CHAPTER 9

CREATING A SINE MODULATED PWM SIGNAL

9.1 Overview

This chapter introduces one method to generate sine modulated PWM signals. This application generates an asymmetrical pulse width modulated (PWM) signal with a varying duty cycle. The period of the PWM signal is 0.05ms, which is equivalent to a 20kHz signal. The duty cycle is modulated with a sine function that can be varied in frequency. The implementation of the sine wave modulation is through a look-up table. This application is implemented using C2xx Assembly code. The algorithm described in this application report was implemented using the TI TMS320F2407 EVM.

This application uses the Event Manager Module and General Purpose Timer 1 of the DSP. The frequency of the sinusoidal function modulated by the PWM signal is determined by variable declaration in the program and therefore no input signal is needed. The output PWM signal goes from pin T1PWM / T1CMP on P1 connector.

9.2 Methodology- The Table Look-up Algorithm

The generation of the sine wave is performed using a look up table. To be able to control the frequency of the modulation with some accuracy, a method based on the modulo mathematical operation is used (i.e. any overflow is disregarded and only the remainder is kept).

In this application a 16-bit counter is used to determine the location of the next value. A step value is added to the counter every time a new value from the sine table is to be loaded. By changing the value of the step, one can accurately control the frequency of the sine wave.

Although a 16-bit counter is used, the upper byte determines the location of the next sine value to be used; thus, by changing how quickly values overflow from the lower byte (i.e., manipulating the step value), the frequency of the sine wave can be changed. The modulo mathematical operation is used when there is overflow in the accumulator from the lower word to the upper word. When an overflow occurs, only the remainder (lower word) is stored.

For example, the counter is set to 0000h and the step value is set to 40h. Every time a value is to be looked up in the table, the value 40h is added to the counter; however, since the upper byte is used as the pointer on the look up table, the first, second, and third values will point to the same location. In the fourth step, which results in an overflow into the upper byte, the value that is loaded will change. Since the upper byte is used as the pointer, the look-up table has 256 values, which is equivalent to the number of possibilities for an 8-bit number: 0 to 255. Additionally, since the upper word of the accumulator is disregarded, the pointer for the sine look up table does not need to be reset.

Table 1. Look-Up Table Example 1

Step	Accumulator	Counter	Pointer	Step Value = 40h
0	0000 0000h	0000h	00h	1st value of sine table
1	0000 0040h	0040h	00h	
2	0000 0080h	0080h	00h	
3	0000 00C0h	00C0h	00h	
4	0000 0100h	0100h	01h	2nd value of sine table
. . . .				
. . . .				
. . . .				
n	0000 FFC0h	FFC0h	FFh	256th value of sine table
n+1	0001 0000h	0000h	00h	1st value of sine table
n+2	0000 0040h	0040h	00h	

The step size controls the frequency that is output; as a result, the larger the step, the quicker the overflow into the upper byte, and the faster the pointer traverses through the sine look-up table.

Table 2. Look-Up Table Example 2

Step	Counter	Pointe r	Step Value = C0h
0	0000h	00h	1st value of sine table
1	00C0h	00h	
2	0180h	01h	2nd value of sine table
3	0240h	02h	3rd value of sine table
4	0300h	03h	4th value of sine table

Although the step size indicates how quickly the pointer moves through the look up table, the step size does not provide much information about the approximate frequency that the sine wave will be modulating the PWM signal. To determine the frequency of the sine wave, determine how often the value in the compare register will be modified.

In this application, the routine to load a new value in the compare register is accessed every time that the timer value matches the value in the period register. Consequently, the routine will be accessed at the same frequency as the PWM signal (20kHz). Because the compare register will be updated each time that the period register and the timer values are equal, the routine that modifies the compare register will be implemented as an interrupt service routine. As a result, the proper registers, EVIMRA and the core IMR need to unmask the proper interrupt levels so that the compare register can be updated.

The frequency that the sine wave will be modulated at can be calculated from the following formula

$$f(step) = \frac{step}{T_s \times 2^n}$$

where:

$f(step)$ = desired frequency

T_s = the time period between each update (in this case, the PWM period)

n = the number of bits in the counter register (n = 16 here)

step = the step size used

The frequency that the PWM signal will be modulated is proportional to the step size and inversely proportional to the size of the counter register and the period at which the routine is accessed. Thus, to increase the resolution that one can increment or decrement the frequency of the PWM modulation, one needs to have a larger counting register or access the routine at a slower frequency by increasing the period.

Since this program is interrupt driven, once the registers have been set for the PWM signal, the program can be ended with an unconditional branch and the output will continue because of the interrupt structure. The output will stop when the user halts the program or the software masks the corresponding interrupt levels.

