clrf f → clear f

↓

The contents of register 'f' are cleared and Z-bit in status register is set.

clrw → clear Working Register

↓

W register is cleared. Zero bit (Z) in Status register is set.

decf f,d → Decrement file register f

↓

Decrement register 'f', if 'd' is 0, the result is stored in W register If 'd' is 1, the result is stored back in register 'f'.

Operation: (destination) ← (f) - 1

Ex: decf 0x06,W

(W) ← (0x06) - 1

↓

content of PORTB

decfsz f,d → Decrement f, Skip if 0

The contents of register 'f' are decremented. If 'd' is 0, the result is placed into the W register. If 'd' is 1, the result is placed back in register 'f'.
If the result is 1, the next instruction is executed. If the result is 0, the next instruction is skipped, i.e, a NOP (no-operation) is executed

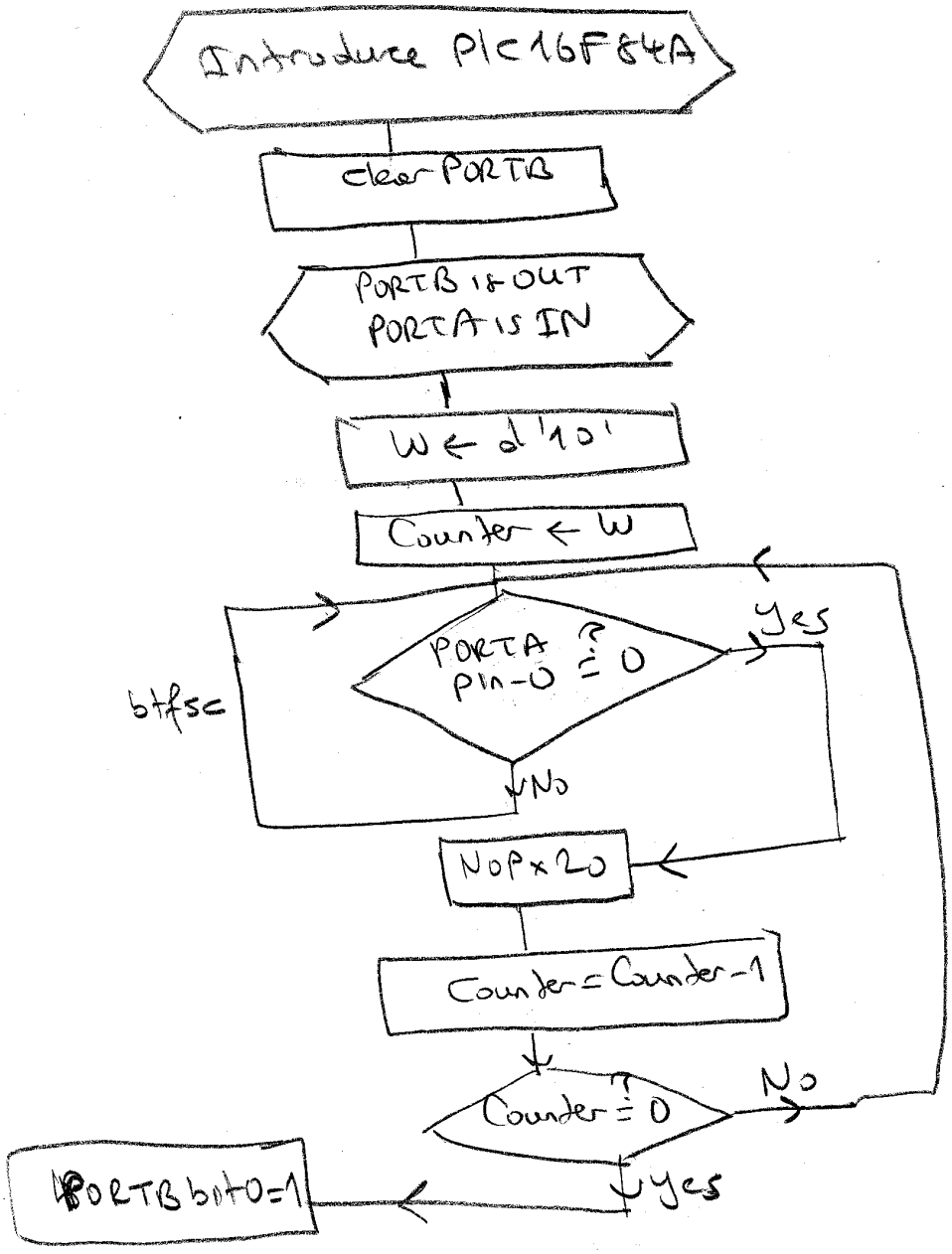
Ex:
```
movlw 10;     (w) ← 10 (decimal 10)
movwf 0x0C;   0x0C is a register address.
decfsz 0x0C,F;  (0x0C) ← (0x0C) − 1
              ; Content of (0x0C) is
              ; decremented and loaded
              ; into register whose address
              ; is 0x0C
decfsz 0x0C,W; (w) ← (0x0C) − 1
              ; w contains 8
              ; (0x0C) contains 9
```

