Using the Pulse Width Modulation with the MCF521x ColdFire® Microcontroller

by: Paolo Alcantára and Lech Olmedo
RTAC Americas

1 Introduction

This document is a quick reference to configure the pulse width modulation (PWM) module for the MCF521x microcontroller (MCU). Basic knowledge about the functional description and configuration options gives the user a better understanding on how the PWM module works. Some examples are included which are intended to be modified to suit the specific needs for any application.

2 MCF521x Pulse Width Modulation Features

The PWM module generates a synchronous series of pulses having a programmable period and duty cycle. With a suitable low-pass filter, the PWM can be used as a digital-to analog converter.
2.1 Description

The main features of PWM module are:

- Double-buffered period and duty cycle
- Left-aligned or center-aligned outputs
- Eight independent PWM channels using 8-bit period register or four PWM channels using 16-bit period register.
- Byte-wide registers provide programmable duty cycle and period control
- Four programmable clock sources

3 Pulse Width Modulation Explanation

The example code shows how the PWM works employing 8 different PWM signals through the M5213EVB board. Half of these signals are displayed on 4 LEDs and the others can be corroborated using an oscilloscope. This demonstrates how to configure the PWM module in an easy way and allows you to include it very quickly in your projects.

A PWM signal has a frequency and a duty cycle. In most applications, the frequency is fixed while the duty cycle varies. Figure 1 illustrates a PWM signal.

![Figure 1. PWM Signal](image)

A general purpose timer (GPT) using output compare mode can also implement a PWM signal. However, the PWM module is more precise with frequency and duty cycle.

A dedicated period and duty register exist for each channel and are double buffered. So if a change occurs in these registers, the change does not take effect until one of the following events occur:

- The effective period ends
- The counter resets to zero
- The channel is disabled

Additionally, the PWM module can generate four types of signals:

- Center-aligned with clock
• Center-aligned with scaled clock
• Left-aligned with clock
• Left-aligned with scaled clock

3.1 Clock Source

Each PWM channel can be associated with a clock generator: simple clock A, simple clock B, scaled clock A and scaled clock B. This distinction allows more flexibility in PWM module between channels. Table 1 shows clock source options per channels.

<table>
<thead>
<tr>
<th>Source Clock</th>
<th>PWM channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock A or Scaled Clock A</td>
<td>0, 1, 4, 5</td>
</tr>
<tr>
<td>Clock B or Scaled Clock B</td>
<td>2, 3, 6, 7</td>
</tr>
</tbody>
</table>

3.2 Scaled Clock

The PWM channel with a scaled clock first receives the simple clock as its clock source. Then, the generated reference is prescaled with another register. A block diagram showing the difference between the simple clocks and scaled clocks is in Figure 2.

3.3 Center-Aligned

This feature can be applied to any channel. A center-aligned signal is handled by a counter and its ability to go up and down. Signal can start each period at high or low digital value. Each period can be divided in four states explained in Figure 3.
The steps 1–4 in Figure 3 are explained below:

1. The counter operates as an up/down counter in center-aligned mode and is set to up-counter at the beginning.
2. When the PWM counter matches the duty register, the output flip-flop changes state causing the PWM also to change state from low to high (or high to low depending on the PPOL bit).
3. A match between the PWM counter and period register changes the counter direction from an up-count to a down-count.
4. When the counter decrements and matches the duty register again, the PWM output channel changes state.
5. When the PWM counter reaches 0x00, the counter changes from a down-count to an up-count and loads the double-buffered period and duty cycle values.

3.4 Left-Aligned

The feature is fully independent of the other channels. A left-aligned signal is handled by a counter and its capability to only count up depending on some registers. The signal can start each period at high or low digital value. Each period can be divided in two states as shown in Figure 4.
The 8-bit counter is configured as an up-counter only.

1. A match between PWM counter and period register resets the counter and a change in output PWM state. A load is performed from the double buffer period and duty register.
2. When PWM counter matches the duty register, the output PWM changes state from low to high (or low to high depending on the PPOL bit).

### 4 Case Studies

#### 4.1 Example: Case study for the PWM Module

This example uses eight PWM channels in 8-bit mode. All channels share the same clock source (simple clock or scaled clock).

There are two ways to calculate frequency for all channels:

- Using equations from Table 2.
- Using the spreadsheet showed in Figure 6 and Figure 5.

With this approach, period (PWMPer) and duty cycle (PWMdty) are the only differences between each PWM channel (without considering polarity and channel alignment).

After configuring each PWM channel, the last step is to enable the desired channels. After this, the PWM are fully independent and do not need any more configuration. Although, the PWM frequency and duty cycle can be changed on the go and are reflected on the next PWM period.

