AN913
APPLICATION NOTE

PWM GENERATION WITH THE ST62 16-BIT AUTO-RELOAD TIMER

by 8-bit Micro Application Team

INTRODUCTION

This note presents how to use the ST62 16-bit Auto-Reload Timer (ARTimer) for generating a DTMF signal (Dual-Tone Multiple Frequency) with the PWM. In the example shown, the PWM output pin generates a DTMF to dial a telephone number.

1 16-BIT AUTO-RELOAD TIMER DESCRIPTION

This timer is a 16-bit downcounter timer with prescaler (see Figure 1.). It includes auto-reload PWM, capture and compare capability with two input (CP1, CP2) and two output pins (OVF, PWM). It is controlled by the following registers:

– Status control registers (8-bit) SCR1, SCR2, SCR3, SCR4

– CP Capture register pair (16-bits total): CPH and CPL

– MASK register pair (16-bits total): MASKH and MASKL

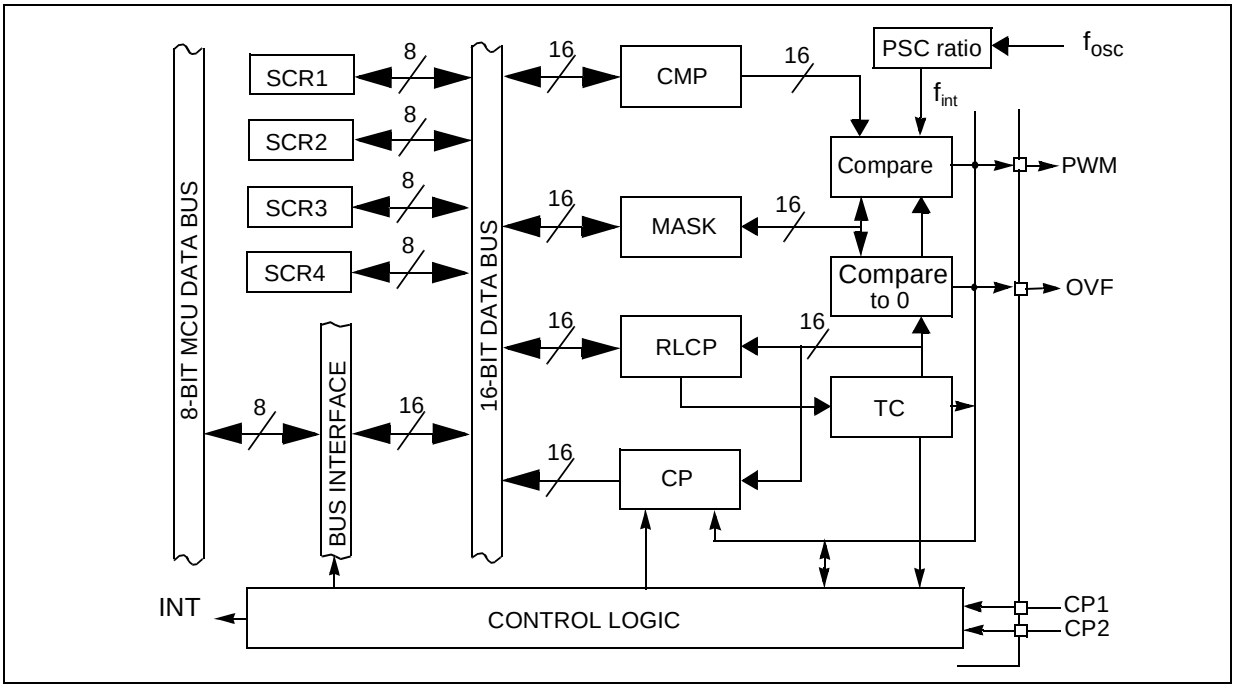
– TC 16-bit downcounter register

– CMP Compare register pair (16-bits total): CMPH and CMPL

– RLCP Reload/Capture register pair (16-bits total): RLCPH and RLCPL

The prescaler ratio can be programmed to choose the timer input frequency f_{int}(see Table 1).

