NetBurner

Mod5213 Pulse Width Modulation Module

Application Note

Revision 1.0
January 30, 2006
Document Status: Initial Release

Table of Contents

Introduction

The pulse width modulation (PWM) module for the Mod5213 generates a synchronous series of pulses having programmable periods and duty cycles. It contains eight independent channels (PWM[7:0]), and each channel can be configured by multiple registers available in the PWM module. With a suitable low-pass filter, the PWM can be used as a digital-to-analog converter.

This application note will provide a brief explanation of each register that is contained within the PWM module, and show how to configure and use the correct pins on the Mod5213 for PWM functionality. A program example that demonstrates pulse width modulation is also provided at the end of this document. For additional detailed information on the MCF5213 PWM module and its registers, please refer to Chapter 24 of the MCF5213 Reference Manual (Revision 1.2).

PWM Module Registers

The following 8-bit registers below allow for the configuration of the eight channels available in the PWM module. Each channel has its own set of counter, period, and duty registers. There can be eight channels with 8-bit channel registers or four channels with 16-bit channel registers for higher resolutions, depending on how the PWM Control Register is set.

PWM Enable Register (PWME)

Each PWM channel has an enable bit in this register to start its waveform output. While in run mode, if all eight PWM output channels are disabled, the prescaler counter shuts off for power savings. If a bit is set/enabled, the associated PWM channel signal becomes available when its corresponding clock source begins its next cycle. See Section 24.2.1 of the MCF5213 Reference Manual for more information.

For PWME[7:0] – ‘0’ disables PWM output, and ‘1’ enables PWM output.

PWM Polarity Register (PWMPOL)

The starting polarity of each PWM channel waveform is determined by the associated bit. If the polarity is changed while a PWM signal is being generated, then a truncated or stretched pulse can occur during the transition. See Section 24.2.2 of the MCF5213 Reference Manual for more information.

For PWMPOL[7:0] – '0' sets a PWM channel's output low at the beginning of the period, then goes high when the duty count is reached. '1' sets a PWM channel's output high at the beginning of the period, then goes low when the duty count is reached.

PWM Clock Select Register (PWMCLK)

Each PWM channel has the capability of selecting one of two clocks. For channels 0, 1, 4, and 5, the clock choices are clock A or SA (Scale A). For channels 2, 3, 6, and 7, the choices are clock B or SB (Scale B). The clock selection is done by setting bits for the associated channels. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

For PWMCLK[7:0] – '0' sets a PWM channel to use a A/B clock source, and '1' sets a PWM channel to use a SA/SB clock source. See Section 24.2.3 of the MCF5213 Reference Manual for more information.

PWM Prescale Clock Select Register (PWMPRCLK)

This register selects the prescale clock source for clocks A and B independently. If the clock prescale is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition. See Section 24.2.4 of the MCF5213 Reference Manual for more information.

For PWMPRCLK[2:0] – These three bits control the rate of clock A which can be used for PWM channels 0, 1, 4, and 5. '000' divides the internal bus clock by 2^0, '001' divides it by 2^1, and so forth until '111' which divides it by 2^7.

For PWMPRCLK[6:4] – These three bits control the rate of clock B which can be used for PWM channels 2, 3, 6, and 7. It is set in the same way as that for clock A above.

PWM Center Align Enable Register (PWMCAE)

The PWMCAE register contains eight control bits for the selection of center-aligned outputs or left-aligned outputs for each PWM channel. Write thse bits only when the corresponding channel is disabled. See Section 24.2.5 of the MCF5213 Reference Manual for more information.

For PWMCAE[7:0] – ‘0’ configures the associated channel to operate in left-aligned output mode, and ‘1’ configures it to operate in center-aligned output mode.

PWM Control Register (PWMCTL)

The PWMCTL register provides various control of the PWM module. Change the CON*n(n+1)* bits only when corresponding channels are disabled. When two channels are concatenated, the even-numbered channel becomes the high order byte and the associated odd-numbered channel becomes the lower order byte. The odd-numbered channel is the output for the 16-bit PWM signal, and the associated even-numbered channel is disabled. Configuration for the 16-bit channel are done through the odd-numbered channel. See Section 24.2.6 of the MCF5213 Reference Manual for more information.

For PWMCTL[7] – CON67; ‘0’ sets channels 6 and 7 as separate 8-bit PWMs. ‘1’ concatenates channels 6 and 7 to form one 16-bit PWM channel.

For PWMCTL[6] – CON45; ‘0’ sets channels 4 and 5 as separate 8-bit PWMs. ‘1’ concatenates channels 4 and 5 to form one 16-bit PWM channel.