#### 4.2 Example: Configuring PWM Module

Table 2 shows all equation to be applied for each PWM signal type.
Table 2. PWM Equations

<table>
<thead>
<tr>
<th>Equations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ f_n = \frac{f_{SYS/2}}{2 \cdot PCK[A,B] \cdot PWMSCLA \cdot PWMPER_n} ]</td>
<td>Left-aligned with scaled clock [A,B]</td>
</tr>
<tr>
<td>[ f_n = \frac{f_{SYS/2}}{4 \cdot PCK[A,B] \cdot PWMSCLA \cdot PWMPER_n} ]</td>
<td>Center-aligned with scaled clock [A,B]</td>
</tr>
<tr>
<td>[ f_n = \frac{f_{SYS/2}}{PCK[A,B] \cdot PWMPER_n} ]</td>
<td>Left-aligned with clock [A,B]</td>
</tr>
<tr>
<td>[ f_n = \frac{f_{SYS/2}}{2 \cdot PCK[A,B] \cdot PWMPER_n} ]</td>
<td>Center-aligned with clock [A,B]</td>
</tr>
<tr>
<td>[ duty = \left\lfloor 1 - \frac{PWMPOL[PPOL_n]}{PWMPER_n} \right\rfloor \times 100% ]</td>
<td>Duty cycle for all PWM type signals.</td>
</tr>
</tbody>
</table>

Additional to this equation, a spreadsheet (PWMparameterCalculator.xls), showed in Figure 6 and Figure 5, contains an easy way to generate a desired frequency using these parameters:

- Desired frequency to be output through pin.
- \( f_{SYS/2} \) — Core internal frequency divided by two.
- PERIOD—PWMPER\(_n\) register. Recommended value is 200 to achieve good duty cycle resolution. (Refer to the duty cycle equation in Table 2.)

The spreadsheet calculates the four possible PWM signal types and the obtained frequency with these parameters. Not all frequencies can be synthesized in all types of signals, so frequencies usually have to use a scaled clock instead of a simple clock. It is your responsibility to choose the results (as SCLA) that are below 256 because the register 8-bits wide. For example, in Figure 5 the following are used:

- \( f_{SYS/2} = 40,000,000 \) Hz
- Desired frequency = 30 Hz
- Period = 200 (to have a more exact duty cycle)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Unit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{SYS/2} )</td>
<td>400000000</td>
<td>Hz</td>
<td>Constant</td>
</tr>
<tr>
<td>desired frequency</td>
<td>30</td>
<td>Hz</td>
<td>Variable</td>
</tr>
<tr>
<td>PERIOD</td>
<td>200</td>
<td>N/A</td>
<td>Constant</td>
</tr>
</tbody>
</table>
After entering 40 MHz as the clock source, 30 Hz as the desired PWM, and 200 as PERIOD into the spreadsheet (PWMparameterCalculator.xls), it will show the different PWM frequencies with the provided information.

You now should select which PCKA and SCLA values can be used according to the following rules:

- **PCKA** allowed values are: 1, 2, 4, 8, 16, 32, 64 and 128. These values are fixed on the spreadsheet.
- **SCLA** must be less than 256. Therefore, in this example you can only choose between 156, 78, 39 or 20 for left-aligned and 156, 78, 39, 20 or 10 for center-aligned.
- Within these SCLA values, you should select the obtained frequency row closest to the desired frequency.

The final parameters that could be used with the same results are shown in **Table 3** and **Table 4**.

### 4.3 Left-Aligned with Scaled Clock

**Table 3. Left-Aligned with Scaled Clock**

<table>
<thead>
<tr>
<th>Option</th>
<th>PERIOD</th>
<th>PCKA</th>
<th>SCLA</th>
<th>Desired freq</th>
<th>Obtained Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>4</td>
<td>156</td>
<td>30 Hz</td>
<td>30.04807 Hz</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>5</td>
<td>78</td>
<td>30 Hz</td>
<td>30.04807 Hz</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>6</td>
<td>39</td>
<td>30 Hz</td>
<td>30.04807 Hz</td>
</tr>
</tbody>
</table>

---

1. Using PCKA and SCLA, PWM channels affected are 0, 1, 4, 5, according to **Table 1**. It is your responsibility to use the appropriate PWMPERn (PWMPER1 for channel 1, for example).

2. PCKA values that appear on the spreadsheet are modified by $2n$. Therefore, PCKA only accepts $n$ values.

3. If the obtained frequency is not close enough to the desired frequency, you must change PERIOD by trial and error until the difference between obtained and desired frequency is adequate.
4.4 Center-Aligned with Scaled Clock

Table 4. Center-Aligned with Scaled Clock

<table>
<thead>
<tr>
<th>Option 1</th>
<th>PERIOD(^1)</th>
<th>PCKA(^2)</th>
<th>SCLA</th>
<th>Desired Freq</th>
<th>Obtained Freq(^3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>200</td>
<td>3</td>
<td>156</td>
<td>30 Hz</td>
<td>30.04807 Hz</td>
</tr>
<tr>
<td>Option 2</td>
<td>200</td>
<td>4</td>
<td>78</td>
<td>30 Hz</td>
<td>30.04807 Hz</td>
</tr>
<tr>
<td>Option 3</td>
<td>200</td>
<td>5</td>
<td>39</td>
<td>30 Hz</td>
<td>30.04807 Hz</td>
</tr>
</tbody>
</table>

\(^1\) Using PCKA and SCLA, PWM channels affected are 0, 1, 4, 5, according to Table 1. It is your responsibility to use the appropriate PWMPER\(n\) (PWMPER1 for channel 1, for example).

