erreicht?
                  jle     CwTon3
                  move     r0,x:CwiShape
CwTon3            rts

;-----

;----- Timer0-IRQ -----
Timer0_init       move     #>CwHalt,r0
                  move     r0,x:CwSM
                  move     #>CwBuf,x0
                  move     x0,x:CwBufPtr

```

```

        move    #0,x0
        move    x0,x:CwBufByteCnt

        bset    #M_TOL0,x:M_IPRP          ; IRQ-Level-Timer
        bclr    #M_TOL1,x:M_IPRP          ;
        movep   #>50000,x:M_TPLR         ; Timer Prescaler Load Register, -> 1 kHz
        movep   #>50,x:M_TCPR0           ; Timer Compare Register 0
        movep   #>$0,x:M_TLR0            ; Timer Load Register 0
        movep   #((1<<M_TCF)+(1<<M_PCE)+(1<<M_DIR)+(1<<M_TRM)+(1<<M_TCIE)),x:M_TCSR0
        bset    #M_TE,x:M_TCSR0          ; starte CW-Timer
        rts

Timer0_OF_isr    rti                    ; not in use

Timer0_CP_isr    move    r0,x:(r6)+
                 move    x:CwSM,r0
                 move    n0,x:(r6)+
                 move    m0,x:(r6)+
                 move    b2,x:(r6)+
                 move    b1,x:(r6)+
                 move    b0,x:(r6)+
                 move    a2,x:(r6)+
                 move    a1,x:(r6)+
                 move    a0,x:(r6)+
                 jsr     r0                ; call state machine
                 move    x:-(r6),a0
                 move    x:-(r6),a1
                 move    x:-(r6),a2
                 move    x:-(r6),b0
                 move    x:-(r6),b1
                 move    x:-(r6),b2
                 move    x:-(r6),m0
                 move    x:-(r6),n0
                 move    x:-(r6),r0
                 rti

;----- Hole nächstes Zeichen -----
CwNextChar      move    x:CwBufByteCnt,b
                 move    x:CwBufPtr,r0
                 clr     a
                 add     #>1,b
                 move    x:(r0),a0
                 asl     #8,a,a          ; extract Byte
                 cmp     #>3,b
                 jne     CwNextChar1
                 move    #0,x0
                 move    x0,x:CwBufByteCnt
                 move    a0,x:(r0)+
                 move    r0,x:CwBufPtr
                 jmp     CwNextChar2
CwNextChar1     move    b1,x:CwBufByteCnt
                 move    a0,x:(r0)
CwNextChar2     move    a1,x:CwChar
                 move    x:CwChar,a
                 cmp     #>$FF,a        ; Wort-Ende?
                 jeq    CwKeyPauseWord
                 cmp     #0,a
                 jeq    CwKeyEnde

;----- Umsetzung in Punkt/Strich/Pause-Einheiten -----
CwKey           move    x:CwChar,a
                 and     #>$0000FF,a
                 cmp     #>$000080,a    ; Zeichen-Ende?
                 jeq    CwKeyPauseSign
                 asl     a
                 bset    #F_CW,x:Flags
                 move    a1,x:CwChar
                 jset    #8,a,CwKeyDash
                 move    #CwPauseDot,x0
                 move    x0,x:CwSM
                 rts

CwKeyDash       move    #CwDash,x0
                 move    x0,x:CwSM
                 rts

CwKeyEnde       bclr    #DSW_CwSeq1_run,x:DspStatusWord
                 jsr     SendStatusToUc
                 move    #CwHalt,x0
                 move    x0,x:CwSM
                 rts

CwKeyPauseSign  move    #CwPauseSign,x0
                 move    x0,x:CwSM
                 rts

;----- Pause nach Wort -----
CwKeyPauseWord  move    #CwPauseWord1,x0

```

```

        move    x0,x:CwSM
        rts

CwPauseWord1    move    #CwPauseWord2,x0
                move    x0,x:CwSM
                rts

CwPauseWord2    move    #CwPauseWord3,x0
                move    x0,x:CwSM
                rts

CwPauseWord3    move    #CwNextChar,x0
                move    x0,x:CwSM
                rts

;----- Strich -----
CwDash          move    #CwDash1,x0
                move    x0,x:CwSM
                rts

CwDash1         move    #CwPauseDot,x0
                move    x0,x:CwSM
                rts

;----- Eine Zeicheneinheit Pause -----
CwPauseDot      bclr    #F_CW,x:Flags      ; CW-Ton aus
                move    #CwKey,x0
                move    x0,x:CwSM
                rts

;----- Pause nach Zeichen -----
CwPauseSign     move    #CwNextChar,x0
                move    x0,x:CwSM
                rts

;----- stand by state -----
CwHalt          jclr    #DSW_CwSeq1_run,x:DspStatusWord,CwHalt1 ; neue CW-Sequenz?
                move    #>CwBuf,x0
                move    x0,x:CwBufPtr
                move    #0,x0
                move    x0,x:CwBufByteCnt
                move    #CwNextChar,x0
                move    x0,x:CwSM

CwHalt1         rts
```