Application Report
SLAA116 – December 2000

Using PWM Timer_B as a DAC

Mike Mitchell
MSP430

ABSTRACT

This application report describes how to simultaneously create a sine wave, a ramp, and a dc level with pulse-width modulated (PWM) signals from Timer_B on the MSP430 ultralow power family of microcontrollers. PWM signals are often used to create analog signals in embedded applications. This report shows how to create both ac signals and dc levels with PWM outputs. The example in this report uses Timer_B on the MSP430F149, but Timer_A could also be used in a similar manner.

Contents

Figures

1 Introduction

Many embedded microcontroller applications require generation of analog signals. Sometimes an integrated or stand-alone digital-to-analog converter (DAC) is used for this purpose. However, PWM signals can often be used for generating the required analog signals. PWM signals can be used to create both dc and ac analog signals. The report below discusses using a PWM timer as a DAC and shows an example of simultaneously creating a sinusoid, a ramp, and a dc level and adding the dc level and sine wave to produce an offset ac signal. This report uses the PWM timer Timer_B. Timer_A could also be used in a similar manner.

2 Theory of Operation

A PWM signal is a digital signal with fixed frequency but varying duty cycle. An example of a PWM signal is shown in Figure 1. If the duty cycle of the PWM signal is varied with time, and the PWM signal is filtered, the output of the filter will be an analog signal. The block diagram for a PWM DAC employing this technique is shown below in Figure 2. In the example code at the end of this report Timer_B on the MSP430F149 is used to simultaneously generate a sinusoid and a ramp waveform of different frequencies, and a dc level. PWM DACS can also be used to generate other signals. In fact, some speech processors from Texas Instruments, Inc. utilize PWM signals to generate speech for their applications.

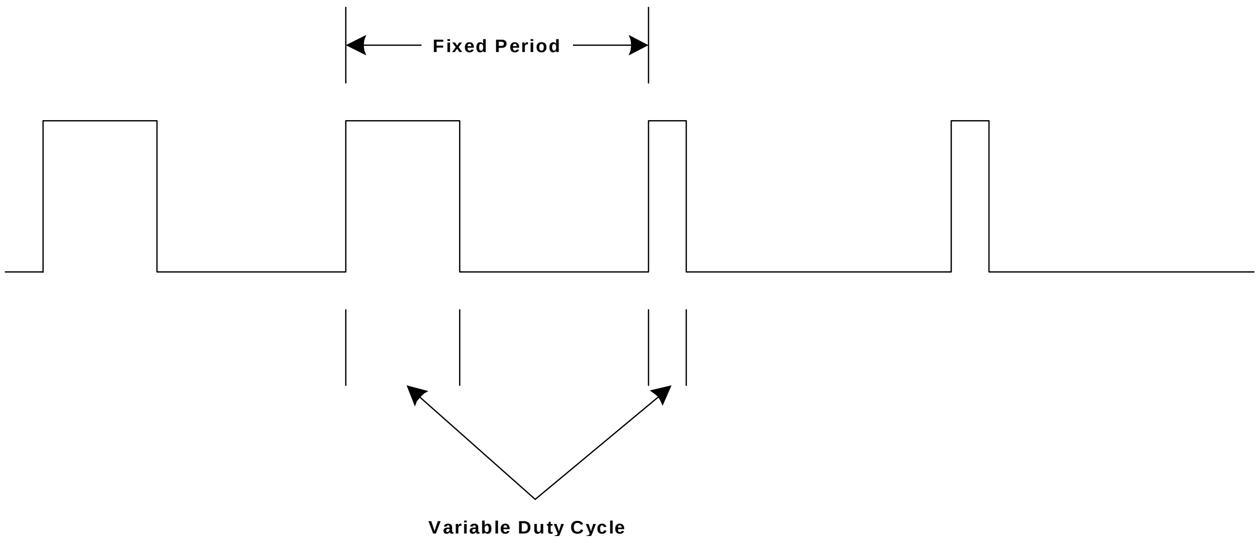

Figure 1. PWM Signal

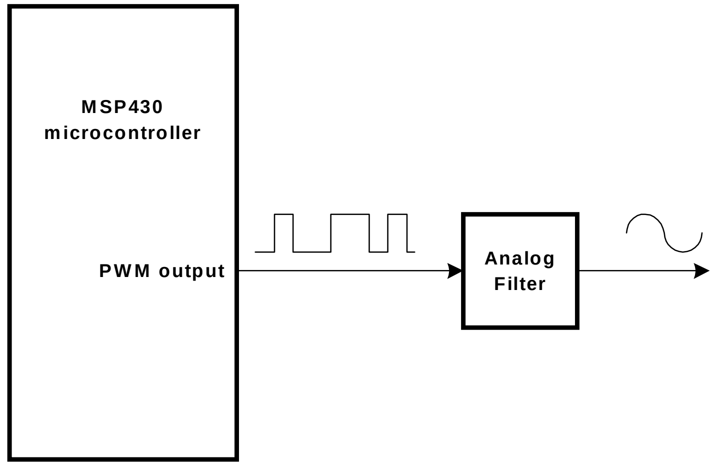

Figure 2. PWM DAC Block Diagram

2.1 Resolution

The resolution of a PWM DAC constructed with Timer_B is simply equivalent to the length of the counter, which is usually the value placed in CCR0. The LSB of the PWM DAC is one count and the resolution is the total number of counts:

$$R_{COUNTS} = L_{COUNTS}$$

R_{COUNTS} = Resolution in counts

L_{COUNTS} = Length of counter in counts

For example, this report implements an 8-bit DAC, so the length of the counter is 8-bits, or 256 counts.

