Lab 6 – Interrupt Latency Test

In this lab, you will perform an interrupt latency test. The pic16f883 supports many interrupts, both internally and externally triggered. One I/O pin is designated for external interrupt, RB0/INT. This interrupt can be configured to respond to a rising or falling edge. A flag bit, INTF, will be set when an interrupt is detected. If this interrupt has been enabled, setting the flag will also trigger a jump to the interrupt vector (located at address 0x0004 in the program memory). See Chapter 14 in the data sheet for more details on interrupts.

In this lab, we will measure the interrupt latency to determine if the PIC responds to an edge on the input signal within an appropriate time. The interrupt pin, RB0, is tied to one of the TMUs (TMU3A), which allows us to precisely measure the time between an event on this pin and another pin. Pin RB1 is tied to TMU3B. We will use this pin as an output, and generate a pulse on this output pin when an interrupt is detected. The time between the two rising edges will allow us to determine the response time. We will use QTMU operations, very similar to Lab #4, except that this time we will be triggering the start and stop on two different signals. Previously, we triggered both the start and stop on TMU0 DirectA, since the signal on RA6 was the only signal observed. This time the start will be triggered by a rising edge on TMU3 DirectA (pin RB0), and stop will be triggered by a rising edge on TMU3 DirectB (pin RB1). You will need to set up and arm the TMU, then generate a rising edge on RB0, and then collect the measurement for the time between the rising edges.

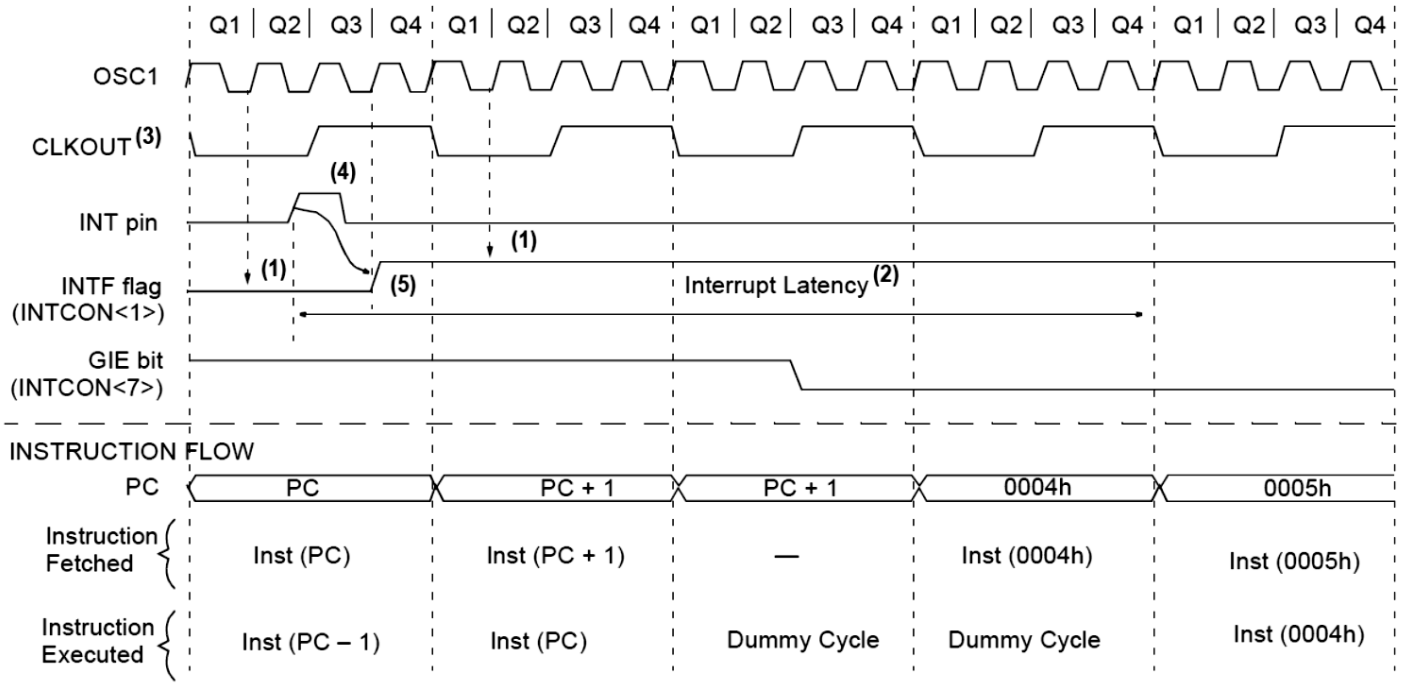
For this lab, you will need to program the PIC with a new test program. This program should respond to a rising edge on the interrupt pin by generating a high pulse on an output pin, RB1. For this, you will need to enable the external interrupt on pin RB0, clear the interrupt flag, enable global interrupts, set RB0 as an input and RB1 as an output, and implement an interrupt handler at the interrupt vector. When interrupts are enabled, an interrupt will cause execution to jump to the interrupt vector. This is accomplished by pushing the interrupt vector (0x0004) into the program counter (PC). The current value in the PC will be pushed onto the stack so that it can be retrieved when you return from the interrupt handler. This is all handled automatically when interrupts are enabled. What you will need to provide is the code that will be executed at the interrupt vector (the interrupt handler function) terminated by a RETFIE operation, which will cause execution to return to the previous PC location (address stored on top of stack). Your interrupt handler should generate a high pulse on RB1.

Your program should be written in assembly so that you can tightly control and determine how many instruction cycles will be executed on each interrupt before the signal on RB1 goes high. The expected response time will be based on the specified latency and the instruction delays. *You will need to determine what this response time should be based on your program and the interrupt latency specified in the data sheet*. See Figure 14-8 in the data sheet for a detailed waveform of the interrupt latency response. Also, see Figure 17-5 for more details of the instruction cycle. Keep in mind that each instruction on the *pic16* requires 1 instruction cycle (except jump operations, which require 2), and each instruction cycle takes 4 clock cycles. Take multiple samples and datalog the min and max response times.

Write-up your results
Record your test time. Turn in your test program code, test application code and datalogs. Also print the Program Memory listing from MPLab (opcodes for assembly program). What response time range did you determine was appropriate for your test? Discuss your test results.

Due: Wednesday, May 18, 2016

FIGURE 14-8: INT PIN INTERRUPT TIMING



Note 1: INTF flag is sampled here (every Q1).

2: Asynchronous interrupt latency = 3-4 TCY. Synchronous latency = 3 TCY, where TCY = instruction cycle time. Latency is the same whether Inst (PC) is a single cycle or a 2-cycle instruction.

3: CLKOUT is available only in INTOSC and RC Oscillator modes.

4: For minimum width of INT pulse, refer to AC specifications in **Section 17.0 "Electrical Specifications"**.

5: INTF is enabled to be set any time during the Q4-Q1 cycles.

FIGURE 17-5: CLKOUT AND I/O TIMING