Ex:

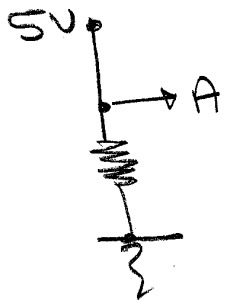
Assume that PORTA is programmed as input and PORTB is programmed as output. Write a program to achieve the following goal

- If the button connected to pin-1 of PORTA is pressed 10 times, the LED connected to pin-0 of PORTB is turned-ON.

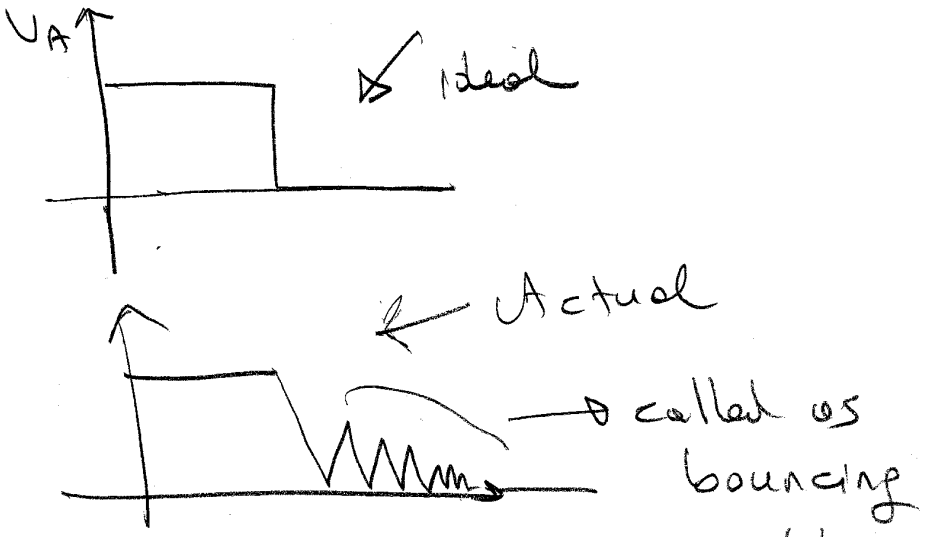
Sln: Flow Chart of the Program is As Follows:



Bouncing Problem



If 5V is disconnected from point A the voltage on point A does not change suddenly



Bouncing problem may be prevented either by hardware of software.

In hardware prevention technique We use logic units such as flip-flops.

In software prevention method, delay routines or NOP command is used.

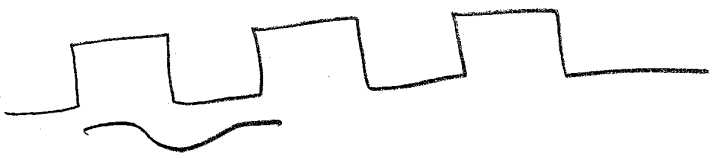
```
        list P=16F84A
        include 'P16F84A.INC'

Counter equ 0x0C; address of counter register
        clrf PORTB; clear PORTB
        bsf STATUS,RP0; go to Bank1
        clrf TRISB; PORTB is OUTPUT
        movlw 0xFF;  } ⇒ PORTA is INPUT
        movwf TRISA  }
        bcf STATUS,RP0; go to Bank0

Start;  Main program starts here
        movlw 10;
        movwf Counter;

Test    btfsc PORTA,1; PORTA-pin1 = 0 ?
        goto Test; Repeat loop
        nop; no-operation, 1 cycle delay
        nop; no-operation, 1 cycle delay
        nop;
        nop;
        nop;
        nop;
        nop;
        nop;
        nop;
        nop;
        nop;
        nop;
        decfsz Counter,F; (Counter)←(Counter)-1
                        ; check Counter=0 ?
        goto Test;
        bsf PORTB,0; set PORTB-Bit0
        end
```

