Experiment 6: Pulse-Width Modulation

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
142	15 March 2015		H

Contents

1 Objectives

- Understanding and using *pulse-width modulation (PWM)*.

2 Parts List

- LPC1769 LPCXpresso board
- USB A-Type to Mini-B cable
- Breadboard
- RGB-LED or buzzer
- Seven-segment display
- Jumper wires
- Servo motor (optional)

3 Background

3.1 Pulse-Width Modulation (PWM)

A pulse-width modulated (PWM) signal is a periodic square wave signal. The difference between a PWM signal and a clock signals is the flexibility of its *duty cycle*.

A periodic square wave is high for some part of its period, and low for the rest of the period. Its *duty cycle* is the percentage of the period for which the signal is high. Usually, a clock wave has a duty cycle of 50%. In a PWM signal, the duty cycle is controllable. The name is derived from the idea that the *width* of the high *pulse* is *modulated* according to some value.

3.2 PWM Applications

PWM has many useful applications in embedded systems. The main two categories are:

1. When a microcontroller does not have a DAC circuit, PWM can be used to *modulate* different analog values.
2. Some devices are built to be used with PWM. The most famous example is servo motors.

   Servo motors usually require a 50-Hz square wave (period of 20 ms). The duration of the high pulse determines the motor's angle. Usually, the full swing of the servo corresponds to a high interval of 1 to 2 ms, whereas a high interval of 1.5 ms corresponds to the neutral servo position.

3.3 Generating PWM with LPC1769

The LPC1769 features a pulse-width modulator peripheral. The generic steps discussed in Experiment 5 for setting up a peripheral device apply here:

1. Power: the PWM circuit is powered on by default.
2. Peripheral Clock (PCLK): recall that the default division factor is 4.
3. Pin functions: PWM pins must be configured for PWM use. Otherwise, the PWM circuit will act as a standard timer (or counter).

Tip

1. There is only one PWM circuit, called `PWM1`. That does not imply that there is a `PWM0` or `PWM2`.
2. There are six PWM channels, referred to as `PWM1.1` to `PWM1.6`.
3. You have the option of more than one pin to pin out any of the channels.

Important
You need to disable the pull-up resistor to avoid affecting the PWM voltage. Recall that you can do that using the `LPC_PINCON->PINMODEx` register.

Exercise

Refer to chapter 8 of the LPC176x manual to determine:

1. Which pins are you going to use for PWM?
2. Which `PINSELx` register should you use?
3. Which `PINSELx` bits should you set?
4. To what value should you set those `PINSELx` bits?
5. How to disable the pull-up resistor?

3.3.1 PWM Configuration

To fully specify a PWM signal, you need to specify:

1. Its period (or, equivalently, its frequency)
2. Its duty cycle

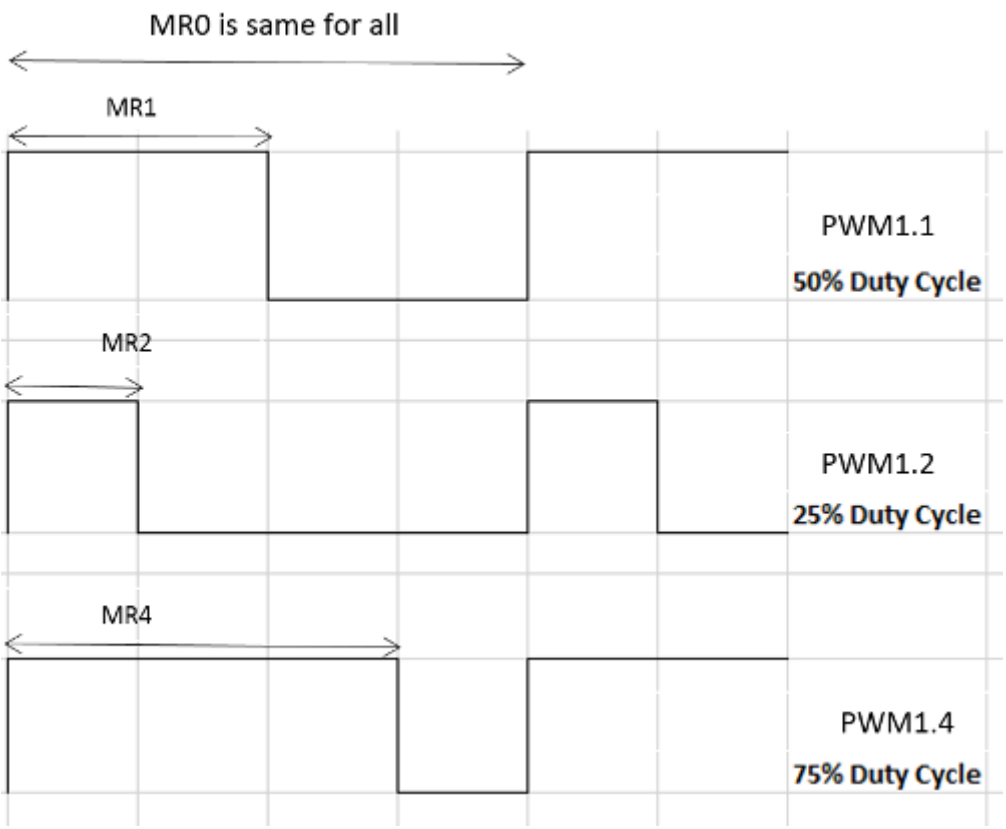
In the simplest configuration, the `MR0` register (aka `PWM1MR0`) determines the period, while `MR1` to `MR6` determine the duty cycle, as illustrated in the following example.

