ANALOG DEVICES

AN-798
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • Tel: 781/329-4700 • Fax: 781/461-3113 • www.analog.com

Using the PWM to Generate Analog Output on the ADuC702x Family

by Aude Richard

INTRODUCTION

The ADuC702x family integrates a 32-bit ARM7TDMI microcontroller and high precision analog blocks. Depending on the model, up to four 12-bit DACs are available. In some applications where more analog output is required, the PWM can also be used as a low resolution DAC. This application note describes how to generate an extra analog output from a PWM.

THEORY

A typical PWM signal is shown in Figure 1. The base frequency (or switching frequency) is fixed and the pulse width (duty cycle) is variable.

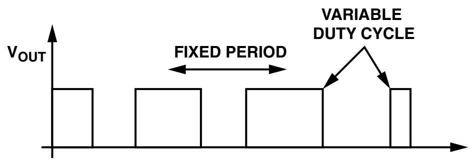

Figure 1. Typical PWM Waveform

The pulse width is proportional to the amplitude of the signal, while the frequency of the waveform is constant.

PWM BLOCK ON THE ADuC702x

The PWM block on the ADuC702x consists of three PWM channels. These three PWMs have a common switching frequency programmable between 343.99 Hz and 11.27 MHz in the PWMDAT0 register.

The switching frequency (f_{PWM}) is calculated as follows:

$$f_{PWM} = f_{CORE}/(2 \times PWMDAT0)$$

The switching frequency is set to 5.5 kHz, or PWMDAT0 = 0x1000. This results in a sine wave at 86 Hz (64 samples per sine wave).

The duty cycle of each channel is independent, therefore the three phases can be used as three independent PWMs. In this application note, one channel of the PWM block, Channel 0, is discussed.

Each PWM channel has two outputs: a high side and a low side. The duty cycle is programmable from 50% to 100% on the high side and 50% to 0% on the low side. The duty cycle of Channel 0 is programmed via the PWMCH0 register. It can be modified at each PWM period, during a synchronization interrupt.

In the following example, a sine wave will be generated using the PWM0H output on P3.0. On the high side output, the duty cycle is only programmable between 50% and 100%, but using the crossover option (in PWMEN MMR) allows switching internally between 0L and 0H resulting in a duty cycle between 0 and 50%.

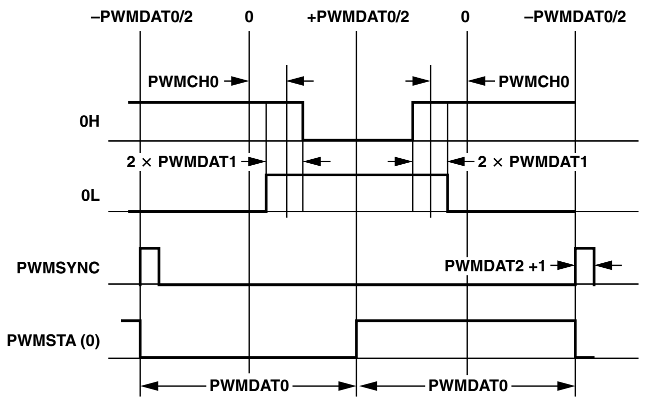

Figure 2. Single Update Mode Timing

The formulas to calculate the value in the PWMA register are as follows:

On the high side:

$$T_{0HH} = PWMDAT0 + 2(PWMCH0 - PWMDAT1) \times t_{CORE}$$

$$T_{0HL} = PWMDAT0 - 2(PWMCH0 - PWMDAT1) \times t_{CORE}$$

$$\text{Or } d_{OH} = 1/2 + (PWMCH0 - PWMDAT1) / PWMDAT0$$

Considering no dead time is being used:

$$PWMCH0 = (d_{OH} - 1/2) \times PWMDAT0$$

On the low side:

$$T_{0LH} = PWMDAT0 - 2(PWMCH0 + PWMDAT1) \times t_{CORE}$$

$$T_{0LL} = PWMDAT0 + 2(PWMCH0 + PWMDAT1) \times t_{CORE}$$

$$\text{Or } d_{OL} = 1/2 - (PWMCH0 + PWMDAT1) / PWMDAT0$$

Considering no dead time is being used:

$$PWMCH0 = (1/2 - d_{OL}) \text{ x } PWMDAT0$$

HARDWARE CONSIDERATION

Conversion of PWM waveforms to analog signal requires a low-pass filter, as shown in Figure 3.