Writing Delay Program Segments

4mHz oscillator $F = 4mHz \rightarrow$ Frequency

$T \rightarrow$ Period $T = \frac{1}{F} \rightarrow T = \frac{1}{4 \times 10^6}$ sec

$= 0.25\,\mu sec.$



1 Cycle $= 4 \times T$

$= 0.25 \times 4$

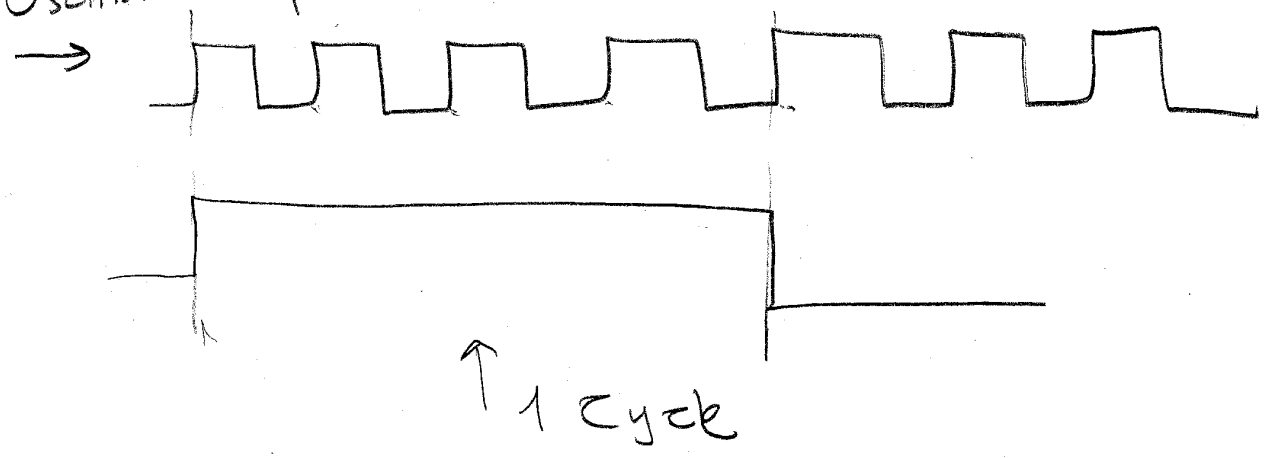
$= 1\,\mu s$

Ex: If $F = 16mHz$

$1\ Cycle = 4 \times \frac{1}{16 \times 10^6}$

$= 0.25\,\mu sec$

Oscillator Freq.



PIC Commands need 1 Cycle for execution except for the followings

goto ↔ 2Cycles
call ↔ 2cycles
return ↔ 2cycles
decfsz f,d → 1 Cycle if f≠0 2Cyle if f=0
retlw ↔ 2Cycles
retfie ↔ 2Cycles
incfsz f,d → 2Cycles if f=0 else 1Cycle
btfsc f,b ↔ 2Cycles if f(b)=0 else 1Cycle
btfss f,b ↔ 2Cycles if f(b)=1 else 1Cycle

Ex's // Writing a delay segment

```
Counter equ 0x0C;
        movlw 10;      → 1 Cycle
        movwf Counter; → 1 Cycle
Loop
        decfsz Counter, F; → 9x1+2 Cycle
        goto Loop          → 9x2 Cycle
```