\(^2\) PCKA values that appear on the spreadsheet are modified by 2\(n\). Therefore, PCKA only accepts \(n\) values.

\(^3\) If the obtained frequency is not close enough to the desired frequency, you must change PERIOD by trial and error until the difference between obtained and desired frequency is adequate.

The above example uses a period of 200. If the desired duty cycle is 50%, PWMDTY\(n\) must be 100 (\(n\) refers to channel number) and PWMPOL[PPOL] must be set, according to PWM equation from Table 2. With a period of 200, you can select a precise duty cycle (0.5% resolution).

5 Register Configuration

This section shows the register configuration that affect the entire PWM channels and the code lines used. This section is meant to merely illustrate how the registers are configured.

5.1 Port Assignment Register

The following code lines configure the PTCPAR (Figure 7) and PTDPAR (Figure 8), to assign the PWM function to the GPIO pins.

```c
/* associate odd PWM channels to a pin */
MCF_GPIO_PTDPAR = 0
| MCF_GPIO_PTDPAR_PTDPAR0 /* PWM channel 1 */
| MCF_GPIO_PTDPAR_PTDPAR1 /* PWM channel 3 */
| MCF_GPIO_PTDPAR_PTDPAR2 /* PWM channel 5 */
| MCF_GPIO_PTDPAR_PTDPAR3; /* PWM channel 7 */

/* associate even PWM channels to a pin (reflected in EVB LEDs) */
MCF_GPIO_PTCPAR = 0
| MCF_GPIO_PTCPAR_TIN0_PWM0/* PWM channel 0 */
| MCF_GPIO_PTCPAR_TIN1_PWM2/* PWM channel 2 */
| MCF_GPIO_PTCPAR_TIN2_PWM4/* PWM channel 4 */
| MCF_GPIO_PTCPAR_TIN3_PWM6; /* PWM channel 6 */
```

Using the Pulse Width Modulation with the MCF521x ColdFire® Microcontroller, Rev. 0
5.2 PWM Clock Select Register

The next code lines configure the PWMCLK.

```c
/* Associate every channel with a source clock */
/* A, or B, SA or SB depending the channel */
MCF_PWM_PWMCLK = u8Clock;
```

5.3 PWM Clock Select Register

The program uses this code to write the values to the PWMPRCLK:

```c
/* change the prescaler of the clock source for every channel */
MCF_PWM_PWMPRCLK = 0
| MCF_PWM_PWMPRCLK_PCKA(BUSCLK_16)
| MCF_PWM_PWMPRCLK_PCKB(BUSCLK_16);
```
5.4 PWM Scale [A,B] Registers

On the next code lines, the scaled clock register is configured according to Figure 11 and Figure 12.

```c
/* set scaled value for scaled clock, if are activated */
MCF_PWM_PWMSCLA = u8ScaledValue;
MCF_PWM_PWMSCLB = u8ScaledValue;
```

![Figure 11. PWM Scaled Register A](image)

![Figure 12. PWM Scaled Register B](image)

6 Configuration Summary

The following steps are needed to configure the pulse width modulator module:

1. Configure PWM pins as their primary function in the pin assignment registers.
2. Associate each PWM channel with a clock source in the PWMCLK register.
3. Configure PWMPRCLK. If a scaled clock is selected, the PWMSCLA or PWMSCLB registers must be configured as well.
4. For every PWM channel, these registers must be filled:
   - PWMPER: set PWM period (see period equation above).
   - PWMDTY: set duty cycle (see duty cycle equation above).
   - PWMPOL: set channel polarity bit (start each period at high or low).
   - PWMCAE: set channel alignment bit (left or center alignment).
   - PWME: enable each channel using respective bit.

7 Configuration Notes

The following details must be considered when configuring the PWM:

- PWMPRCLK[PCKA] and PWMPRCLK[PCKB] affect all PWM channels.
- PWMSCLA and PWMSCLB affect only PWM channels linked to scaled clocks A or B.
8 Conclusion

The PWM module is a versatile module with eight channels in 8-bit mode or four channels in 16-bit mode. With a combination of clock source and prescaler, you can calculate precise PWM frequencies and duty cycles. After configuration, the module is fully independent and if a change in frequency or duty cycle is needed, a simple write in configuration registers is reflected at the next PWM period.

8.1 Considerations and References

Find the latest software updates and configuration files for the MCF521x devices on the Freescale Semiconductor website: http://www.freescale.com

This application note applies to:

- MCF5211, MCF5212, and MCF5213 devices
- M5213EVB Development Board employed with the PWM software demo

For more information on the pulse width modulation module refer to the MCF5213 ColdFire Integrated Microcontroller Reference Manual.

The PWMSoftwareDemo software was developed in CodeWarrior for Coldfire v6.3.

Download the source files for PWMSoftwareDemo software (PWMSoftwareDemo.zip) from http://www.freescale.com. Inside you’ll find source code and a spreadsheet to easily calculate different PWM parameters.