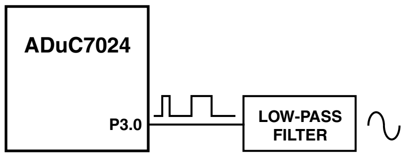

Figure 3. External Filter

For an ac signal, a simple 2-pole stacked RC filter can be used to reconstruct the sine wave. Since the switching frequency in this example is 5.5 kHz and the sine wave required is 86 Hz, the cut of frequency of the filter should be around 550 Hz, 10 times less than the switching frequency of the PWM, but far enough from the bandwidth edge to reduce the amount of attenuation.

The cutoff frequency of the filter can be calculated as follows:

$$FC = 1/2\pi RC$$

For the first filter, R = 2 kΩ and C = 0.2 μF; for the second filter, R = 1 MΩ and C = 200 pF.

If a dc level is required instead of an ac signal output, this can be generated by using a low-pass filter. An external capacitance is necessary to hold the value. A 330 kΩ resistor and a 0.047 μF capacitance gives a cutoff frequency of 10 Hz. The switching frequency used is 5.5 kHz.

SOFTWARE

The range of the PWMCH0 register is [0;PWMDAT/2] and 0 corresponds to 50%. Table I shows the correspondence between register contents and duty cycle.

Table I. MMR Contents and Duty Cycle Value

High Side		Low Side	
PWMA	**Duty Cycle**	**PWMA**	**Duty Cycle**
0	50%	0	50%
PWMDAT0/2	100%	PWMDAT/2	0%

DC VOLTAGE

With the filter on the output, a small attenuation is given. For a 50% duty cycle, 1.25 V is measured. So to output 500 mV, a 20% duty cycle needs to be programmed. The formula for duty cycle less than 50% is

$$PWMCH0 = PWMDAT0\ (1/2 - d_{OH})$$

To obtain 500 mV output, PWMCH0 = 0x999 with PWMDAT0 = 0x2000.

SINE WAVE

The sine wave values are stored in an array. The first half of the sine wave needs to be outputted on the high side (50% to 100%), and the second part on the low side.

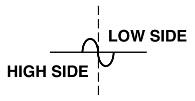

Figure 4. Sine Wave

In terms of algorithm this is translated as

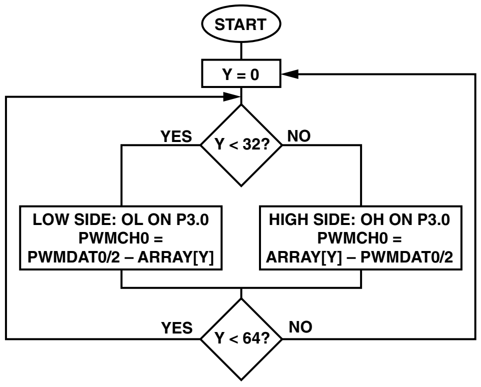

Figure 5. Flow Chart

See Appendix B for relevant code.

PWM ON THE ADuC7020/ADuC7021/ADuC7022

The PWM is not available on the ADuC7020/ADuC7021/ ADuC7022 because of the restricted number of GPIO. However, one channel can be output at a time by using the PLA. See Appendix C for the relevant code.