In more general terms, the resolution of a PWM DAC constructed with a PWM timer and a filter is equivalent to the resolution of the PWM signal used to create the DAC. The resolution of the PWM signal is then dependant on the length of the counter and the smallest duty-cycle change the PWM counter is capable of making. The resolution is expressed mathematically as:

$$R_{COUNTS} = \frac{L}{C}$$

R_{COUNTS} = Resolution in counts

L = Length of counter in counts

C = Smallest duty-cycle change in counts

Expressed as number of bits the resolutions is calculated as:

$$R_{BITS} = Log_2(R_{COUNTS}) = Log_2\left(\frac{L}{C}\right) = \frac{\ln(\frac{L}{C})}{\ln(2)}$$

Or

$$R_{BITS} = Log_2(R_{COUNTS}) = \frac{\ln(R_{COUNTS})}{\ln(2)}$$

For example, if a PWM counter has a length of 512 counts and can vary the duty cycle by a minimum of 2 counts, the resolution in counts of the resulting PWM DAC would be:

$$R_{COUNTS} = \frac{L}{C} = \frac{512}{2} = 256$$

And the resolution in bits would be

$$R_{BITS} = Log_2(256) = \frac{\ln(256)}{\ln(2)} = 8bits$$

2.2 Frequency

The frequency required for the PWM output signal is equivalent to the update rate of the DAC, since each change in PWM duty cycle is the equivalent of one DAC sample. The frequency required for the PWM timer will depend on the required PWM signal frequency and the desired resolution. It is shown as:

$F_{clock} = F_{PWM}\, x 2^n$ where:

F_{clock} is the required PWM timer frequency

F_{PWM} is the PWM signal frequency, which is the DAC update rate

n is the desired resolution of the DAC in bits

This report shows how to construct an 8-bit PWM DAC and how to simultaneously generate a 250Hz sine wave and a 125Hz ramp. The desired sampling rate for this example is 8KHz (32 samples for each sine wave cycle (16x oversampled), and 64 samples for each ramp cycle (32x oversampled). This results in a required PWM signal frequency of 8KHz and a required PWM clock frequency of 2.048MHz.

It is usually best for the PWM signal frequency to be much higher than the desired sine wave frequency or the desired bandwidth of signals to be produced. Generally, the higher the PWM frequency the lower the order of filter required and the easier it is to build a suitable filter.

2.3 MSP430 resources used

The example code at the end of this report shows how to simultaneously generate a 250Hz sine wave, a 125Hz ramp, and a 2/3 V_{cc} dc value using Timer_B and external filters. Timer_A could be used in a similar manner.

Timer_B is used in the 16-bit mode, but is configured to operate in *up* mode where the counter counts up to the contents of capture/compare register 0 (CCR0) and then restarts at zero. CCR0 is loaded with 255 therefore giving the counter an effective 8-bit length. CCR1 and output TB1 are used for the sine wave. CCR2 and TB2 are used for the ramp, and CCR3 and TB3 are used for the dc value. For each output, the output mode is selected to be mode 7, or reset/set mode. In this mode, each output is reset when the counter reaches the respective CCRx value and is set when the counter reaches the CCR0 value. This provides positive pulses equivalent to the value in CCRx on each respective output. Finally, SMCLK is used as the clock source for Timer_B.

Other resources include:

- 32768Hz crystal oscillator
- On-chip digitally controlled oscillator (DCO) operating at 2.048MHz
- SMCLK and MCLK operating at 2.048MHz
- Timer_A used to calibrate the DCO
- Two CPU registers

2.4 Circuit Diagram and Signals

The complete circuit diagram used for this example is shown below in Figure 3. The ac signals produced by this example are shown in Figure 4. The dc value produced in this example is shown in Figure 5 together with its PWM signal.

