

Experiment 4: Hardware Timers

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
142	22 February 2015		H

Contents

1 Objectives	1
2 Parts List	1
3 Background	1
3.1 Peripheral Clock (PCLK)	1
3.2 How Timers Work	2
3.3 Important Notes	3
4 Tasks	3
5 Resources	3
6 Grading Sheet	3

1 Objectives

- Using hardware timers
- Using the [LPC1769 manual](#) to figure out how to use a given register
- Identifying how to access a given register by referring to the `LPC17xx.h` file

2 Parts List

- LPC1769 LPCXpresso board
- USB A-Type to Mini-B cable
- Breadboard
- LEDs
- 330-Ohm Resistors
- Jumper wires

3 Background

There are four hardware timers in LPC1769. They are identical and can work independently with different settings.

3.1 Peripheral Clock (PCLK)

Timers, among other devices, rely on *peripheral clocks* (PCLK), which in turn are derived from the *core clock* (CCLK).

There are four possible frequency configurations for the peripheral clock (PCLK), which are set using a pair of bits.

Table 1: Peripheral Clock (PCLK) Frequency Configurations

Bit Values	Frequency Configuration
01	$PCLK = CCLK$
10	$PCLK = CCLK / 2$
00	$PCLK = CCLK / 4$
11	$PCLK = CCLK / 8$

These pairs of bits belong to the `PCLKSEL0` and `PCLKSEL1` registers, which control the PCLK frequency for all peripherals.

The [PCLKSEL0 and PCLKSEL1 Register Fields figure](#) illustrates some of the fields of the `PCLKSEL0` and `PCLKSEL1` registers. Every two bits control the PCLK frequency for a specific peripheral.

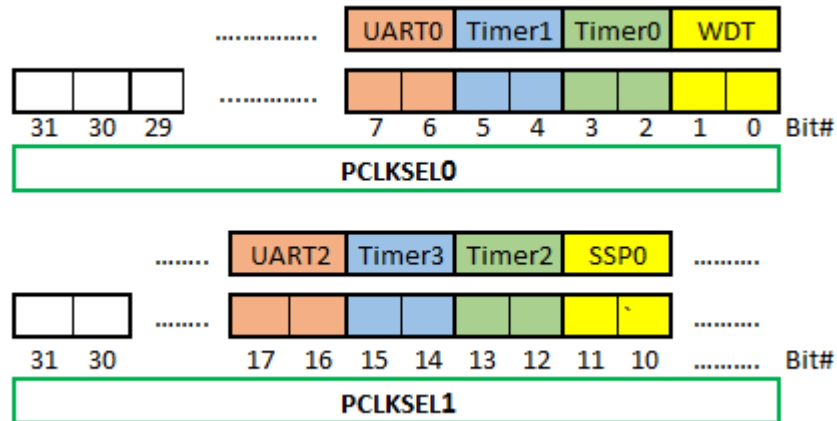


Figure 1: PCLKSEL0 and PCLKSEL1 Register Fields

Question

Can you ignore this step? What would happen if we skip it?

Note

For the full list of peripherals and their corresponding two bits in PCLKSEL0 or PCLKSEL1, you can refer to Chapter 4 (section 4.7.3) in the [LPC1769 manual](#).

Note

This section is not specific to timers. It is about configuring the frequency of PCLK, which is required for timers.

Note

You may want to refer back to this section whenever you want to use a peripheral that requires PCLK.

3.2 How Timers Work

By default, the *Timer Counter* (TC) register is incremented every PCLK cycle. When the value of TC reaches the value in *Match Register 0* (MR0), an action can be taken. Therefore, setting MR0 specifies the timer's period.

The action to be taken every time TC reaches MR0 can be set using the *Match Control Register* (MCR) to one (or more) of the following:

1. Generate an interrupt
2. Reset TC
3. Stop TC

You can enable or disable the above actions by setting or clearing the three least significant bits of the MCR register.

Table 2: Setting Timer Actions Using the MCR Register

MCR bit	Bit value = 1	Bit value = 0
0	Enable timer interrupt	Disable timer interrupt
1	Reset TC	Disable this feature
2	Stop TC	Disable this feature

3.3 Important Notes

- For TC registers to start counting, the least significant bit of the *Timer Control Register* (TCR) must be set. This bit is known as *Counter Enable*.
- If you choose to enable the timer interrupt, remember to enable the the NVIC and to clear the interrupt bit in the ISR. To clear the MR0 interrupt flag, set the least significant bit in the *Interrupt Register* (IR).
- A common misconception is to assume that register MR0 can be used with timer 0 only, register MR1 with timer 1 only, and so on. Any of the MR_x registers can be used with any of the four timers. Moreover, multiple MR_x registers can be used with the same timer.
- As usual, all the registers in this experiment are fields of some structures. Refer to the LPC17xx.h header file to find the required name and field to access the required register.

4 Tasks

Use a hardware timer and timer interrupts to blink an LED.

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
Use hardware timers and timer interrupts	5
Identifying required register settings for alternative configurations	3
Discussion	2