APPENDIX A: SOFTWARE FOR DC VALUE

```
  GP3CON = 0x10000011;            // Enable the PWM outputs to P3.0 and P3.1
                                  // Setup the PWM
  PWMCON = 0x0001;                // 0x01 is enabled
  PWMDAT0 = 0x2000;               // Period register
  PWMDAT1 = 0x00;                 // Dead time
  PWMDAT2 = 0xFF;                 // PWM pulse width
  PWMCFG = 0x00;                  // Chop
  PWMEN = 0x12F;                  // Enable low side on P3.0
  PWMA = 0x999;                   // Duty cycle of 20%
```

APPENDIX B: SOFTWARE FOR SINE WAVE GENERATION

```
volatile int y = 0 ;

const static unsigned short SinArray[64] = {
  0x07FF, 0x08C8, 0x098E, 0x0A51, 0x0B0F, 0x0BC4, 0x0C71, 0x0D12,
  0x0DA7, 0x0E2E, 0x0EA5, 0x0F0D, 0x0F63, 0x0FA6, 0x0FD7, 0x0FF5,
  0x0FFF, 0x0FF5, 0x0FD7, 0x0FA6, 0x0F63, 0x0F0D, 0x0EA5, 0x0E2E,
  0x0DA7, 0x0D12, 0x0C71, 0x0BC4, 0x0B0F, 0x0A51, 0x098E, 0x08C8,
  0x07FF, 0x0736, 0x0670, 0x05AD, 0x04EF, 0x043A, 0x038D, 0x02EC,
  0x0257, 0x01D0, 0x0159, 0x00F1, 0x009B, 0x0058, 0x0027, 0x0009,
  0x0000, 0x0009, 0x0027, 0x0058, 0x009B, 0x00F1, 0x0159, 0x01D0,
  0x0257, 0x02EC, 0x038D, 0x043A, 0x04EF, 0x05AD, 0x0670, 0x0736
  };

void initPWM(void){
  GP3CON = 0x10000011;                  // Enable the PWM outputs to the GPIO
  PWMCON = 0x0001;                      // 0x01 is enabled
  PWMDAT0 = 0x1000;                     // Period register
  PWMDAT1 = 0x00;                       // Dead time
  PWMDAT2 = 0xFF;                       // PWM pulse width
  PWMCFG = 0x00;                        // Chop
  PWMCH0 = 0x0000;                      // Channel 0
/* workaround for PWM_SYNC errata */
  PLAELM15 = 0x0035;                    // Configure individual elements
  PLAIRQ = 0x001F;                      // IRQ output configuration
  }

void Sinus_IRQ(){                       // Interrupt routine
  if((IRQSIG & PLA_IRQ0_BIT)!=0){       // Interrupt PWMSYNCH Signal workaround
    if (y<32) {                         // high side
        PWMEN = 0x02F;
        PWMDAT0 = 0x1000;
        PWMA = SinArray[y] - 0x800;
    }
    else {                              // low side
        PWMEN = 0x12F;
        PWMDAT0 = 0x1000;
        PWMA = 0x800 - SinArray[y];
    }
    y++;
    if(y==64) y=0;
  }
  return ;
}
```

```
int main(void) {
  initPWM();
  IRQ = Sinus_IRQ;                       // Specify Interrupt Service Routine
  IRQEN = PLA_IRQ0_BIT ;                 // Enable PWMSYNCH IRQ
    while (1){}                          // Wait for PWMSYNC interrupt
   }
```

APPENDIX C: PWM ON THE ADuC7020/ADuC7021/ADuC7022

```
// configuration of PLA, PWM and GPIO to output 16.384 kHz on P1.7

PWMCON = 0x1;                            // enables o/p of the pwm
GP3CON = 0x000000001;
PWMDAT0 = 0x055F;                        // PWM switching frequency of 16.384 kHz
PWMDAT1 = 0x0;                           // dead time is zero

// Configure Port Pins
GP1CON = 0x30000000;                     // If you want to drive the pwm onto
GP4CON = 0x30000000;                     // p1.7 you need at least element 15 as
                                         // it is the one feedback to Block0 Elt0

// PWM0 onto SPM7 via PLAO[0]
PLAELM0 = 0x0059;                        // PWM from element 15
PLAELM8 = 0x0035;                        // PWM input
PLAELM15 = 0x0059;                       // PWM from element
```

AN05567–0–6/05(0)