Implementing Pulse-Width Modulation through MSP430 Timers

Roy Dong

11/19/2010

Executive Summary: Pulse-width modulation has a variety of uses, and is particularly useful because it allows digital sources, such as microcontrollers, to convey values between 0 and 1. This application note presents a brief discussion of pulse-width modulation, as well as how to code pulse-width modulation using the timer registers on a MSP430G2231 microprocessor using C. In closing, I provide the two applications of pulse-width modulation on the DRV8412 motor driver card: driving a brushed DC motor and communicating analog values.

Keywords: MSP430, timer, pulse-width modulation

Introduction

Pulse-width modulation (PWM) is a method by which digital circuit elements can output analog values using only high and low voltage signals. This is achieved by alternating between high and low at the correct intervals to achieve a signal with an equivalent DC voltage to the desired analog value. The fraction of the period in which the signal is high is known as the duty cycle. PWM signals have a variety of uses. This application note will discuss two possible uses, both implemented on the Texas Instruments (TI) DRV8412 motor driver card: 1) driving a brushed DC motor, and 2) communicating an analog value to a test point.

The information presented in this application note is an implementation of PWM on the TI MSP430G2231 microcontroller in C. This is one of the microcontrollers featured in TI's recently released MSP430 LaunchPad (MSP-EXP430G2). More specifically, this application note will cover how to code PWM on the MSP430 LaunchPad, as well as possible uses for the DRV8412 motor driver card.

Coding PWM through Software

Pulse-width modulation is done on the MSP430 through the timer. A timer on the MSP430 increases a register by one every clock cycle on the MSP430. There are four options for the MSP430 timer, pictured in Table 1 below. MC_0 disables the timer. MC_1 counts from 0x0000 to the value stored in the CCR0 register, resets to 0x0000, and repeats. MC_2 counts from 0x0000 to 0xFFFF, resets to 0x0000, and repeats. MC_3 counts from 0x0000 to the value stored in the CCR0 register, counts down from this value to 0x0000, and repeats. Graphical representations of MC_1 through MC_3 are attached as Figure 1 through Figure 3, which are taken from TI's MSP430x2xx User's Guide.

MC_0	Stop
MC_1	Up to CCR0
MC_2	Continuous Up
MC_3	Up/Down

Table 1: Timer modes

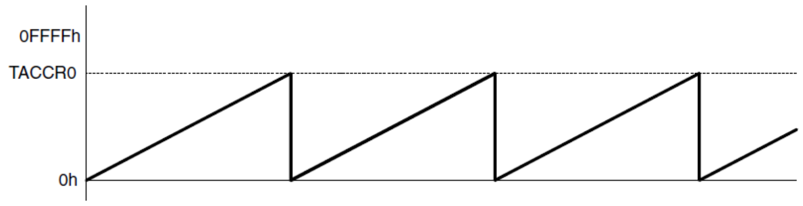

Figure 1: Up to CCR0

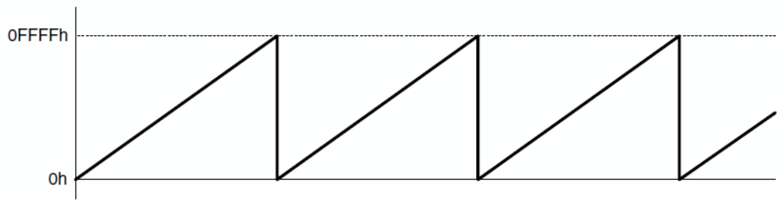

Figure 2: Continuous Up

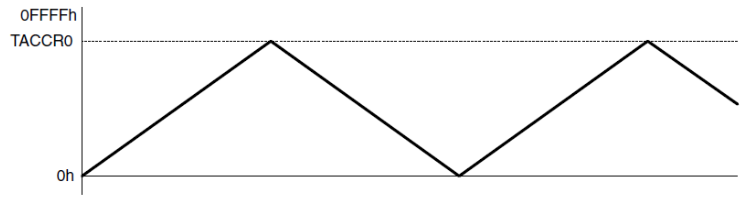

Figure 3: Up/Down Mode

To program the timer in a certain mode, we have to use the control register on the MSP430. For example, if we wish to program Timer A to count 1000 values, we use the TACTL register. TACTL stands for"Timer A Control". Also, since the count begins from 0x0000, to count 1000 values, we would set CCR0 to 999. The code would be as follows:

```
CCR0 = 1000 - 1;

TACTL = MC_1;
```

