Subsystem Design

PWM DAC

1 Description

The PWM DAC subsystem example demonstrates how to use an MSPM0 timer, and a simple RC filter to create a PWM DAC. The example software creates a 10-bit DAC with a PWM frequency of 31250Hz. The duty cycle of the PWM signal updates continuously to create a sinusoidal waveform at the filter output. While the MSPM0Gx50x devices include a 12-bit DAC, and the internal comparators include 8-bit reference DACs that can buffered through the OPA, a PWM DAC allows for the generation of analog output voltages on devices lacking these peripherals, or just for additional DAC outputs whenever needed. Figure 1-1 shows the block diagram for a single PWM DAC.

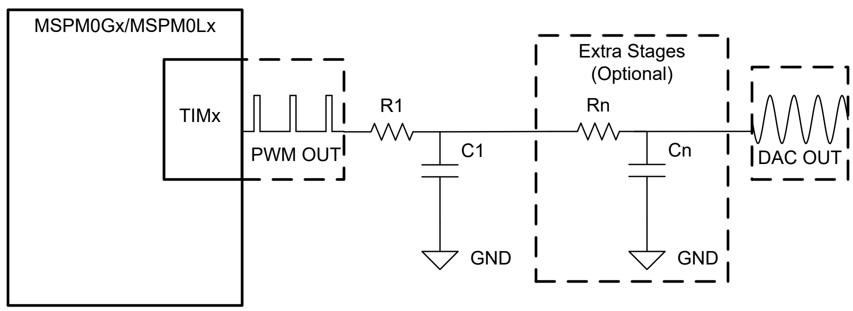

Figure 1-1. Subsystem Functional Block Diagram

2 Required Peripherals

This application requires the PWM peripheral and a TIMGx instance with shadow capture compare registers.

Table 2-1. Required Peripherals

Subblock Functionality	Peripheral Use	Notes
PWM	TIMGx	Example uses TIMG4 shadow registers to avoid signal glitching

3 Compatible Devices

Based on the requirements in Table 2-1, the compatible devices are listed in Table 3-1. The corresponding EVM can be used for quick evaluation.

Table 3-1. Compatible Devices

Compatible Devices	EVM
MSPM0Lx	LP-MSPM0L1306
MSPM0Gx	LP-MSPM0G3507

4 Design Steps

1. Configure the PWM to use shadow registers and interrupts.
2. Configure the PWM frequency for the desired DAC resolution.
3. Determine the number of samples needed to adjust the duty cycle. This subsystem example uses 128 samples stored in an array.
4. Cycle through the sample array. This example increments the array index during the associated ISR, and loads a new compare value to change the duty cycle of the PWM.
5. Design a low-pass filter for the PWM output, to create the analog voltage. This example uses a single pole RC filter.

5 Design Considerations

1. **PWM frequency:** The PWM frequency is related to the DAC resolution by:

$$2^N = \frac{f_{CLOCK}}{f_{PWM}} \quad (1)$$

   where

   - f_{CLOCK} is the clock frequency of the timer
   - f_{PWM} is the output PWM frequency
   - N is the duty-cycle resolution of the PWM DAC in bits.

   This subsystem example uses either a 32MHz clock frequency or a 16MHz clock frequency to create a 10-bit DAC. Table 5-1 details some example PWM DAC resolutions based on clock and PWM frequencies.
2. **PWM configuration:** This application configures the Timer for edge-aligned PWM, and sets the capture compare updated value to take effect after the zero event.
3. **Synchronization of duty cycle update:** Shadow registers are used to prevent missed counter compare value updates. This is done in MSPM0 by enabling the shadow load functionality of an appropriate timer instance. This allows the duty cycle to be updated while the timer is running, without worry of a glitch in duty cycle output.
4. **PWM Interrupt configuration:** Here the timers are configured in down-count mode, so the interrupt is configured to occur a capture or compare down event. If updating the duty cycle on the very next cycle is desired, using the capture compare down or up interrupt helps make sure the captured value can be updated before the next load event or zero event. Any other system interrupt can be used as well, and needs to be synchronized by the enabling of the shadow load capability.
5. **Sample array:** When outputting a signal or waveform greater than the number of samples results in a higher resolution output. The samples values need to be formatted to align with the resolution of the PWM DAC.
6. **Filter design:** A basic RC filter is usually enough to filter the PWM output. The filter cutoff frequency needs to be at least an order of magnitude below the PWM frequency.