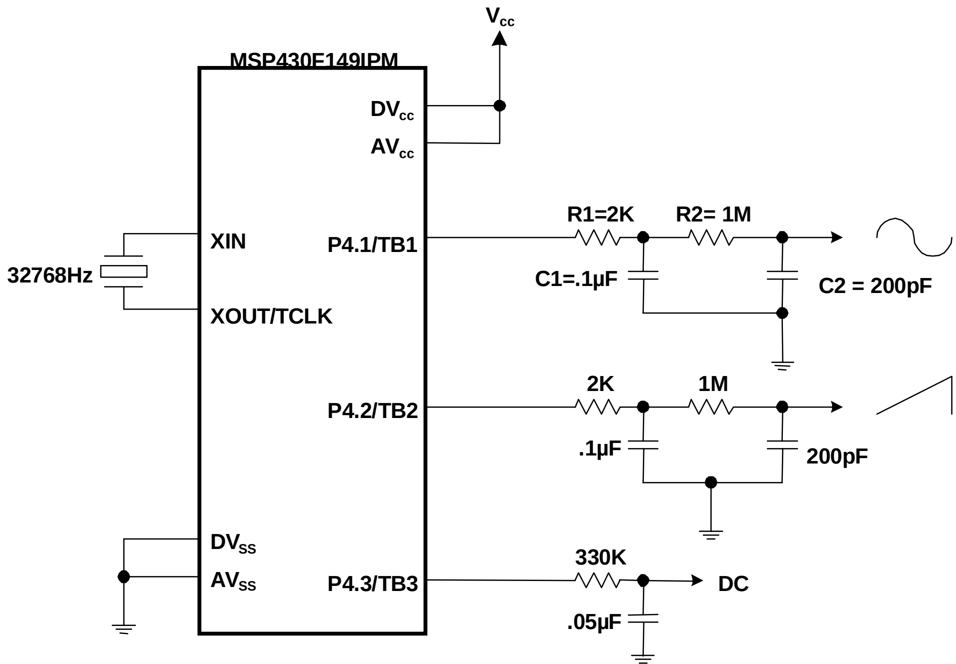

Figure 3. Circuit Diagram

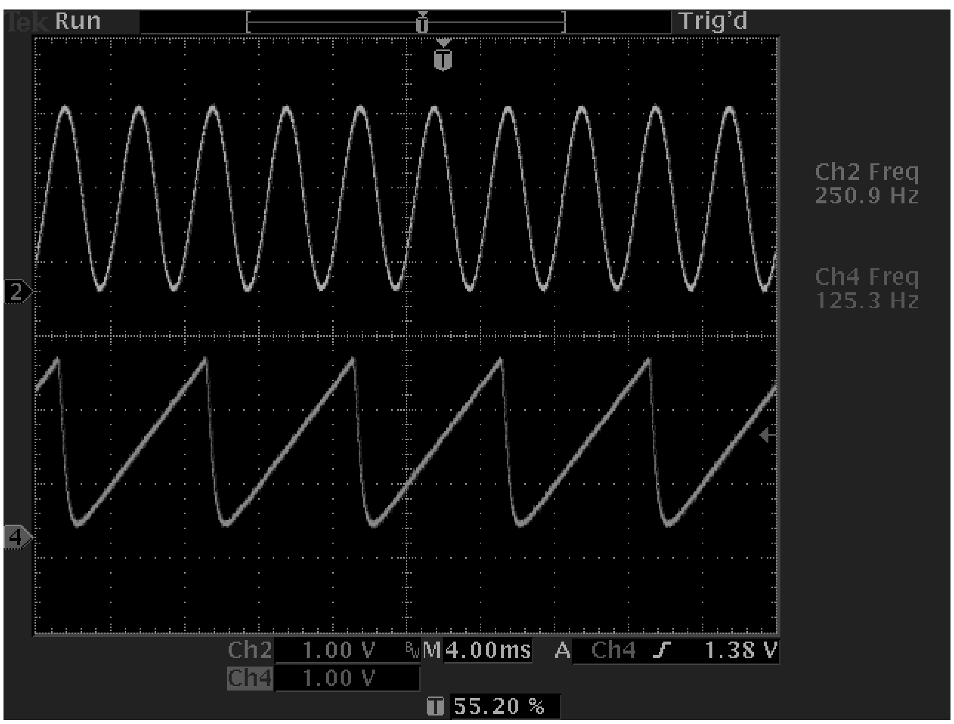

Figure 4. AC Signals

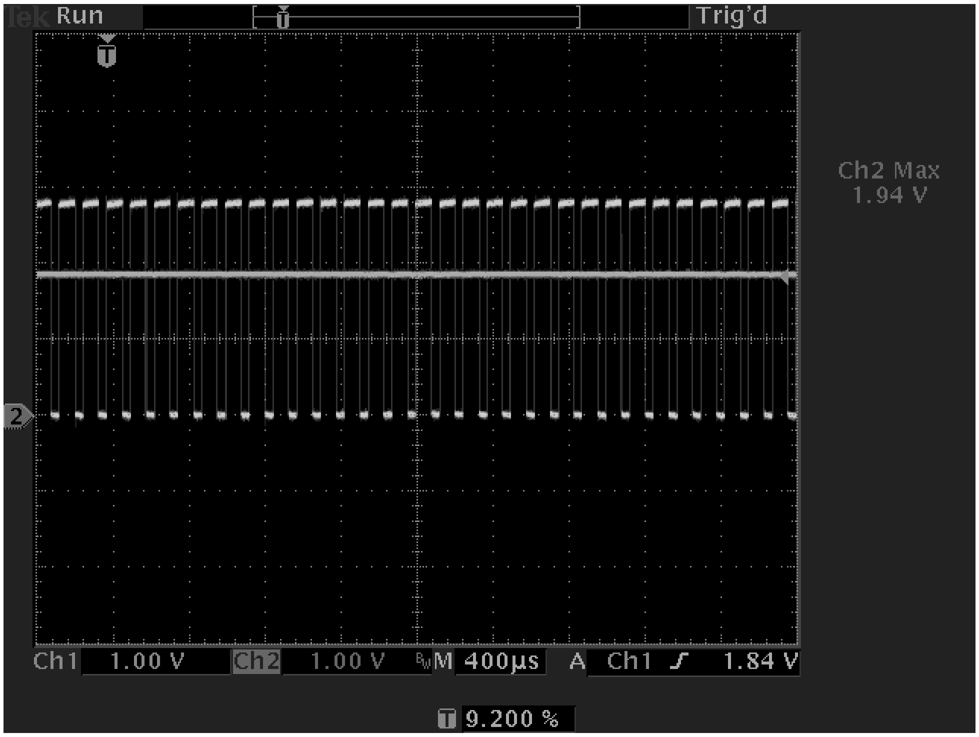

Figure 5. DC Signal

The sine wave produced by this example used 32 samples per cycle. The sample values are contained in a table at the beginning of the program. A pointer is used to point to the next value of the sine table so that at the end of each PWM cycle, the new value of the sine wave is written to the capture/compare register of the PWM timer.

The ramp in this example does not require a table of data values. Rather, it was generated by simply incrementing the duty-cycle each cycle of the PWM signal until the maximum was reached and then starting over at the minimum duty-cycle. This gradual increase in PWM signal duty-cycle results in a ramp voltage when it is filtered.

The dc level in this example was set by simply setting the value of the PWM signal duty-cycle and never changing it. The dc level is directly proportional to the value of the duty-cycle of the PWM signal. So, since the duty-cycle of the PWM signal on TB3 does not change, when it is filtered by the RC network, a dc value results.

2.5 Filter Requirements