For PWMCTL[5] – CON23; ‘0’ sets channels 2 and 3 as separate 8-bit PWMs. ‘1’ concatenates channels 2 and 3 to form one 16-bit PWM channel.

For PWMCTL[4] – CON01; ‘0’ sets channels 0 and 1 as separate 8-bit PWMs. ‘1’ concatenates channels 0 and 1 to form one 16-bit PWM channel.

For PWMCTL[3] – ‘0’ allows the clock to the prescaler while in doze mode. ‘1’ stops the input clock to the prescaler whenever the core is in doze mode.

For PWMCTL[2] – ‘0’ allows the PWM counters to continue while in debug mode. ‘1’ disables the PWM input clock to the prescaler when the core is in debug mode. Useful for emulation as it allows the PWM function to be suspended.

PWM Scale A Register (PWMSCLA)

PWMSCLA is the programmable scale value used in scaling clock A to generate clock SA. Clock SA is generated with the following equation:

```
Clock SA = [ Clock A / ( 2 x PWMSCLA ) ]
```

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLA). See Section 24.2.7 of the MCF5213 Reference Manual for more information.

For PWMSCLA[7:0] – All seven bits are used in inputting part of the divisor value to form clock SA from clock A, as indicated by the formula above for clock SA. “0x00” represents the value 256, “0x01” is 1, “0x02” is 2, and so forth until “0xFF”, which is 255.

PWM Scale B Register (PWMSCLB)

PWMSCLB is the programmable scale value used in scaling clock B to generate clock SB. Clock SB is generated with the following equation:

```
Clock SB = [ Clock B / ( 2 x PWMSCLB ) ]
```

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLB). See Section 24.2.8 of the MCF5213 Reference Manual for more information.

For PWMSCLB[7:0] – All seven bits are used in inputting part of the divisor value to form clock SB from clock B, as indicated by the formula above for clock SB. “0x00” represents the value 256, “0x01” is 1, “0x02” is 2, and so forth until “0xFF”, which is 255.

PWM Channel Counter Registers (PWMCNT)

Each channel has a dedicated 8-bit up/down counter that runs at the rate of a selected clock source. The user can read the counters at any time without affecting the count or the operation of the PWM channel. Any value written to the counter causes the counter to reset to 0x00, the counter direction to be set to up for center-aligned mode, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit.

The counter is also cleared at the end of the effective period. When the channel is disabled, the associated PWMCNT register does not count. When a channel is enabled, the associated PWM counter starts at the count in the PWMCNT register. See Section 24.2.9 of the MCF5213 Reference Manual for more information.

For PWMCNT[7:0] – Current value of the PWM up counter. Resets to zero when written. Each channel has its own PWMCNT register.

PWM Channel Period Registers (PWMPER)

The PWM period registers determine the period of the associated PWM channel. Calculating the output period depends on the output mode (center-aligned has twice the period as left-aligned mode) as well as PWMPER:

```
PWM period = Channel clock period x ( PWMCAE[CAEn] + 1 ) x PWMPERn
```

See Section 24.2.10 of the MCF5213 Reference Manual for more information.

For PWMPER[7:0] – The value written to this register represents the PWM period of the associated channel. When the counter reaches this value, it resets to 0x00 (left-aligned mode), or counts down (center-aligned mode).

PWM Channel Duty Registers (PWMDTY)

The PWM duty registers determine the duty cycle of the associated PWM channel. To calculate the output duty cycle (high time as a percentage of period) for a particular channel:

```
Duty Cycle = | ( 1 – PWMPOL[PPOLn] – ( PWMDTYn / PWMPERn ) | x 100%
```

See Section 24.2.11 of the MCF5213 Reference Manual for more information.

For PWMDTY[7:0] – The value written to this register represents the PWM duty cycle. When the counter reaches this value, the signal goes high (if polarity bit was ‘0’) or goes low (if polarity bit was ‘1’).

PWM Shutdown Register (PWMSDN)

The PWM shutdown register provides emergency shutdown functionality of the PWM module. The PWMSDN[7:1] bits are ignored if PWMSDN[SDNEN] is cleared. See Section 24.2.12 of the MCF5213 Reference Manual for more information.

