Using Interrupts with the C167

1. Introduction

This document explains how to program interrupt service routines (ISRs) for the C167 microcontroller. An interrupt, as the name suggests, refers to an interruption of the microcontroller (or microprocessor) execution cycle.

As a simple example, consider what happens when you press keys on your keyboard. When you press or release a key, that event is signalled up the keyboard cable to raise an interrupt. It is the operating system's job to watch for such interrupts. For each possible kind of interrupt, there will be an interrupt handler (or interrupt service routine). This is a piece of code that an operating system executes whenever an interrupt associated with the interrupt handler occurs. For instance, the keyboard interrupt handler has code that may store characters in a buffer.

Once an interrupt handler has finished executing, the processor starts off from where it was interrupted. Of course this is a very simple explanation of how an interrupt handler works. It is NOT the purpose of this document to explain all the behind the scenes stuff related to interrupt handler.

2. Who gives a f#@! about interrupts?

Interrupts guarantee\(^1\) the execution of code when the interrupt occurs. This is useful in many contexts. For instance, suppose you want to sample a data only once every 15 milliseconds. You can write an interrupt service routine and hook it to the timers of your C167.

3. Sample Interrupt code

The code below should flash the LED connected to port 2 bit 0 (D21) once every second.

```c
void Init(void) {
    /// ------------  Timer 3 Control Register ----------
    ///  timer 3 works in timer mode
    ///  prescaler factor is 512
    ///  up/down control bit is reset
    ///  external up/down control is disabled
    ///  alternate output (toggle T3OUT) function is disabled
    T3CON = 0x0006;
    T3    = 0x5F00;  //  load timer 3 register
```

\(^1\) When you have multiple interrupts, you can still prioritize them in the C167 to control which one executes first.
/** enable timer 3 interrupt**
**/ timer 3 interrupt priority level(ILVL) = 1
**/** timer 3 interrupt group level (GLVL) = 0
T3IC = 0x0044;

T3R = 1;   // set timer 3 run bit

// Initialize port 2
DP2  = 0x0001;
P2 = 0x0000;
}

void GT1_viIsrTmr3(void) interrupt T3INT {
P2 = ~P2;
}

void main(void) {
    Init();   // GLOBALLY ENABLE INTERRUPTS
    P2 = ~P2;

    while(1)
    {
    
    }
}

The salient features of the code above are:

**a. Initializing the timer registers.**
Read chapter 9 – The General Purpose Timer Units in your C167 hardware manual for more information.

**b. Setting the interrupt priority and group level.**
You can read chapter 5 – Interrupts and Trap Functions in your C167 hardware manual. But, I only read pages 5-1 to 5-8 and skimmed through the rest of the chapter.

**c. Writing the interrupt service routine.**
Should be self explanatory. Of course, if you are interrupting every 1 ms, you should make sure that your ISR executes within 1 ms! If not, your interrupts will start “piling up”. Thus, please do not use printfs or a lot of floating point calculations in your ISRs. They take considerable time and can easily extend the execution of your ISRs into the hundreds of milliseconds.

4. Conclusion

   This document is just an introduction. You should be able to do a lot from here. The trick is not to be afraid to explore!