The reconstruction filters used for each signal in this example are shown above in Figure 3. The filter for the ac signals is a simple two-pole stacked-RC filter. It was chosen for it’s simplicity and lack of active components for low power designs. This necessitates a higher sampling rate than would be required if the filter were a higher order. With the type of filter shown above, it is recommended to use at least 16x oversampling for the DAC.

The cutoff frequency of the filter can be calculated by:

$$F_C = \frac{1}{2\pi RC} \text{ where } R1C1 = R2C2 = RC$$

The filter gives better response when R2 >> R1. Also, setting the cutoff frequency too close to the bandwidth edge will cause a fair amount of attenuation. To reduce the amount of attenuation caused by the filter, set the cutoff frequency above the bandwidth edge, but << than the frequency of the PWM signal.

The filter used for the dc value is used for charge storage rather than ac signal filtering. Therefore, a simple, single-pole RC filter is used.

2.6 Adding the DC and AC Signals Together

The signals produced by the PWM DAC may be added together. The circuit in Figure 6 shows the dc offset of 2/3 V_{cc} being added to the sine wave to produce an offset sine wave. In addition, the sine wave was attenuated by approximately 2/3 to allow it to be moved up or down on the dc offset without clipping the summing amplifier. This produced the $1V_{PP}$ sine wave sitting on a dc offset of approximately 2V shown below in Figure 7. The PWM signal is shown in the background for reference.