Figure 1. 16-bit Auto-Reload Timer Block Diagram



2 PULSE WIDTH MODULATION (PWM) GENERATION

Using the PWM generation capability of the ARTimer, the CPU of the microcontroller has only to start/stop the timer and update the duty cycle. High speed PWM signals in the range of up to 4 MHz can be generated (RLCP=1). The timer clock input frequency f_{int} can be selected by the oscillator clock f_{osc} and the prescaler ratio (**PSC2**,**PSC1** bits of **SCR1** register). The PWM signal period is controlled by the Reload register RLCP. The duty cycle is defined by the Compare register CMP (see Figure 2.).

The TC register is decremented from the RLCP value to 0, then reloaded at the RLCP value to decrement again. The PWM output is set or reset or toggled when **MASK&TC=0** (depending on the **PWMPOL** bit of the SCR4 register and the **PWMMD** bit of the SCR3 register) and is reset or set or toggled when **MASK&TC=MASK&CMP**.

The CMP register can have any value between the RLCP value and 0, but in PWM mode, this CMP register value should not be equal to zero.

So the RLCP value should be as high as possible, the MASK value equal to FFFFh and the prescaler ratio as low as possible to achieve maximum resolution.

Major equations for PWM generation are:

$f_{int} = f_{osc}$ / (prescaler ratio)

$f_{PWM} = f_{osc}$ / ((RLCP+1)*prescaler ratio)

With PWMPOL=0

Duty cycle: (RLCP-CMP) / (RLCP+1)

With PWMPOL=1

Duty cycle: (CMP+1) / (RLCP+1)

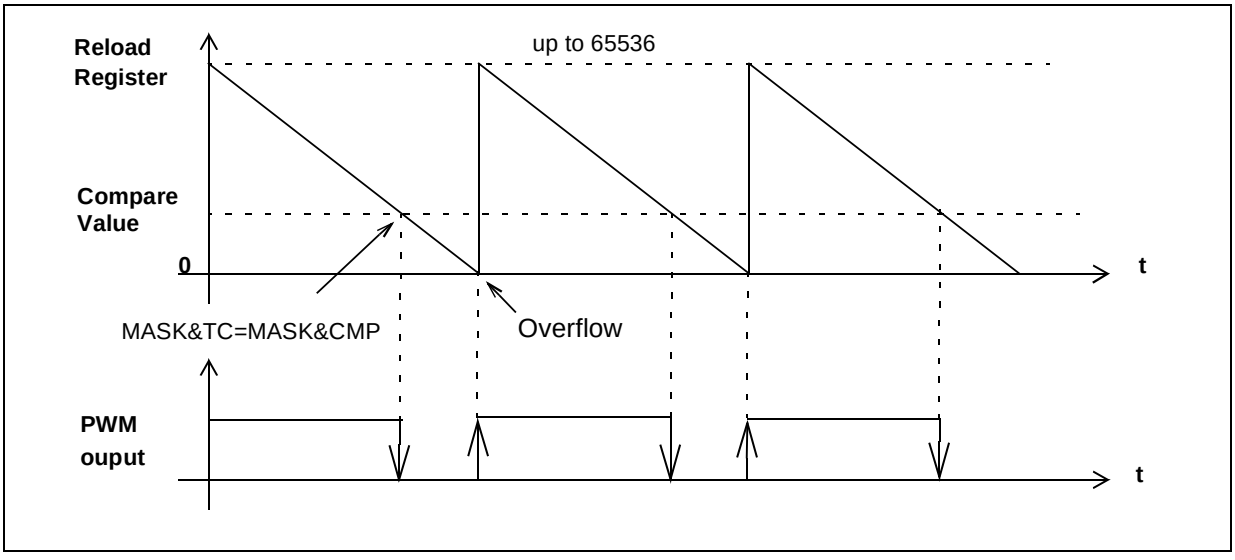
Resolution: 1 / (RLCP+1)

With a 8 MHz oscillator and a prescaler ratio of 1, the maximum resolution (1/(RLCP=FFFFh)) leads to a PWM frequency f_{PWM} of 122Hz. In this case an analog lowpass filter with a high order gives an analog output equivalent to a 16 bit DAC!