The value of `*FREQSTEP` can be changed in the watch window of Code Composer under real-time mode, so that one can modify the step size to change the frequency of modulation.

The complete code is given below -

```
;*********************************************************************
; File Name:       ch9_e1.asm
; Target System: C240x Evaluation Board
; Description:    Pulse Width Modulator - Sets up the registers for an
;           asymmetric PWM output. The output is a square wave with
;           a sine wave modulated duty cycle. Timer compare value
;           is updated every PWM cycle, which is the timer period.
;           PWM (timer T1) Period is 0.05ms => 20kHz
;
;           Real-time mode enabled.
;*********************************************************************

;*********************************************************************
;                               SYSTEM OPTIONS
;*********************************************************************
real_time         .set 1      ; 1 for real time mode, otherwise set 0
BUFFER            .set 8000h  ; buffer location
BUFFER_size       .set 500    ; buffer size 500
T1COMPARE         .set 0      ; T1Compare Initialized to 0
T1PERIOD          .set 1500   ; T1Period Initialized to 1500 = 20kHz
NORMAL            .set 750    ; Half of T1PR value
STEPSIZE          .set 4      ; Step size constant value
;*********************************************************************

;---------------------------------------------------------------------
; External references
;---------------------------------------------------------------------
      .include    "f2407.h"
      .global     MON_RT_CNFG

      .ref  SYS_INIT, STABLE  ; STABLE is the starting address of
                              ; the sine table

;---------------------------------------------------------------------
; Local Variable Declarations
;---------------------------------------------------------------------
      .def  GPR0              ;General purpose register.

      .bss  GPR0,1            ;General purpose register.
      .bss  ctr,1             ;variable for the main background loop
      .bss  BUFFER_write_ptr,1 ;data buffer location pointer
      .bss  BUFFER_ctr,1      ;data counter for writing buffer
      .bss  TABLE,1           ;Stores pointer address in the SINE Table
      .bss  TOPTABLE,1        ;Stores the reset value for the pointer
      .bss  COMPARET1,1       ;A variable to do calculations since the
                              ;T1CMPR register is double buffered
      .bss  FREQSTEP,1        ;Step size of counter increment
      .bss  MODREG,1          ;Rolling Modulo Register
      .bss  SINEVAL,1         ;Value from look up table

;=====================================================================
; V E C T O R   T A B L E   ( including RT monitor traps )
;=====================================================================
      .include "c200mnrt.i"   ; Include conditional assembly options.

      .global _c_int0,PHANTOM,GISR1,GISR2,GISR3,GISR4,GISR5,GISR6
```

```
;=========================================================================
; M A I N   C O D E  - starts here
;=========================================================================
      .text
_c_int0:
      CALL  SYS_INIT          ;DSP initialization

;-----------------------------------------------------------
; Initialise the Real time monitor
;-----------------------------------------------------------
;---Real Time option---------------
      .if (real_time)
            CALL        MON_RT_CNFG       ;For Real-Time
      .endif
;------------------------------------

;-----------------------------------------------------------
; System Interrupt Init.
;-----------------------------------------------------------

;---Real Time option -----------------------------------------------------
      .if (real_time)
            SPLK  #0000000001000010b,IMR  ;En Int 2 & 7 for T1 and RT
                  ;|||||||||!||||||
                  ;5432109876543210
      .endif


      .if (real_time != 1)
            SPLK  #0000000000000010b,IMR  ;Enable Int 2 only for T1
                  ;||||!!!!||||!!!!
                  ;5432109876543210
      .endif

            SPLK  #0FFFFh, IFR            ;Clear any pending Ints
;-------------------------------------------------------------------------

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Setup shared I/O pins
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #DP_PF2                 ;set data page
        SPLK    #0011000000000000b,MCRA ;set TxPWM pins
*                ||||||||||||||||
*                FEDCBA9876543210
* bit 13        1:      0=IOPB5,     1=T2PWM/T2CMP
* bit 12        1:      0=IOPB4,     1=T1PWM/T1CMP

;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;- Event Manager A Module Reset
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      LDP  #DP_EVA            ;DP_EVA Data Page for the Event Manager

      SPLK #0000h,GPTCONA     ;Clear General Purpose Timer Control
      SPLK #0000h,T1CON       ;Clear GP Timer 1 Control
      SPLK #0000h,T2CON       ;Clear GP Timer 2 Control
      SPLK #0000h,COMCONA     ;Clear Compare Control
      SPLK #0000h,ACTRA       ;Clear Compare Action Control Register
      SPLK #0000h,DBTCONA     ;Clear Dead-Band Timer Control Register
```

```
      SPLK #0000h,CAPCONA     ;Clear Capture Control

      SPLK #0FFFFh,EVAIFRA    ;Clear Interrupt Flag Register A
      SPLK #0FFFFh,EVAIFRB    ;Clear Interrupt Flag Register B
      SPLK #0FFFFh,EVAIFRC    ;Clear Interrupt Flag Register C
      SPLK #0000h,EVAIMRA     ;Clear Event Manager Mask Register A
      SPLK #0000h,EVAIMRB     ;Clear Event Manager Mask Register B
      SPLK #0000h,EVAIMRC     ;Clear Event Manager Mask Register C