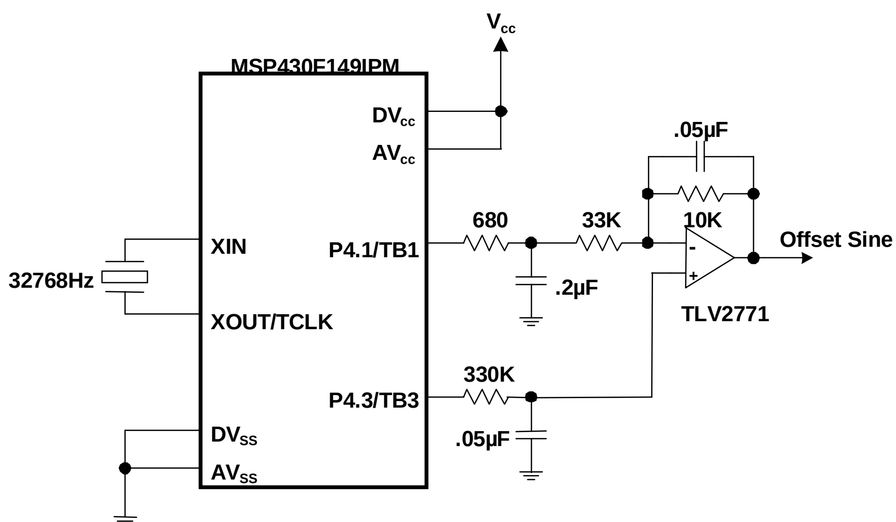

Figure 6. Summing Circuit

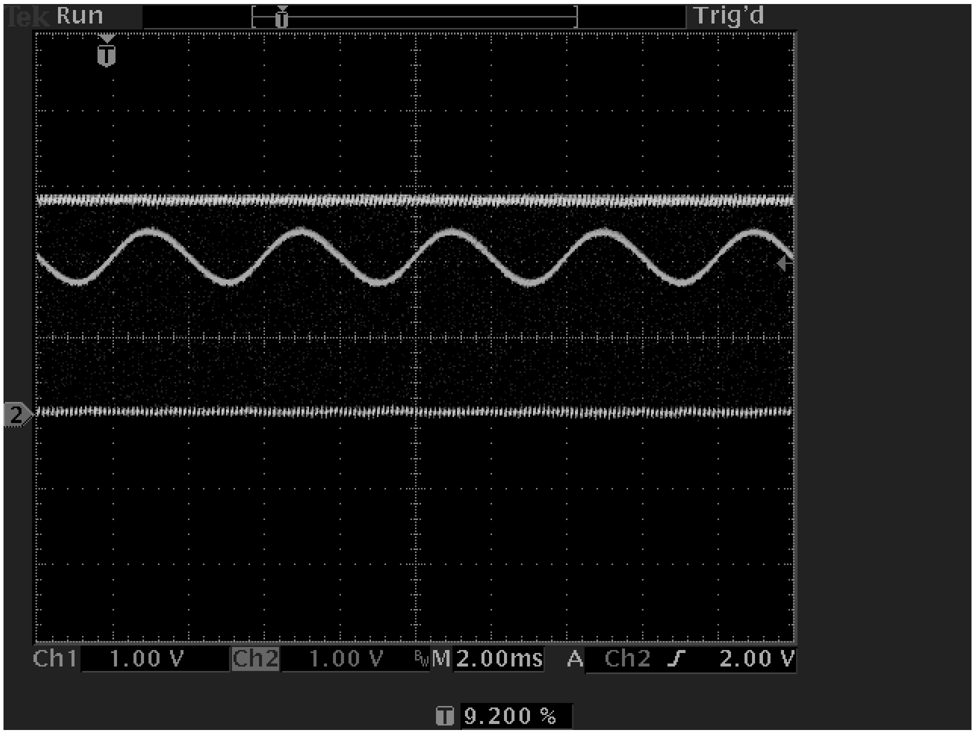

Figure 7. Offset Sine Wave

Adding the signals as shown in this example provides the flexibility to adjust the ac and dc signals separately and easily via the PWM outputs. For example, the sine wave may be kept constant, while the dc offset may be moved up or down simply by changing the PWM duty-cycle used to create the dc value. Note that the filter for the sine wave changed slightly. Since the summing amplifier was added to the circuit to add the offset to the sine wave, the filter required for the sine wave was integrated into the amplifier circuit. Other active filters and summing amplifier circuits could also be used to achieve the same results.

3 Software Listing and Description

The complete software listing is shown in Appendix A. The software flow is shown below in Figure 8.

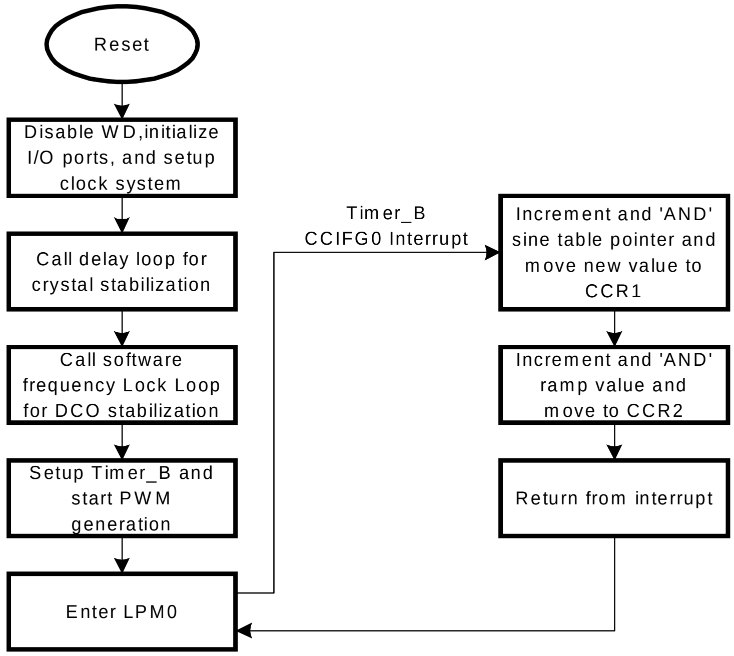

Figure 8. Software Flow

After a reset, the watchdog timer is stopped, output ports are configured, and the clock system is set up. Next, a delay is called to allow the 32768Hz crystal to stabilize because it must be stable to calibrate the DCO. Next the DCO calibration routine is called. After the DCO calibration is complete, Timer_B, CCR1 and CCR2 are setup for PWM generation and the timer is started. Finally, the MSP430 is put into low power mode 0 (LPM0) to conserve power. The CPU is awakened to handle each CCIFG0 interrupt, then reenters LPM0.