Table 1. Prescaler Programming Ratio

PSC2	PSC1	PRESCALER Ratio
0	0	Clock Disabled
0	1	1
1	0	4
1	1	16

Figure 2. PWM Timer Operation with PWMPOL=0, MASK=FFFFh and RELOAD mode (Reload bit=1)



Example:

The purpose of this example is to use the PWM as an 8-bit precision DAC for a Telecom DTMF application.

DTMF is a succession of breaks and tones. The tone is made of two sinus waves for each digit to send.

To make the sinus waves, we use a window look-up table of 64 samples. For each frequency the phase is calculated with 16-bit precision and the look-up table is indexed with the 8 most significant bits. The sum of the two values obtained from the look-up table is loaded in the CMP register.

The PWM is selected with the set/reset mode.

With a 8MHz clock frequency, a PSC value of 4 and a RLCP register value at 177h (RLCPH=100h and RLCPL=77h) the PWM gives a sample frequency of:

8MHz/(4*(177h+1)) = 5319Hz

This frequency is the highest one in order to have 115 CPU cycles to calculate the tone.

To calculate the sinus wave with only 8 bits, the sinus table gives data between 1 and 7Fh. The addition of the two sinus waves gives an 8-bit data stored in the CMP register when an overflow occurs.

To avoid the TC register going below the maximum value of the CMP register (which is the maximum value of the DTMF result FFh) when the program wants to reload the CMP register after an ARTimer overflow, the CPU cycle time between the overflow and TC=FFh equals:

(177h-FFh)*4/8MHz*8MHz/13 = 36 cycles

36 cycles are enough to load the CMP register after the overflow.

This program dials a telephone number, for example 012345678 and loops when it is finished.

To manage the timing (40ms for the break and 96ms for the tone), the standard 8-bit timer interrupts the sinus calculation. The input clock pin of the standard 8-bit timer is connected to the ARTimer OVF output pin which is set in toggle mode to give a frequency of:

5319/2=2659.5Hz. (toggle mode divide the 5319Hz by 2).

The TCR register of the standard 8-bit timer is loaded with:

40ms*2659.5 = 6Ah for the break

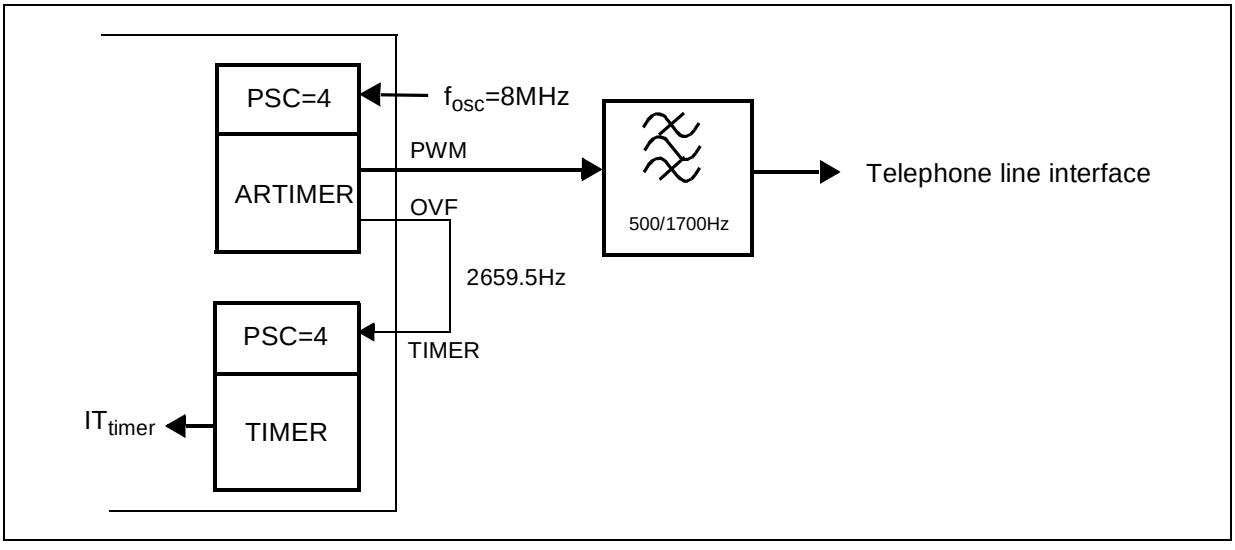
96ms*2659.5 = FFh for the tone

To be connected to a telephone line, a 2-order analog 500Hz/1700Hz Tchebycheff bandpass filter is enough (see Figure 3.).

To respect the maximum shift frequency of +/- 10Hz for each frequency the clock frequency precision must be better than:

10/1633=0.61% (1633 is the maximum DTMF frequency used).