   If better filtering of the PWM edges is desired, a higher order or more complex filter can be employed.

Table 5-1. PWM DAC Resolutions

f_{CLOCK}	f_{PWM}	N
32MHz	125kHz	8
32MHz	31.3kHz	10
32MHz	7.8kHz	12
16MHz	62.5kHz	8
16MHz	15.6kHz	10
16MHz	3.9kHz	12

6 Software Flow Chart

Figure 6-1 shows the software flow chart for this subsystem example and shows the software flow of the ISR used to create the PWM DAC in this example.

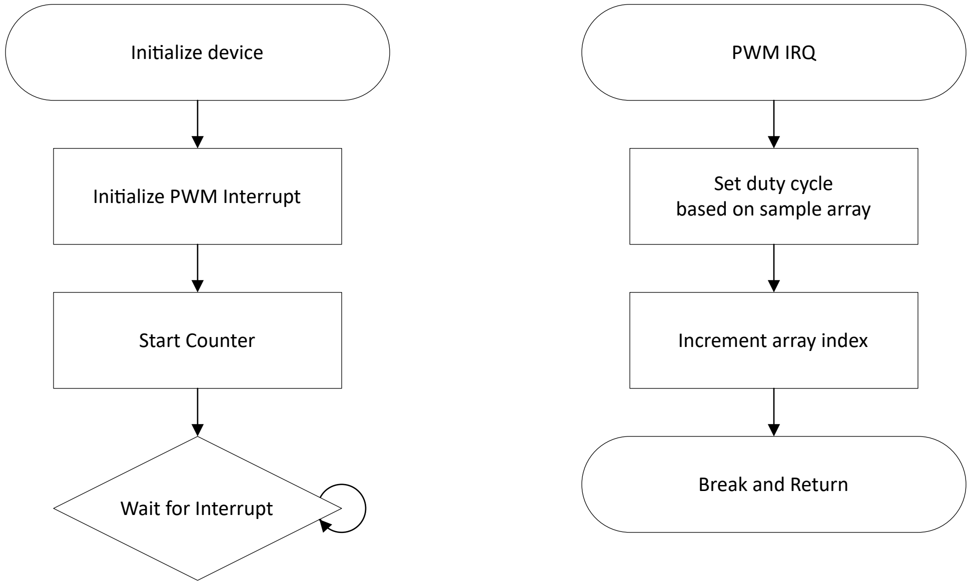

Figure 6-1. Application Software Flow Chart

7 Application Code

This application makes use of the TI System Configuration tool (SysConfig) graphical interface to generate the configuration code for the device peripherals. Using a graphical interface to configure the device peripherals streamlines the application prototyping process.

This example application code uses an array of 128 samples to continuously change the duty cycle of a single PWM output. This creates a sinusoidal wave after filtering. The duty cycle is changed through timer interrupt and shadow registers. The interrupt is generated on a counter compare down event. During this interrupt, the next counter compare value in the array index is set and ready to be loaded on the next TIMCLK cycle after the timer reaches zero. This helps prevent the application from missing any PWM duty cycle changes, which can cause glitches in the final output.

```
void PWM_0_INST_IRQHandler(void){
    switch (DL_TimerG_getPendingInterrupt(PWM_0_INST)){
        case DL_TIMERG_IIDX_CC0_DN: /* Interrupt on CC0 Down Event */
            /*Set new Duty Cycle based on sine array sample value */
            DL_TimerG_setCaptureCompareValue(PWM_0_INST, gSine128[gSineCounter%128],
                DL_TIMER_CC_0_INDEX);

            /* Increment gSineCounter value */
            gSineCounter++;

            break;
        default:
            break;
    }
}
```

8 Results

Figure 8-1 displays the PWM digital output compared to the filter output when using a 32MHz clock frequency. The top half shows a zoomed view of half the final sine wave period to clearly display the duty cycle changes in the PWM signal. The bottom half shows a more zoomed out view to clearly display the final sine wave output.