3.1 DCO Calibration

The DCO on MSP430 devices is essentially an RC-type oscillator and exhibits RC-oscillator characteristics such as inaccurate frequency settings and drift. However, because the DCO is digitally controllable, it can be tuned to an accurate frequency using a stable, known frequency source such as the 32768Hz crystal oscillator. Some MSP430 devices have digital logic known as a frequency lock loop (FLL) to perform this function automatically, but the MSP430x1xx devices do not. See the clock system chapter of the appropriate family user’s guide for a more detailed discussion on the DCO and MSP430 clock systems.

Because the MSP430x1xx devices do not contain an FLL, the DCO must be calibrated with software and an external frequency source to produce a known, stable frequency. The software shown in this example implements the FLL in software to calibrate the DCO. It works by comparing the number of DCO clock cycles in a single 32768Hz clock cycle. Actually, in this example, the 32768Hz clock (ACLK) is divided by 4, so the software FLL counts the number of DCO clock cycles in one-fourth of an ACLK cycle. Then, that number is compared to the desired number and the DCO is either incremented or decremented one step based on the results and the count is taken again. This routine is continued until the desired comparison value is reached. When the desired value is reached, the DCO is calibrated to the desired frequency.

This example shows executing the software FLL only once because the hardware used was not subject to environmental changes, V_{cc} changes, or extended run-time lengths. However, in a real-world application, the software FLL routine would need to be executed periodically to keep the DCO in calibration given these types of effects and the general drift of RC-type oscillators. See the application note *Controlling the DCO Frequency of the MSP430x11x* (literature number SLAA074) for more details on implementing a software FLL.

An alternative to the DCO calibration routine would be to use an external crystal of the required PWM timer frequency. This is possible on the 13x and 14x devices because of the XT high-speed crystal oscillator. If this approach were used, SMCLK would need to be sourced from the XT crystal oscillator instead of the DCO. See chapter 7 of the *MSP430x1xx Family User's Guide*, literature number SLAU049, for more details on the clock system of the 1xx family. See also device data sheet for applicable limits on frequency of operation.

References

1. MSP430x13x/14x data sheet (SLAS272)
2. *MSP430x1xx Family User's Guide* (SLAU049)
3. *MSP430x3xx Family User's Guide* (SLAU012)
4. *Controlling the DCO of the MSP430x11x* (SLAA074)

