Experiment 5: Interrupts — Part II

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
141	Oct. 18, 2014		H

Contents

1 Objectives

- Non-GPIO external interrupts
- Interrupt priority

2 Parts List

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

3 Background

3.1 Non-GPIO External Interrupts

External interrupts can be generated using any of 4 dedicated external interrupt pins, named `EINT1`, `EINT2`, `EINT3`, and `EINT4`.

EINT0	P2.10
EINT1	P2.11
EINT2	P2.12
EINT3	P2.13

If an external interrupt is requested through one of the above pins, the CPU will jump to the corresponding ISR, which is identified using the C function name.

Name	Pin	Handler (ISR)
`EINT0`	P2.10	`void EINT0_IRQHandler()`
`EINT1`	P2.11	`void EINT1_IRQHandler()`
`EINT2`	P2.12	`void EINT2_IRQHandler()`

Name	Pin	Handler (ISR)
EINT3	P2.13	void EINT3_IRQHandler()

Note
External interrupt no. 3, EINT3, can be activated in two ways:

1. Using P2.13 EINT function, or
2. Using a GPIO pin from port 0 or port 2.

In other words, the EINT3 channel is shared with GPIO interrupts.

To force one of the above pins to act as EINTx, you have to program that pin using the PINSEL register: LPC_PINCON->PINSEL1.

3.1.1 Enabling Non-GPIO External Interrupts

The function of each pin is controlled by a set of function selection registers named PINSEL0, PINSEL1, PINSEL2, ..., PINSEL10. PINSEL0 controls the functions of the lower half of port 0 (P0.0 to P0.15). As you may expect, PINSEL1 controls the functions of pins P0.16 to P0.31, PINSEL2 controls the functions of pins P1.0 to P1.15, and so on.

Because each 32 bits in PINSELx control the functions of 16 pins, you can correctly conclude that every 2 bits in PINSELx control the function of a single pin:

00	Primary (default) function, (GPIO)
01	First alternate function
10	Second alternate function
11	Third alternate function

For example, the two least significant bits in PINSEL0 control the function for pin P0.0 as follows:

00	P0.0 as GPIO
01	RD
10	TXD3
11	SDA1

So, to have P0.0 function as TXD3 instead of GPIO:

```
PINSEL0 = 0x00000002;   // Assignments like this are not the best way, unless you are sure ↩
    that the remaining pins are GPIO
```

Note
Using 00 for any pin sets its function to GPIO. The reset value for PINSELx registers is 0x00000000. That is why the default function for all I/O pins after a reset is GPIO.

Exercise

Which `PINSELx` register(s) do you need to modify to have access to each of `EINT0`, `EINT1`, `EINT2`, and `EINT3`?

Refer to the [lpc1769-manual].

Tip

All `PINSELx` registers are fields in the `LPC_PINCON` structure.

Tip

Setting the function of a pin to external interrupt (`EINTx`) *enables* the corresponding interrupt.

3.1.2 Level and Edge Sensititivy

The `EXTPOLAR` and `EXTMODE` registers control external interrupt behavior.

Tip

Although these register are 32-bit, only the least significant 4 bits are used. Bit `0` controls `EINT0`, bit `1` controls `EINT1`, bit `2` controls `EINT2`, and bit `3` controls `EINT3`.

`EXTMODE` selects whether external interrupts are *level*-sensitive (`0`) or *edge*-sensitive (`1`).

`EXTPOLAR`, the External Interrupt Polarity Register, controls which level or edge on each pin will cause an interrupt (depending on `EXTMODE`):

1. If `EXTMODE` is set to level sensitivity, setting a bit in `EXTPOLAR` to a `0` specifies that the corresponding external interrupt is *LOW-active* (triggered by the `0` level), and setting it to a `1` makes it *HIGH-active* (triggered by the `1` level).
2. If `EXTMODE` is set to edge sensitivity, setting a bit in `EXTPOLAR` to a `0` specifies that the corresponding external interrupt is *falling-edge sensitive*, and setting it to a `1` makes it *rising-edge sensitive*.

Both `EXTMODE` and `EXTPOLAR` registers are fields of the `LPC_SC` structure (`SC` for System Control).

3.1.3 Clearing Pending Interrupts

When an external interrupt request is received, an interrupt flag is set in the `EXTINT` register, which would assert the request to the NVIC. The 4 least significant bits in the `EXTINT` register indicate the pending external interrupts. For example, when `EINT0` is enabled and activated correctly, bit 0 in `EXTINT` will be set to `1` by the CPU.

Active bits in the `EXTINT` register *must be cleared in the ISR*. Otherwise, future similar events will not be recognized. To clear a bit in the `EXTINT` register, write `1` to it.

The `EXTINT` register is also a field in the `LPC_SC` structure.

For example, to clear all external interrupts:

```
LPC_SC->EXTINT |= 0xF;
```

3.1.4 Enabling External Interrupt in the NVIC

One of common things between GPIO and non-GPIO external interrupts is how to enable the interrupt in the NVIC using the `NVIC_EnableIRQ` function.

Note
As a matter of fact, *NVIC_EnableIRQ* is essential not only in external interrupts, but in any interrupt-based code.

Before enabling external interrupts in the NVIC, however, you must:

1. Enable the required external interrupt by modifying the `LPC_PINCON->PINSEL4` register correctly.
2. Configure the sensivity: level (high/low) vs. edge (falling/rising).

3.2 Interrupt Priority

In CMSIS, you can set interrupt priorities using the function:

```
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority);
```

- The fist argument is the interrupt ID or number (refer to Experiment 4).
- The second argument is simply a number representing the priority where 0 is the highest priority and 31 is the lowest priority.

4 Tasks

Build an experiment where two external interrupts result in an LED blinking 10 times, each with a different rate. The LED blinking is performed by the interrupt handler.

The faster rate interrupt should have a higher priority; if it is activated while the slow rate interrupt is being serviced, the slow rate interrupt handler will be paused to service the fast rate inerrupt and then come back to the *pending* slow interrupt.

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