Figure 3. Connection between modules



3 PROGRAM EXAMPLE

```
       .vers "st6230"
       .romsize 8
;*** data registers ***
  .input "623x.asm"
;*** data RAM ***
Msb_inc1.def        084h  ; Msb of the increment tone 1
Lsb_inc1.def        085h  ; Lsb of the increment tone 1
Msb_inc2.def        086h  ; Msb of the increment tone 2
Lsb_inc2.def        087h  ; Lsb of the increment tone 2
Msb_ton1.def        088h  ; Msb of the tone 1
Lsb_ton1.def        089h  ; Lsb of the tone 1
Msb_ton2.def        08Ah        ; Msb of the tone 2
Lsb_ton2.def        08Bh  ; Lsb of the tone 2
WaitTone.def        08Ch  ; even -> tone, odd -> break
NToSend.def         08Dh  ; Number to dial from the telephone number
Tel_Num.def         08Eh  ; Telephone number to dial
;*** Definition ***
OVFFLG .def 2             ; Overflow flag bit position
tmz    .equ 7             ; This bit must be cleared by user software
                          ; before starting a new count.
;*************************************************************************
       .org 800h
;*** sinus() ***
sinus:
      .byte        040h,046h,04ch,052h,058h,05eh,063h,068h
      .byte        06dh,071h,074h,078h,07ah,07ch,07eh,07fh
      .byte        07fh,07fh,07eh,07ch,07ah,078h,074h,071h
      .byte        06dh,068h,063h,05eh,058h,052h,04ch,046h
      .byte        040h,03ah,034h,02eh,028h,022h,01dh,018h
      .byte        013h,0fh,0ch,08h,06h,04h,02h,01h
      .byte        01h,01h,02h,04h,06h,08h,0ch,0fh
      .byte        013h,018h,01dh,022h,028h,02eh,034h,03ah
num_dial:
;*** f(16 bits)=F(Hz)/5319*64*256 ***
;                  FREQ1   FREQ2
;                  Msb1 Lsb1 Msb2 Lsb2
      .byte        0bh, 53h, 10h, 13h;number 0
      .byte        08h, 63h, 0eh, 8ch;number 1
      .byte        08h, 63h, 10h, 13h;number 2
      .byte        08h, 63h, 11h,0c6H;number 3
      .byte        09h, 44h, 0eh, 8ch;number 4
      .byte        09h, 44h, 10h, 13h;number 5
      .byte        09h, 44h, 11h,0c6h;number 6
```

```
       .byte        0ah, 40h, 0eh, 8ch;number 7
       .byte        0ah, 40h, 10h, 13h;number 8
       .byte        0ah, 40h, 11h,0c6h;number 9
       .byte        08h, 63h, 13h,0a6h;number A
       .byte        09h, 44h, 13h,0a6h;number B
       .byte        0ah, 40h, 13h,0a6h;number C
       .byte        0bh, 53h, 13h,0a6h;number D
       .byte        0bh, 53h, 0eh, 8ch;number *
       .byte        0bh, 53h, 11h,0c6h;number #
;********************** INITIALIZATION *********************************
reset
       reti
       ldi          drwr,20h    ; Rom window with sinus
       ldi          Lsb_ton1,0h ; Initialize the tone for break
       ldi          Msb_ton1,0h
       ldi          Lsb_ton2,0h
       ldi          Msb_ton2,0h
       ldi          Lsb_inc1,0h
       ldi          Msb_inc1,0h
       ldi          Lsb_inc2,0h
       ldi          Msb_inc2,0h
       ldi          x,#Tel_Num  ; The Telephone number is 01234567
       ldi          a,0
       ld           Tel_Num,a
       ldi          a,1
       ld           Tel_Num+1,a
       ldi          a,2
       ld           Tel_Num+2,a
       ldi          a,3
       ld           Tel_Num+3,a
       ldi          a,4
       ld           Tel_Num+4,a
       ldi          a,5
       ld           Tel_Num+5,a
       ldi          a,6
       ld           Tel_Num+6,a
       ldi          a,7
       ld           Tel_Num+7,a
       ldi          a,7
       ld           Tel_Num+8,a
       ldi          a,8
       ld           Tel_Num+9,a
       ldi          WaitTone,0        ; WaitTone is even to start with tone
                                      ; instead of break
       ldi          NToSend,0         ; First number to dial
```

```
;*** TIMER Initialisation ***
       ldi          tcr,0ffh          ; Load the counter
       ldi          tscr,01001000b    ; Restart the timer with input tone
                                      ; from input pin which is connected to the
                                      ; OVFFLG of ARTIMER.
                                      ; The prescaler is 1
;*** PortA initialisation for PWM and OVF output
       ldi          ddra,0ch
       ldi          ora,0ch

;*** ART16 Initialisation ***
       ldi          SCR2,00h          ; CP1 input interrupt disable
       ldi          SCR3,00h          ; CP2 and ZEROFLG interrupt disable
                                      ; PWM output is run in SET/RESET mode
       ldi          SCR4,0Fh          ; OVFPOL=1, Overflow enable, PWMPOL=1,
                                      ; PWM enable
       ldi          RLCPH,001h        ; 100h for 8 bit of PWM, 77h to have time to
       ldi          RLCPL,077h        ; load the CMP register, and to have with
                                      ; 8MHz and a prescaler of 1/4 a
                                      ; Digital Analog Converter
                                      ; sample tone equivalent to 5319hz.
       ldi          MASKH,0FFh        ; MASK = 0FFFFh
       ldi          MASKL,0FFh
       ldi          SCR1,0B2h         ; prescal by 4 to have a sample tone of 2MHz
                                      ; Reload mode
                                      ; Runres =1 to start the counter
                                      ; No interrupt with overflow
