EE 2004
Week 6 Tutorial

1. Creating delay using loops

Motivations of implementing delays in microcontroller

Microcontroller is a very fast device that executes many instructions in a few microseconds. There are situations in which we want to put the microcontroller in a waiting state. Two examples are given below:

a. In some situations, we have to put the microcontroller on hold in order to demonstrate the effects of the operations performed by the microcontroller. For example, to visualize the 8-bit LED module is actually counting up, we have to implement delays between each increment. Without the delay, the LED module would count up too quickly and human eyes are not able to detect the counting operation due to the persistent of vision phenomenon. All bits just appear to light up all the time.
b. Peripherals usually would not respond as quickly as the microcontroller. When interfacing with them, the microcontroller must be put on hold to give the peripheral enough time to respond. For example, when interfacing an EEPROM, the microcontroller must give the EEPROM enough time to perform the requested read/write operations and delays must be implemented between consecutive read/write operations.

1.1. Calculate the time delay generated by the following code fragment if [DELAY_L] and [DELAY_H] are set to be 0x00 and 0x04 respectively. Validate your answer using the Stopwatch tool (Debugger → Stopwatch) in MPLAB Sim.

```
DelayLoop:  decfsz  DELAY_L, F, A
            bra DelayLoop
            decfsz  DELAY_H, F, A
            bra DelayLoop
```

1.2. Using the Stopwatch tool, validate that the bra instruction takes two machine cycles to execute. Explain why two machine cycles are required.

1.3. Change the two lines of "`bra DelayLoop`" to "`goto DelayLoop`". Determine the time delay generated for the same initial values of [DELAY_L] and [DELAY_H]. Again, validate your answer using the Stopwatch tool (Debugger → Stopwatch) in MPLAB Sim.

1.4. Use the loop provided in Question 1.1 as a building block, write a subroutine to generate a 8ms delay.

2. Subroutine and stack

Answer questions below using the following code listing:

```
Program
Memory        LINE            SOURCE
Address


              00012 COUNT equ 0x00
              00013 MyReg equ 0x01
              00014 DELAY_H equ 0x02
              00015 DELAY_L equ 0x03
              00016
              00017 ORG 0x000000
000000        00018 goto Main
              00019 ORG 0x000030
000030        00020 Main:     movlw 0x00
000032        00021           movwf COUNT
000034        00022           movlw 0xFF
000036  D???  00023 Back:     rcall Display
000038 EF?? F??? 00024        goto Back
00003C        00025 Display:  decfsz COUNT, F
00003E        00026           bra Display
000040        00027           movf COUNT, W
000042        00028           movwf PORTB
000044 EC?? F??? 00029        call Delay
000048        00030           return
00004A        00031 Delay:    clrf    DELAY_H
00004C        00032           clrf    DELAY_L
00004E D???   00033 DelayLoop: rcall DelayLoopLow
000050        00034            decfsz  DELAY_H
000052 EF?? F??? 00035              goto    DelayLoop
000056        00036            return
000058        00037 DelayLoopLow: decfsz  DELAY_L
00005A D???   00038               bra DelayLoopLow
00005C        00039               return
```

2.1. Identify the contents of the stack, stack pointer and the TOS register after the execution of Line 23, 29, 30 and 33. Step through the program by pressing F7 and inspect the contents of the “Hardware Stack” (Go to the “View” menu and select “Hardware Stack”) and the STKPTR, TOS and PCLAT (Go to the “Watch” window and type STKPTR, TOS and PCLAT under the “Symbol Name” column) to validate your answers.

2.2. Determine the opcode of the instructions in Line 23, 24, 29, 33, 35 and 38 and validate your answers by inspecting the .lst file.

2.3. What is the difference between the `call` and `rcall` instructions?