Now, we can set which pin to output the PWM signal. For example, from pg. 6 of the MSP430G2231 datasheet, pin 4 is"P1.2/TA0.1/A2". We want to use this pin as"TA0.1". To do this, we have to set second bit of both the P1SEL and P1DIR registers accordingly. Generally, to set the Px.y pin, we must set the PxSEL and PxDIR registers accordingly at bit y. A 0 in PxDIR is input; a 1 is output. A 0 in PxSEL means general purpose input/output, while a 1 in PxSEL reflects a special purpose based on the pin. For example, to use the timer, we would set the appropriate bits in PxSEL and PxDIR to 1. To use the ADC converter (A2), we would set PxSEL to 1 and PxDIR to 0. The code to set the pin as a timer is as follows:

```
P1DIR |= BIT2;

P1SEL |= BIT2;
```

We use the |= operator so the bits we do not wish to change remain the same.

Next, we must set the PWM mode through the CCTLx register, where x is the port number of the pin. For example, to use "P1.2/TA0.1/A2", we would use CCTL1. The modes are pictured below in Table 2.

OUTMOD_0	PWM Disabled
OUTMOD_1	Set
OUTMOD_2	PWM Toggle/Reset
OUTMOD_3	PWM Set/Reset
OUTMOD_4	Toggle
OUTMOD_5	Reset
OUTMOD_6	PWM Toggle/Reset
OUTMOD_7	PWM Reset/Set

Table 2: PWM modes

If the mode used has only one description, e.g. "Set", then the pin performs this action when it reaches CCR1. So, if the PWM is in OUTMOD_1, the register will go high when the timer reaches CCR1 and stay high until it is reset manually. If the mode used has two descriptions, the first description is performed when CCR1 is reached, and the second is performed when CCR0 is reached. For example, if we are in OUTMOD_3, the register will set at CCR1 and then reset at CCR0.

To use the PWM to achieve a duty cycle of 20%, we would use the following code:

```
CCTL1 = OUTMOD_7;

CCR1 = 200 - 1;
```

Since we are in MC_2, which counts up to CCR0, the pin is set high just before 0x0000, when the timer register equals CCR0. Then, the pin outputs high until CCR1 is reached, when it resets. The timer, in our example code, will count from 0 to 999. From 0 to 199, the register will be high, and from 200 to 999, the register will be low. This achieves a 20% duty cycle.

Results

PWM has many possible applications. TI's DRV8412 motor driver card provides two examples of how PWM can be used. First, the PWM signal can be used to drive a brushed DC motor. In this case, increasing the duty cycle from 0% to 100% increases the speed and torque of the motor. Sensing current feedback through the DRV8412 can also allow for advanced PI control of the brushed DC motor.

Also, the DRV8412 has special PWM DACs. This allows a microcontroller to communicate analog values through digital signals sent to the DRV8412. At first, it may seem that DACs should merely provide an analog version of the pulse-width modulated signal. However, one must remember that a DAC is essentially a low-pass filter, and if the frequency of the pulses is significantly higher than the cut-off frequency of the DAC, the DAC will just output an analog value equal to the average DC voltage. The output of this DAC will be: duty cycle * Vcc.

Beyond the DRV8412, there are many other uses for PWM signals. PWM allows us to digitally create analog voltage levels for control functions and power supplies. Also, PWM can create analog signals for arbitrary waveforms, including music and speech. With the ability to code PWM on the MSP430, a broad set of options for analog signal generation through digital sources are available.

References

"Timers and Clocks and PWM! Oh My!" NJC's MSP430 LaunchPad Blog.

http://www.msp430launchpad.com/2010/07/timers-and-clocks-and-pwm-oh-my.html

Accessed 11/19/2010.

"MSP430G2231 Datasheet." Texas Instruments.

http://focus.ti.com/docs/prod/folders/print/msp430g2231.html Accessed 11/19/2010.

"MSP430x2xx User's Guide." Texas Instruments. http://focus.ti.com/lit/ug/slau144e/slau144e.pdf

Accessed 11/19/2010.

"Wireless Sensor Network and Laboratories." NTU School of Engineering and Applied Science at Walden

University. http://nslab.ee.ntu.edu.tw/courses/wsn-labs-fall-07/ Accessed 11/19/2010.

"MSP430 Interrupts." Dublin Institute of Technology.

http://eleceng.dit.ie/frank/msp430/msp430Interrupts.pdf Accessed 11/19/2010.