ECE251 - PIC Assignment 1

Henry Son

May 14th, 2020

1 Introduction

For this assignment, the two tasks were to (1) have two inputs, one that disables the counter and another that forced the counter to count by 2, and (2) implement a subroutine that provides a delay, in milliseconds, equal to the number contained in the W register when the subroutine is called. This assignment was completed using the Debugger as the simulator on MPLABxIDE. The chip that was simulated was the PIC 16F877A.

***Note:** All files used for this assignment can be found here: Assignment 1.

2 Implementation

2.1 Two Inputs

In order for the chip to take two inputs, TRISB has to be set as an input. For the purpose of the assignment, only pins 1 and 2 were set as inputs.

```
movlw   0x06
movwf   TRISB
```

After the pins were 'initialized', the first pin was used to enable/disable the counter. This was done by using *btfss* followed by a *goto*. This works by checking to see if PORTB at pin 1 is either 0 or 1. If PORTB at pin 1 is 0, it will move onto the *goto* function and loop back to L1. However, if it is 1, it will skip the *goto* function.

```
btfss   PORTB,1
goto    L1
```

The state of the second pin will determine whether the counter goes up by 1 or 2. To accomplish this, the *btfsc* function was used, followed by two *incf* functions. Now with this function, if it detects that pin 2 on PORTB is 0, it will skip the first of the two *incf* functions, so the counter will go up by 1. Likewise, if it detects that the pin is 1, it will run both *incf* functions, so the counter will go up by 2.

```
btfsc   PORTB,2
incf    COUNT,F
incf    COUNT,F
```

2.2 Delay Subroutine

In order to move the previous delay mechanism implemented in the demo program as a subroutine, a different name was used, a return call was added, and *call* was used instead of *goto*. Now before the DELAY routine is called, the number of millisecond the DELAY should prolong for should be within the W register beforehand. Therefore, a *movlw* function was used. In the beginning, only '0x1' was used in order to determine how long the DELAY subroutine would take. Next, after setting break points at the DELAY subroutine was called and at the function immediately following it, the length of time the subroutine would take was determined by looking at the 'Logic Analyzer' graph at the two break points and taking the difference of the two times. An extra 'VAR_K' variable was added because each variable can only hold up to 255, and 1000 was needed.

```
movlw   0x1
call    DELAY
goto    L1
```

3 Final Code

Code Download – Video Download

```
; ECE251 - Comp Arch
; Modified 8-bit counter
; Henry Son (@sonbyj01)
;
        LIST    P=PIC16F877A
        include <p16F877A.inc>

        __CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF & _LVP_OFF

COUNT   EQU     0x20            ; General purpose registers used for variables
VAR_I   EQU     0x21
VAR_J   EQU     0x22
VAR_K   EQU     0x23

        ORG 0x00
        goto    start

        ORG 0x10
start
        bsf     STATUS ,RP0     ; Bank 1
        movlw   0x00            ; All bits output
        movwf   TRISD           ; in PORTD
        movlw   0x06            ; 1 and 2 are input
        movwf   TRISB           ; in PORTB
        bcf     STATUS, RP0     ; Bank 0

        clrf    COUNT           ; Set COUNT to 0
```

```
L1      movf    COUNT, W        ; Get value of COUNT
        movwf   PORTD           ; and write it to PORTD
        btfss   PORTB,1         ; 0 -> disable counter
        goto L1

        btfsc   PORTB,2         ; 0 -> increment by 1
        incf    COUNT, F        ; increment count by 1
        incf    COUNT, F

        movlw   0x3             ; ~3 ms delay
        call    DELAY
        goto    L1

DELAY   ; subroutine delay for around ~1 ms
        ; Stall
        movwf   VAR_I

L2              movlw   .200    ; Number of iterations for inner loop
                movwf   VAR_J

L3                      movlw   .5
                        movwf   VAR_K

L4                              nop     ; Waste time : 10 cycles :
                                nop     ; ~961 nano sec/L3cycle
                                nop
                                nop
                                nop
                                nop
                                nop

                        decfsz VAR_K, F
                        goto L4

                decfsz  VAR_J, F
                goto    L3

        decfsz  VAR_I, F
        goto    L2

        return

        END
```