**University of Hawaii**
**Department of Electrical Engineering**

# EE 361L Digital Systems and Computer Design Laboratory

# Lab 2.3: Hardware Interrupts

## 1. Introduction

The purpose of this laboratory experiment is to introduce interrupts on the PIC 16F84A. An interrupt is a useful feature on any micro-controller and it will be explained in ***Section 2***. ***Section 3*** has a design problem using an interrupt.

## 2. Interrupts

Here is an example to illustrate the concept of an Interrupt. Suppose we have a PC that is normally used to play video games as shown in *Figure 1*. The PC also has Internet telephony software that makes it work like a telephone. The computer user would like to play the game until he/she gets a telephone call. Then the user would like to talk on the telephone. When the conversation is over, the user would like to resume playing the game. Now, the telephone connection acts like an *interrupt* (interruption) to the game, and the telephone software has to step in (*interrupt handler)*.

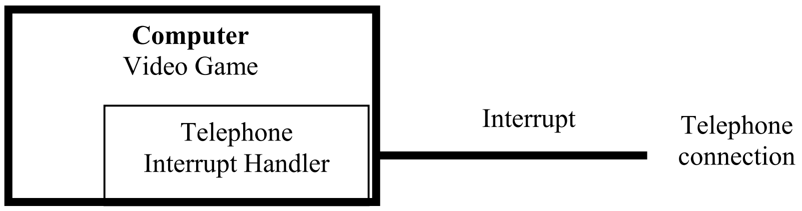

**Figure 1. Computer that runs video games and a telephone.**

Now, interrupts can be implemented by hardware or by software. A software interrupt is pretty much like a function call in C. A hardware interrupt is also like a C function call except that *invoking the call is done by hardware...ie a signal from a pin*. Before explaining interrupts further, we will take some time to review a function call. The following is an example of a function and a function call.

```
1.  int sum;  // Global variable
2.
3.  main()
4.  {
5.        int i;
6.        int j;
7.
8.        sum = 0;
9.        for (i=0; i<100, i++) {
10.                 incr2();
11.       }
12. }
13.
14. void incr2()
15. {
16.       sum++;
17.       sum++;
18. }
```

Note that lines 14-18 belong to the function *incr2*. Also note that *main* calls the function *incr2* at line 10. Note that *incr2* is invoked every time line 10 is executed by the computer.

Interrupt handlers are useful whenever a computer has a main program to execute, but also another *critical* task that is performed less frequently. I*n hardware*, this other task is invoked by a signal to a pin on the micro-controller. This pin and the associated circuitry are called as the *interrupt*. On the PIC 16F84A, pin RB0 is an interrupt. An *interrupt handler* is like a function. The following is an example of a simple program *main* and an interrupt handler *clear*.

```
1.  int sum;  // Global variable
2.
3.  main()
4.  {
5.        sum = 0;
6.        while(1){
7.                  sum++;
8.                  sum++;
```

```
                sum--;
        }
}

void interrupt clear()                  // Interrupt Handler
{
        sum = 0;
}
```

Note that *main* runs a while loop that loops around forever. Each pass through the loop, it increments the variable *sum* three times. Now, as long as the interrupt pin does not get a signal, the *main* routine goes on undisturbed. But when the interrupt gets a hardware signal, the micro-controller immediately jumps to the interrupt handler *clear*, executes it, and returns back to where it came from. Note that *clear* will set the variable *sum* to zero.

Note that in order for the (RB0/INT) pin to operate as an interrupt, it must be programmed to be an interrupt. This can be accomplished by enabling the GIE and INTE bit of the INTCON Register. The interrupt will be generated only when this bit is set, or else it would be ignored. Refer to the Special function registers in the Datasheet of PIC16F84A for further details. Here is a complete program for the PIC 16F84A that illustrates the use of an interrupt to reset a Counter. *Complete your understanding of the following program before going on to the assignment in **Section 3**.*

```
#include <pic1684.h>

int count;  // Global variable that is the state of a counter
void backup(void);
//  The main routine constantly increments "count" and calls the function "back-up"

main()
{
        INTCON=0b10010000;
        //Check what is being initialized in the above statement

        count = 0;

        while(1) {
                count++; // Increments count 4 times
                count++;
                count++;
                count++;
                backup(); // Decrements count 3 times
        }
}
// The function backup will decrement the "count" three times
void backup()
{
        count--;
        count--;
        count--;
}

// This is the interrupt handler.  It clears "count".
void interrupt clear_count()
{
        count = 0;
        INTCON=0b10010000;
        // Why do u have INTCON here?
}
```

## 3. Assignment

The Sheraton Waikiki hotel has a restaurant on its top floor. There is a single express elevator that connects the ground floor of the hotel directly to the restaurant. Say a guest calls the elevator. The elevator keeps the door open for 6 seconds, before it actually starts moving to the desired floor level. Every time the door starts closing and the '*Open*' button is pressed from the inside, the elevator keeps the door open for another 6 seconds. The time taken by the elevator to move between the floors is 18 seconds and the time to close the doors is 1 second.

Implement the above system for the elevator **by using the External interrupt pin of the PIC16F84A**. Identify the Interrupt vector used by the PIC16F84A and also report the status of the *Special Function Registers* that you might have used. Identify the size of the *Stack* provided in the PIC and show what happens when a *Stack Overflow* condition occurs. In your lab report, include the C program and also indicate the problems that you might have encountered.