Appendix A. Software Listing

```
; THIS PROGRAM IS PROVIDED "AS IS". TI MAKES NO WARRANTIES OR
; REPRESENTATIONS, EITHER EXPRESS, IMPLIED OR STATUTORY,
; INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
; FOR A PARTICULAR PURPOSE, LACK OF VIRUSES, ACCURACY OR
; COMPLETENESS OF RESPONSES, RESULTS AND LACK OF NEGLIGENCE.
; TI DISCLAIMS ANY WARRANTY OF TITLE, QUIET ENJOYMENT, QUIET
; POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY
; INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO THE PROGRAM OR
; YOUR USE OF THE PROGRAM.
;
; IN NO EVENT SHALL TI BE LIABLE FOR ANY SPECIAL, INCIDENTAL,
; CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY
; THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED
; OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT
; OF THIS AGREEMENT, THE PROGRAM, OR YOUR USE OF THE PROGRAM.
; EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO, COST OF
; REMOVAL OR REINSTALLATION, COMPUTER TIME, LABOR COSTS, LOSS
; OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, OR LOSS OF
; USE OR INTERRUPTION OF BUSINESS. IN NO EVENT WILL TI'S
; AGGREGATE LIABILITY UNDER THIS AGREEMENT OR ARISING OUT OF
; YOUR USE OF THE PROGRAM EXCEED FIVE HUNDRED DOLLARS
; (U.S.$500).
;
; Unless otherwise stated, the Program written and copyrighted
; by Texas Instruments is distributed as "freeware".  You may,
; only under TI's copyright in the Program, use and modify the
; Program without any charge or restriction.  You may
; distribute to third parties, provided that you transfer a
; copy of this license to the third party and the third party
; agrees to these terms by its first use of the Program. You
; must reproduce the copyright notice and any other legend of
; ownership on each copy or partial copy, of the Program.
;
```

```
; You acknowledge and agree that the Program contains
; copyrighted material, trade secrets and other TI proprietary
; information and is protected by copyright laws,
; international copyright treaties, and trade secret laws, as
; well as other intellectual property laws.  To protect TI's
; rights in the Program, you agree not to decompile, reverse
; engineer, disassemble or otherwise translate any object code
; versions of the Program to a human-readable form.  You agree
; that in no event will you alter, remove or destroy any
; copyright notice included in the Program.  TI reserves all
; rights not specifically granted under this license. Except
; as specifically provided herein, nothing in this agreement
; shall be construed as conferring by implication, estoppel,
; or otherwise, upon you, any license or other right under any
; TI patents, copyrights or trade secrets.
;
; You may not use the Program in non-TI devices.

NAME        PWMDAC
;****************************************************************************
;  PWM DAC Demonstratin Program
;  Generate a 250Hz sine wave using PWM timer Timer_B.
;
;  Description: This program demonstrates the usage of a PWM timer together
;  with external filters to implement a DAC.  The program shows how to
;  create a 250Hz sine wave, a 125Hz ramp, and a DC level with Timer_B.
;  Timer_A could also be used in the same manner.  A sine table holds the
;  sample values for the sinusoid.  To create the ramp, the PWM value is
;  simply incremented. The DC level is created by storing charge on an
;  RC network using a PWM output to provide the charge.  The value of the DC
;  voltage directly corresponds to the duty cycele of the PWM signal.  After
;  initialization, the CPU is put into LPM0.  It remains there until the
;  CCIFG0 interrupt from Timer_B wakes it up.  In the Timer_B ISR, the next
;  value for the sinusoid is loaded into CCR1 and the ramp value is incremented
;  and loded into CCR2. Upon return form the ISR, the CPU goes back into LPM0.
```

```
;
;  Mike Mitchell
;  MSP430 Applications
;  Texas Instruments, Inc
;  November 2000
;
;****************************************************************************
#include    "MSP430X14x.H"                  ; Include Standard Defs

Delta       EQU     250                     ; Delta = Target DCO/8192
                                            ; Target DCO frequency = 2.048MHz
                                            ; This value is used in the
                                            ; software FLL routine to
                                            ; calibrate the DCO frequency
                                            ; using the 32768Hz oscillator
                                            ; as a reference.  For more
                                            ; information on stabilizing
                                            ; the DCO or the FLL routine
                                            ; see the application report
                                            ; titled "Controlling the DCO
                                            ; frequency of the MSP430x11x"
                                            ; Literature number SLAA074
;------------------------------------------------------------------------------
            RSEG    CODE
;------------------------------------------------------------------------------
Sine_Tab    DW      255                     ; Sine Table.  These are the count
            DW      254                     ; values in decimal that will
            DW      246                     ; go into TBCCR1 to change the
            DW      234                     ; PWM duty cycle.
            DW      219                     ; Must use words instead of bytes
            DW      199                     ; because must move words into
            DW      177                     ; TB registers.
            DW      153                     ; Don't use a '0' as a sample value
            DW      128                     ; The timer will glitch.
            DW      103
```

```
            DW      79
            DW      57
            DW      37
            DW      22
            DW      10
            DW      2
            DW      1
            DW      2
            DW      10
            DW      22
            DW      37
            DW      57
            DW      79
            DW      103
            DW      128
            DW      153
            DW      177
            DW      199
            DW      219
            DW      234
            DW      246
            DW      255
;-----------------   Code Starts Here   ---------------------------------------
RESET       mov     #02FEh,SP               ; Initialize stackpointer

StopWDT     mov     #WDTPW+WDTHOLD,&WDTCTL  ; Stop WDT

SetupP4     bis.b   #00Eh,&P4SEL            ; Select TB1, TB2, TB3 instead of
            bis.b   #00Eh,&P4DIR            ; P4.x, and set as outputs

SetupBC     mov.b   #0A6h,&BCSCTL1          ; ACLK is divided by 4. RSEL=6,
                                            ; no division for MCLK or SMCLK,
                                            ; DCO sources MCLK and SMCLK.
                                            ; XT2 is off.
                                            ; NOTE: To determine the value of
```

```
                                        ; Rsel for a desired DCO frequency,
                                        ; refer to the DCO table in the
                                        ; datasheet.

            call    #Delay              ; Delay for crystal stabilization.
                                        ; Need to put a delay here because
                                        ; the 32768Hz crystal is used as
                                        ; a reference to stabilize the DCO
                                        ; frequency.  Therefore, the 32768
                                        ; crystal needs to be stable.

            call    #SW_FLL             ; Call the routine to Stabilize
                                        ; the DCO clock.

            call    #TB_SETUP           ; Setup Timer_B for PWM generation

            clr     R15                 ; R15 and R14 used as pointers
            clr     R14                 ; to the sine table and to hold the