Pin Assignment Configuration

The GPIO module must be configured to enable the peripheral function of the appropriate pins prior to configuring the PWM module. On the Mod5213, the PWM pins are 21 through 28 with the following assignments:

- Pin 21: PWM Channel 6
- Pin 22: PWM Channel 4
- Pin 23: PWM Channel 2
- Pin 24: PWM Channel 0
- Pin 25: PWM Channel 7
- Pin 26: PWM Channel 5
- Pin 27: PWM Channel 3
- Pin 28: PWM Channel 1

Configuring the pins can be done by directly writing to the registers, or it can be done via the Pin class. For example, if one would like to configure pin 24 for PWM Channel 0 functionality by writing directly to the registers, then it would be written like this:

```
sim.gpio.ptcpar &= 0xFC;
sim.gpio.ptcpar |= 0x03;
```

However, you would need to know what register to write to, as well as know what bits to set in order to correctly configure for PWM functionality. The Pin class exists to avoid this hassle. The following example shows the equivalent of configuring pin 24 through the usage of the Pin class:

```
Pins[24].function( PIN24_PWM0 );
```

The information that needs to be known is the pin to be configured (pin 24) and the type of functionality required for that pin (`PIN24_PWM0`). The list of functionalities available for each pin can be found in the pinconstant.h file of the \Nburn\MOD5213\include directory.

Program Example

```
/**********************************************************************/
/*   This example program exercises the pulse width modulation        */
/*   module in the MCF5213 CPU.  When running this program on the     */
/*   MOD5213, there should be an output frequency of about 829.4 kHz */
/*   on pin 24 (use of an oscilloscope to measure this is             */
/*   recommended).  To achieve this value through calculation,        */
/*   consider the following case:                                     */
/*                                                                    */
/*   The CPU frequency of the MCF5213 is 66.3552 Mhz. The internal    */
/*   bus clock is determined by dividing this value by 2. Therefore, */
/*   the internal bus clock is 33.1776 Mhz. Channel 0 is programmed  */
/*   to use clock SA, so the internal bus clock passes through a      */
/*   divisor prescaler of clock A, and then through an additional    */
/*   divisor prescaler of clock SA, which is then again further      */
/*   divided by 2. After dividing the resulting frequency by the      */
/*   value present in the channel period register (5), you get the   */
/*   output frequency on pin 24.                                      */
/*                                                                    */
/*   The duty cycle value in the channel duty register is 3. When     */
/*   the PWM counter matches the duty register, the output flip-flop */
/*   changes state causing the PWM waveform to also change state.    */
/*   The duty cycle with the current configuration is:               */
/*                                                                    */
/*   Duty Cycle = (1 - PWMPOL[PPOLn] - (PWMDTYn/PWMPERn)) x 100%      */
/*              = (1 -       0       - (   3   /   5   )) x 100%      */
/*              = 40%                                                 */
/**********************************************************************/

#include "predef.h"
#include <stdio.h>
#include <ctype.h>
#include <basictypes.h>
#include <serialirq.h>
#include <system.h>
#include <constants.h>
#include <ucos.h>
#include <serialupdate.h>
#include <pins.h>
#include <..\MOD5213\system\sim5213.h>

//
// Instruct the C++ compiler not to mangle the function name
//
extern "C" {
   void UserMain( void *pd );
}

//
// Name for development tools to identify this application
//
const char *AppName = "Mod5213PWMDemo";

//////////////////////////////////////////////////////////////////////
// UserMain - Main task.
```

```
//
void UserMain( void *pd ) {
   OSChangePrio( MAIN_PRIO );
   EnableSerialUpdate();

   SimpleUart( 0, SystemBaud );
   assign_stdio( 0 );

   //////////////////////////////////////////////////////////////////////
   // Configure pin 24 for PWM Channel 0 functionality

   // Set for PWM functionality
   Pins[24].function( PIN24_PWM0 );

   // Disable all PWM channel output before making any settings
   sim.pwm.pwme = 0;

   // Set to have an initial low signal, then set high on duty output
   // compare
   sim.pwm.pwmpol &= ~0x01;

   // Set to use clock SA (Scale A)
   sim.pwm.pwmclk |= 0x01;

   // Set to use a clock A prescale value of 2 (Internal Bus Clock /
   // 2^1)
   sim.pwm.pwmprclk |= 0x01;

   // Set to operate channel 0 in left-aligned output mode
   sim.pwm.pwmcae &= ~0x01;

   // All channels are independent 8-bit channels; doze and debug mode
   // disabled
   sim.pwm.pwmctl = 0;

   // Use scale divisor value of 2 to generate clock SA from clock A
   sim.pwm.pwmscla = 0x02;

   // Write any value to this register to reset the counter and start
   // off clean
   sim.pwm.pwmcnt[0] = 1;

   // Set PWM Channel 0 period register to a value of 5
   sim.pwm.pwmper[0] = 5;

   // Set PWM Channel 0 duty register to a value of 3
   sim.pwm.pwmdty[0] = 3;

   // Enable PWM output for PWM Channel 0
   sim.pwm.pwme |= 0x01;

   iprintf( "Application started\r\n" );

   while ( 1 ) {
      OSTimeDly( TICKS_PER_SECOND );
   }
}
```