;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
; The main program - Launch T1 and initialize table look-up:
;*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      LDP  #DP_EVA            ;DP_EVA Data Page for EVA
      SPLK #T1COMPARE,T1CMPR  ;T1CMPR <-- 0

      SPLK #0000001010101b, GPTCONA
;           |||||||||||||
;           2109876543210
      SPLK #T1PERIOD,T1PR     ; T1PR = 1500 = 30000000/1/20000
      SPLK #0000h,T1CNT       ; Initialize Timer 1
      SPLK #0000h,T2CNT       ; Initialize Timer 2

      SPLK #0080h,EVAIMRA     ; Enable Timer 1 Period Interrupt

      SPLK #0001000001000110b,T1CON ; Start T1 counting, continuous up
;           ||||||||||||||||            Prescaler= 1
;           5432109876543210
      SPLK #0000000000000000b,T2CON       ; Not used

      LDP  #TABLE             ; Load data page of user variables
      SPLK #0000h,TABLE       ; Initialize Pointer to Top
      SPLK #STABLE,TOPTABLE   ; Initialize TOPTABLE to
                              ; address of sine table
      SPLK #STEPSIZE,FREQSTEP ; Set the step size
      SPLK #0000h,MODREG      ; Initialize the 16 bit counter register

* Initialize data buffer:
      LDP   #BUFFER_write_ptr    ; Initialization for buffer operation
      SPLK  #BUFFER, BUFFER_write_ptr   ; Initial value for the pointer
      SPLK  #BUFFER_size, BUFFER_ctr    ; Initial value for the counter

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Enable global interrupts
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      EINT        ;Enable all interrupts; clear the INTM to 0.

;=======================================================
;Main system background loop
;=======================================================
      LDP   #ctr
MAIN: SPLK  #7FFFh, ctr ;An infinite loop is running in background
loop: LACC  ctr         ;doing counter decrementing operation while
      SUB   #1          ;waiting for next ADC interrupt request.
      SACL  ctr         ;This background loop can be used to test
      BCND  loop,NEQ    ;the real-time running mode
      B     MAIN

;======================================================================
; I S R  -  GISR2
```

```
;
; Description: Look up the sine table for next value and update T1CMPR
;              and update the data buffer for graphical display
;======================================================================
GISR2 ;Context save regs
      MAR   *,AR1       ;AR1 is stack pointer
      MAR   *+          ;skip one position
      SST   #1, *+      ;save ST1
      SST   #0, *+      ;save ST0
      SACH  *+          ;save ACC high
      SACL  *           ;save ACC low
;=========================================================
;Start main section of ISR
;=========================================================
SINE  LDP  #MODREG      ; DP <-- user variable page
      LACC MODREG       ; ACC = 16 bit Counter Register
      ADD FREQSTEP      ; ACC = Counter + Step
      SACL MODREG       ; Counter assigned new value
      LACC MODREG,8     ; ACC = Counter shifted to left by 8
      SACH TABLE        ; TABLE = upper byte of counter = pointer
      LACC TABLE        ; ACC = TABLE = Pointer
      ADD TOPTABLE      ; Offset Addr from top of table
      TBLR SINEVAL      ; Read sine value and store to SINEVAL

;Normalization of the Sine value to prevent the compare value
;from being negative:
      LT SINEVAL        ; TREG = SINEVAL
      MPY #NORMAL       ; PREG = TREG * NORMAL  where NORMAL = T/2
      PAC               ; ACC = PREG
      SACH COMPARET1,1 ;COMPARET1=high 16 bits of (ACC left shift 1 bit)
            ; which is equivalent to multiply by 2 and devide by 65536

      LACC COMPARET1    ; ACC = COMPARET1
      ADD #NORMAL       ; ACC = COMPARET1 + NORMAL
      LDP #DP_EVA       ; DP <-- EVA page
      SACL T1CMPR       ; T1CMPR = ACC = Normalize Sine Value

      ; Write BUFFER
      LDP   #BUFFER_write_ptr       ; DP <-- user variable page
      LARP  5                       ; Use AR5 to be the current AR
      LAR   AR5, BUFFER_write_ptr   ; Load buffer pointer into AR5
      SACL  *+         ; Write ACC=T1CMPR value into the buffer location
                       ; pointed by the pointer; Increment AR5
      SAR   AR5, BUFFER_write_ptr   ; Store updated AR5 value back
                                    ;into the pointer
      LACC  BUFFER_ctr              ; Load data counter
      SUB   #1                      ; Decrement by 1
      BCND  ST_CTR,NEQ  ; If ACC >= 0, branch to ST_CTR (store counter)
RS_BUF:
      SPLK  #BUFFER, BUFFER_write_ptr ; If ACC = 0, buffer full, reset
                                    ; ptr to the beginning of the buffer
      LACC  #BUFFER_size; Reset the data counter back to the buffer size
ST_CTR:
      SACL    BUFFER_ctr

      ; Enable EVA interrupt:
      LDP   #DP_EVA                 ; DP <-- EVA page
      SPLK  #0FFFFh,EVAIFRA         ;Clear Interrupt Flag Register A
;=========================================================
;End main section of ISR
;==========================================================
      ;Context restore regs
      POINT_PG0         ;load data page 0
      MAR   *, AR1      ;make stack pointer active
      LACL  *-          ;Restore Acc low
      ADDH  *-          ;Restore Acc high
      LST   #0, *-      ;load ST0
      LST    #1, *-     ;load ST1
      EINT              ;Enable all interrupts; clear the INTM to 0.
      RET

;========================================================================
; I S R  -  PHANTOM
;
; Description:    Dummy ISR, used to trap spurious interrupts.
;========================================================================
PHANTOM B   PHANTOM
GISR1       RET
;GISR2      RET
GISR3       RET
GISR4       RET
GISR5       RET
GISR6 RET
```