                                        ; ramp value after the DCO is
                                        ; stabilized

            eint                        ; Enable interrupts

            bis     #LPM0,SR            ; Put CPU to sleep.
                                        ; This is the end of the program
                                        ; except for handling the CCIFG0
                                        ; interrupt, which is where the
                                        ; PWM values are updated.
;------------------------------------------------------------------------------
Delay;      Software delay for crystal stabilization
;------------------------------------------------------------------------------
            mov     #0004h,R15
L1          mov     #0FFFFh,R14         ; This should ideally be about a sec.
L2          dec     R14                 ;
            jnz     L2                  ;
```

```
                                        ;
            dec     R15                 ;
            jnz     L1                  ;
            ret                         ;
                                        ;
;-----------------------------------------------------------------------------
SW_FLL;    Subroutine: Stabilizes DCO frequency.
           ; This routine uses the 32768Hz crystal oscillator as a reference
           ; frequency to stabilize and trim the DCO oscillator to the desired
           ; frequency of 2.048MHz.  This is only required in applications that
           ; need a specific DCO frequency and for MSP430 devices that do not
           ; have an FLL module.  See the MSP430x3xx and MSP430x1xx Family
           ; User's Guides (literature numbers SLAU012 and SLAU049 repsecitvely)
           ; for more information on the clock systems employed on MSP430 devices
           ;
           ; The routine works by counting how many DCO clock cycles are inside
           ; of one ACLK cycle (actually 1/4 ACLK cycle because ACLK is divided
           ; by 4).  Timer_A is used to determine the number of DCO clocks and
           ; this value is then compared to the target value (Delta).  If the
           ; number is too high, the DCO is decremented.  If the number is too
           ; low, the DCO is incremented. The comparison is then made again.
           ; This process is repeated until the target value is reached.  When
           ; the target value is obtained, the DCO is oscillating at the desired
           ; frequency.  See the application report "Controlling the DCO
           ; Frequency of the MSP430x11x devices", literature number SLAA074,
           ; for more application information related to controlling the DCO.
           ;
           ; This routine is run only once in this example, but in an
           ; application it would likely need to be run on a periodic
           ; basis to make sure the DCO remained calibrated.
;-----------------------------------------------------------------------------
            clr     R15                 ;

Setup_TA    mov     #TASSEL1+TACLR,&TACTL   ; SMCLK clocks TA
Setup_CC2   mov     #CCIS0+CM0+CAP,&CCTL2   ; Define CCR2,CAP,ACLK
            bis     #MC1,&TACTL             ; Start timer_A: Continous Mode
```

```
Test_DCO    bit     #CCIFG,&CCTL2          ; Test capture flag
            jz      Test_DCO               ;
            bic     #CCIFG,&CCTL2          ; Clear capture flag
                                           ;
AdjDCO      mov     &CCR2,R14              ; R14 = captured SMCLK
            sub     R15,R14                ; R14 = capture difference
            mov     &CCR2,R15              ; R15 = captured SMCLK
            cmp     #Delta,R14             ; Delta = SMCLK/(32768/4)
            jlo     IncDCO                 ;
            jeq     DoneFLL                ;
DecDCO      dec.b   &DCOCTL                ;
            jmp     Test_DCO               ;
IncDCO      inc.b   &DCOCTL                ;
            jmp     Test_DCO               ;
DoneFLL     clr     &CCTL2                 ; Stop CCR2
            clr     &TACTL                 ; Stop timer_A
            ret                            ; Return from subroutine
;-------------------------------------------------------------------------------
TB_SETUP;    Subroutine: Setup Timer_B for PWM generation
;-------------------------------------------------------------------------------
            mov     #TBSSEL1+TBCLR,&TBCTL  ; SMCLK clocks TB.
            mov     #CCIE,&TBCCTL0         ; Set CCR0 in compare mode, enable
                                           ; it's interrupt
            mov     #0FFh,&TBCCR0          ; Put 255d in CCR0.  This will set
                                           ; the period of the PWM output to
                                           ; 256 counts(8-bits).  This gives
                                           ; an 8-bit DAC.
            mov     #02E0h,&TBCCTL1        ; Set CCRx in compare mode, disable
            mov     #02E0h,&TBCCTL2        ; interrupt, set outmode to '7' which
            mov     #02E0h,&TBCCTL3        ; is reset/set.  EQU0 sets the output
                                           ; EQU1 will reset it. Set the load
                                           ; condition for the compare latch
                                           ; to be when the counter counts to
                                           ; 0.
            mov     #Sine_Tab,&TBCCR1      ; Load first sample value into CCR1
```

```
            mov     #01h,R14                ; Load inital ramp value into R14.
            mov     #0AAh,&TBCCR3           ; This is for the DC value.  It will
                                            ; result in a voltage of approximately
                                            ; 2/3 Vcc because #0AAh is 2/3 of
                                            ; #0FFh.
            bis     #MC0,&TBCTL             ; Start timer_B in up mode

            ret
;------------------------------------------------------------------------------
TB_ISR;    Timer_B ISR: changes the value in the CCR1 and CCR2 registers to
;          vary the PWM for the sinusoid and the ramp.  The CCR3 value is left
;          unchanged for the DC signal.
;------------------------------------------------------------------------------
            incd    R15                     ; Increment the pointer R15 to
                                            ; to point to next word of sine
                                            ; table.  Must increment by 2
                                            ; because the sine table is words
                                            ; not bytes.
            and     #03Fh,R15               ; ANDing with 03Fh gives an
                                            ; effective modulo 32 counter for
                                            ; pointing to each value in the
                                            ; sine table
            mov    Sine_Tab(R15),&TBCCR1    ; Move new sine value to CCR1

            add    #04h,R14                 ; Increment ramp value.
                                            ; Changing the step size in R14
                                            ; will change the frequency of
                                            ; the ramp.
            and    #0FFh,R14                ; And off unwanted bits
            mov    R14,&TBCCR2              ; Move new ramp value to CCR2

            reti                            ; return with interrupts enabled
```

```
;-----------------------------------------------------------------------------
            COMMON  INTVEC                  ; MSP430x14x interrupt vectors
;-----------------------------------------------------------------------------
            ORG     TIMERB0_VECTOR
            DW      TB_ISR                  ; CCIFG0 interrupt
            ORG     RESET_VECTOR
            DW      RESET                   ; POR, ext. Reset, Watchdog
            END
```

Exit | Route Map

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.