Example 3.1 Period and Duty Cycles

If `MR0` is set to 80, then:

Register	Value	Duty Cycle	PWM Channel
`MR1`	40	50%	1 (`PWM1.1`)
`MR2`	20	25%	2 (`PWM1.2`)
`MR4`	20	75%	4 (`PWM1.4`)
`MR5`	72	90%	5 (`PWM1.5`)

The figure below shows the different PWM outputs for the same `MR0`.



Single Edge Controlled PWM
In the example above, the periodic signal on all channels will go high at the beginning of the period, and each channel will be reset when matching the number in the corresponding `MR1` to `MR6` register.
This PWM configuration is called *single edge controlled PWM*.

To have different PWM channels be set and reset at different times, some PWM channels can be configured as *double edge controlled PWM* signals.

Double Edge Controlled PWM
In double edge controlled, you can control when to set or reset the pulse within the period, and whether to set or reset first.
The `MR0` register still controls the duration of the full period.

Example 3.2 Double Edge Controlled PWM
PWM channel 2 (`PWM1.2`) is set by `MR1` and reset by `MR2`.
So, setting `MR0` = 100, `MR1` = 50, and `MR2` = 75 will result in a signal that is low at the beginning of the period, becomes high in the middle of the period, and goes back to low in the middle of the second half of the period.
In contrast, setting `MR0` = 100, `MR1` = 75, and `MR2` = 50 will result a signal that is high at the beginning of the period, becomes low in the middle of the period, and goes back to high in the middle of the second half of the period.

Note
PWM channels can be configured to be *single edge controlled* or *double edge controlled* using the `PWMSELn` bits of the *PWM Control Register* (`PWM1PCR` or `LPC_PWM1->PCR`).
For details, see Table 444 and Table 452 in the LPC176x manual.

3.3.2 The Prescale Register

In most PWM applications, the timing accuracy is crucial. When `MR0` is set to 100, every time the *PWM Timer Counter* register (`PWM1TC`, or `TC` for short) matches 100, a new period starts. The question is how long is this 100?

`TC` is a 32-bit register that is incremented every `PR` + 1 cycles of `PCLK`, where `PR` is the *Prescale Register* (`PWM1PR` or `LPC_PWM1->PR`).

Therefore, `TC` frequency is given by:

$$\texttt{TC} \text{ frequency in Hz} = \frac{\text{System clock}}{\text{PCLK divisor} \times (\text{PR} + 1)}$$

where *PCLK divisor* is 1, 2, 4, or 8, depending on the setting of the `PCLKSELx` register (default is 4).

For *system clock*, you can use the `SystemCoreClock` variable, which is set by CMSIS to the CPU clock speed.

Example 3.3 Setting the Prescale Register

To set the prescale register such that `TC` is incremented every 1 μs (frequency of 1,000,000 Hz):

```
LPC_PWM1->PR = SystemCoreClock / (4 * 1000000) - 1;
```

Now you can control the period duration of the PWM signal by setting the `MR0` register.

For example:

```
LPC_PWM1->MR0 = 1000000; // PWM period is 1 second
```

3.3.3 Other settings

- `LPC_PWM1->LER` is used to latch the new `MRx` values. You must use it every time you change any of the `MRx` values.
- `LPC_PWM1->PCR` is used to enable PWM1 with single or double edge operation. If ignored, PWM will act as a counter.
- `LPC_PWM1->TCR` is used to enable, disable, or reset counting in the `TC` register. You should use it at least once to enable counting.
- `LPC_PWM1->MCR` is similar to the timers `MCR` registers. It is used to generate interrupts or reset `TC` when matches occur.

4 Tasks

Use the PWM feature of the LPC1769 microcontroller in an experiment of your choosing.

The output PWM signal can be used to control a servo, an RGB LED (color LED), or a buzzer.

5 Resources

[lpc1769-manual]
NXP Semiconductors. *UM10360 LPC176x/5x User manual*. Rev. 3.1. 2 April 2014.
http://www.nxp.com/documents/user_manual/UM10360.pdf

6 Grading Sheet

Task	Points
Generating PWM output	7
Discussion	3
Bonus: Use a servo	+2