                                      ; Toggle mode to give 2000 Hz to the timer
                                      ; (with toggle mode it's not necessary to
                                      ; reset the OVFFLAG)
;*** GENERAL INTERRUPT ***
       ldi          ior,10h           ; Enable all interrupts.

;*****************************************************************************
;************************** DTMF GENERATION ******************************
DTMF:
;*** calculation of the first tone ***
       ld           a,Lsb_ton1        ; load tone Lsb
       add          a,Lsb_inc1        ; add it to the increment Lsb
       ld           Lsb_ton1,a        ; save the result
       ldi          a,0               ; initiate a to 0 for no carry
       jrnc         nocarry1          ; test carry of the previous result
       ldi          a,1               ; if carry a=1 to add 1 to the Msb addition
nocarry1:
       add          a,Msb_ton1        ; add the carry to the tone Msb
```

```
      add           a,Msb_inc1        ; add it to the increment Lsb
      ld            Msb_ton1,a        ; save the result
;*** calculation of sin(2*PI*f) with look up table ***
      andi          a,03fh            ; mask the Msb tone to call the look up
table
      addi          a,040h            ; add the result to the address table
      ld            x,a               ;use the indirect pointer x to read the
data
;*** calculation of the second tone ***
      ld            a,Lsb_ton2        ; load tone Lsb
      add           a,Lsb_inc2        ; add it to the increment Lsb
      ld            Lsb_ton2,a        ; save the result
      ldi           a,0               ; initiate a to 0 for no carry
      jrnc          nocarry2          ; test carry of the previous result
      ldi           a,1               ; if carry a=1 to add 1 to the Msb addition
nocarry2:
      add           a,Msb_ton2        ; add the carry to the tone Msb
      add           a,Msb_inc2        ; add it to the increment Lsb
      ld            Msb_ton2,a        ; save the result
;*** calculation of sin(2*PI*f) with look up table ***
      andi          a,03fh            ; mask the Msb tone to call the look up
table
      addi          a,040h            ; add the result to the address table
      ld            y,a               ; use the indirect pointer y to read the
data
      ld            a,(y)             ; read the data
;*** add the two sinus ***
      add           a,(x)             ; add to the first tone
;*** wait overflow ***
wait_ovf:
      jrr           OVFFLG,SCR1,wait_ovf
      res           OVFFLG,SCR1       ; reset the overflow
;*** load compare register with a new value ***
      ld            CMPL,a
      jp            DTMF
;*************Subroutine Tone Generation **************
;*** Input with accumulator a  for the number to make
InitTone:
      ldi           drwr,21h          ; Rom window with num_dial
      sla           a
      sla           a
      addi          a,40h
      ld            x,a
      ld            a,(x)
      ld            Msb_inc1,a; First tone
```

```
       inc          x
       ld           a,(x)
       ld           Lsb_inc1,a
       inc          x
       ld           a,(x)
       ld           Msb_inc2,a        ; Second tone
       inc          x
       ld           a,(x)
       ld           Lsb_inc2,a
       ldi          drwr,20h          ; Rom window with sinus
       ret
;**********************END OF DTMF GENERATION***********************
;*******************************************************************************
;***************************** interrupt routine *************************
;*** timer interrupt ***
it_timer
       jrr          0,WaitTone,Break
       ld           a,NToSend
       jrr          3,NToSend,TelNotFinish
TelLoop:
       jrs          1,NToSend,TelLoop; infinit loop for this application note
                                      ; after dialing 10 numbers
TelNotFinish:
       addi         a,Tel_Num
       ld           x,a
       ld           a,(x)
       call         InitTone          ; Initialize the increment tone
       ldi          tcr,0ffh          ; Reload the counter for 96ms
       inc          NToSend
       JP           end_tim
Break:
       ldi          Lsb_inc1,0        ; All data to 0 for no sound.
       ldi          Msb_inc1,0        ; For other applications, the time break
       ldi          Lsb_inc2,0        ; could be used for process instead of
       ldi          Msb_inc2,0        ; calculate nothing
       ldi          Lsb_ton1,0
       ldi          Msb_ton1,0
       ldi          Lsb_ton2,0
       ldi          Msb_ton2,0
       ldi          tcr,06ah          ; Reload the counter for 40ms
end_tim
       inc          WaitTone
       res          tmz,tscr          ; Restart the timer.
       reti
```

```
;*** Restart and interrupt Vectors ***
       .org          0ff0h
timer  jp            it_timer          ;FF0h
       reti                            ;FF2h
       reti
       reti                            ;FF4h
       reti
       reti                            ;FF6h
       reti
       .org          0ffch
nmi    nop
       reti
res    jp            reset
```

THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNEXION WITH THEIR PRODUCTS.