Laboratory Experiment 9

Objectives

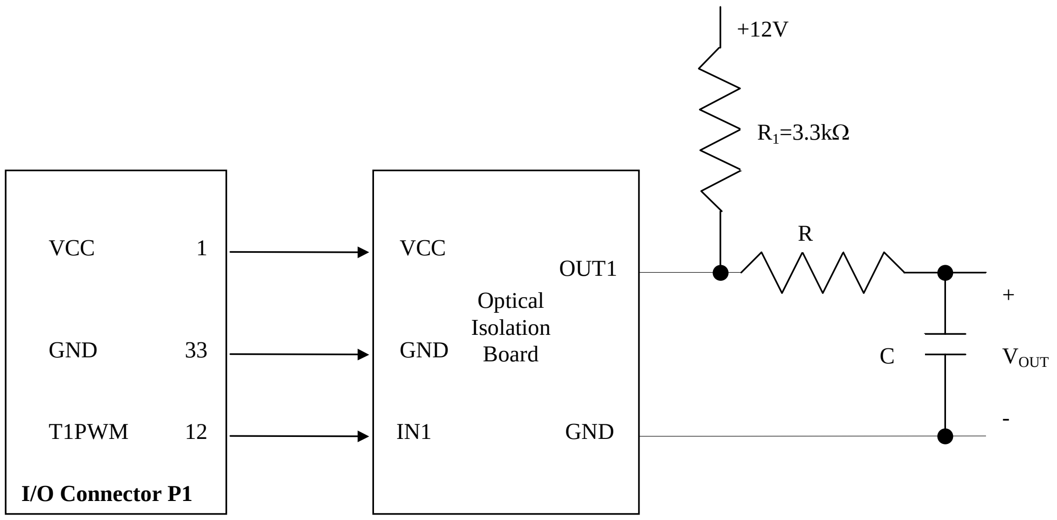
To understand one technique to create a sine modulated PWM signal using the DSP and control the frequency of the sinusoidal modulated function.

To observe the output sinusoidal waveform by applying a simple RC low-pass filter.

Laboratory Assignments

1. Run the example program `Ch9_e1.asm` and observe the output PWM waveform from the pin T1PWM on P1 connector using an oscilloscope.
2. Connect the output signal to an RC filter through an optical isolation board as shown in the figure below. Select filter parameters R=6.8kΩ, C=1μF. Change sine modulation frequency to 60Hz by modifying the step value in the program. Observe the output signal V_{OUT} using an oscilloscope. Answer:
   - What's the STEPSIZE value? What is the frequency of V_{OUT} observed from the oscilloscope? How close is it from the desired value?

- Why V_{OUT} is not purely sinusoidal?
- Try different filter resistance R values based on the resistor provided in the lab and observe the V_{out} waveform changes.


+12V
R1=3.3kΩ
VCC
1
GND
33
T1PWM
12
I/O Connector P1
VCC
GND
IN1
OUT1
GND
Optical Isolation Board
R
C
+
VOUT
-