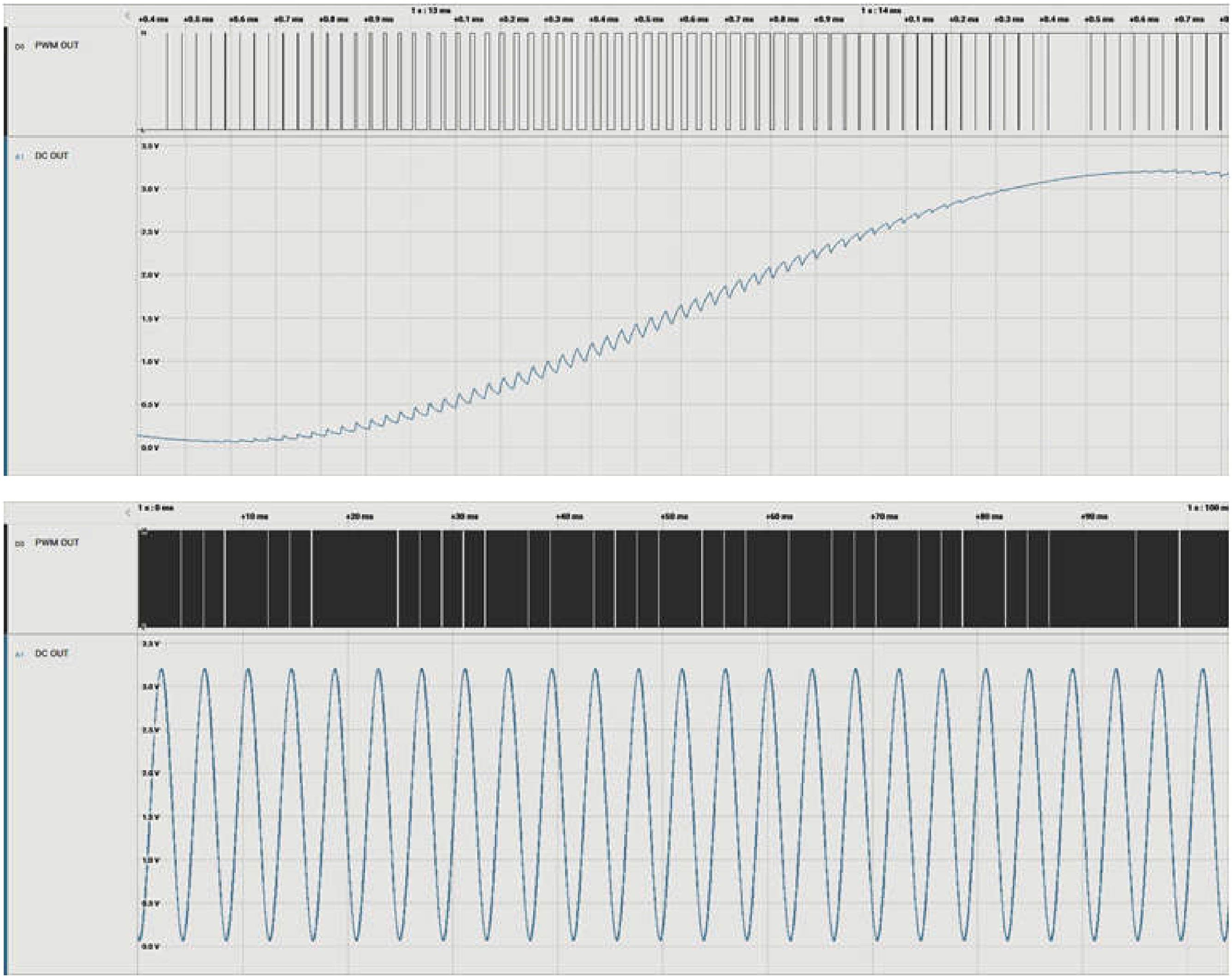

Figure 8-1. Results

9 Additional Resources

- Texas Instruments, Download the MSPM0 SDK
- Texas Instruments, Learn more about SysConfig
- Texas Instruments, MSPM0L LaunchPad™
- Texas Instruments, MSPM0G LaunchPad™
- Texas Instruments, MSPM0 Academy
- Texas Instruments, PWM DAC Using MSP430 High-Resolution Timer Application Note
- Texas Instruments, Using PWM Timer_B as a DAC Application Note
- Texas Instruments, Voice Band Audio Playback Using a PWM DAC

10 E2E

See TI's E2E™ support forums to view discussions and post new threads to get technical support for utilizing MSPM0 devices in designs.

11 Trademarks

LaunchPad™ and E2E™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.