In Total ⇒ 2 + 11 + 18
= 13 + 18
= 31 Cycle → 31 μsec

Ex°

```
Counter  equ  0x0C
N        equ  0XFF ; ← Write any number
                   ; Here

         movlw  N;      → 1 Cycle
         movwf  Counter; → 1 Cycle

loop     decfsz Counter,F; → (N-1)x1+2 Cycle
         goto   Loop;      → (N-1)x2 Cycle
```

In Total : $1+1+(N-1)+2+2N-2$

$= 3N+1$

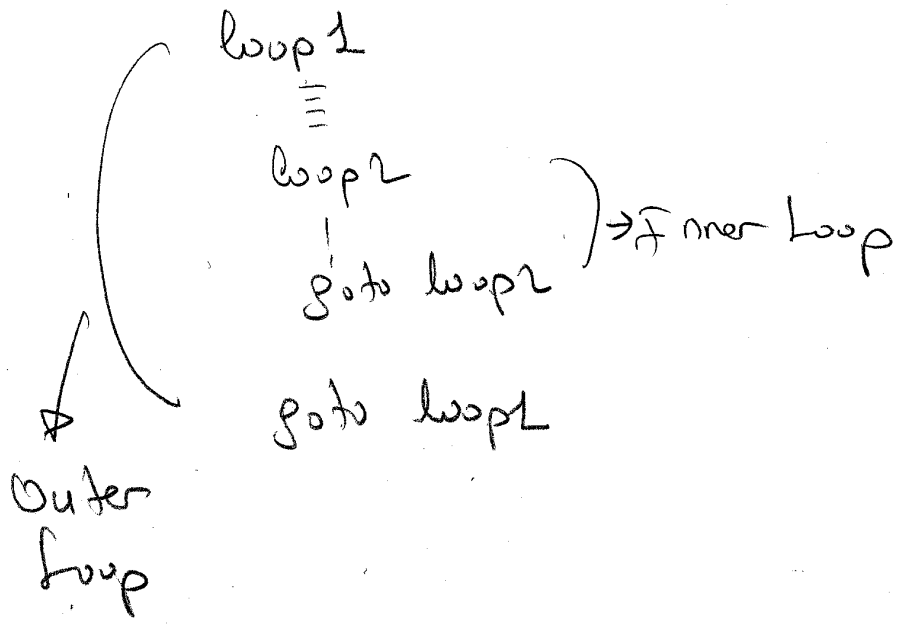
$\approx 3N$

If N = 0XFF i.e, N = 255

$3N = 765$ μsec

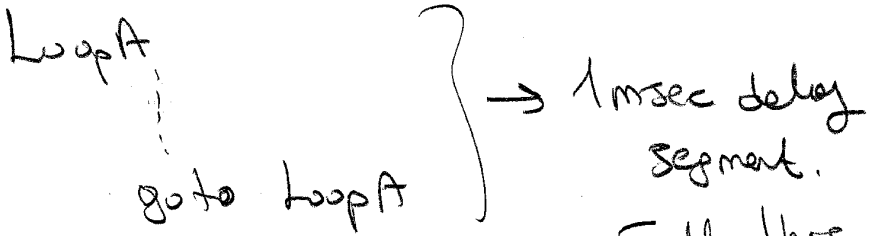
≈ 0.8 msec

Concatenated Loop Delay Segments



Loops can be used inside other loops to create large delays, for instance 1 sec delay, 100 msec delay e.t.c.

Ex: Assume that the following segment is 1msec delay segment



Now consider the following

```
Counter equ 0x0C;
        movlw 100;
        movwf Counter

LoopB
        A; <- 1 msec delay segment is written here
           Assume

        decfsz Counter;
        goto LoopB;
```

Now consider the delay amount produced by the above segment

```
LoopB
        A       ; Executed 100 times 100x1 msec
        decfsz Counter; 99x1+2 Cycle
        goto LoopB ; 99x2 Cycle
```

Total Delay = 100msec + 3x99+2 Cycle

= 100msec + 299 Cycle

= 100msec + 299.0x25μsec

= 100msec + 75μsec

= 100msec + 0.075msec

= 100.075 msec ≈ 100 msec

Ex: Write 1msec delay segment for 4mHz oscillator

Sln: For 1msec we need

$$\frac{1\text{msec}}{0.25\,\mu\text{sec}} = \frac{1000\,\mu\text{sec}}{0.25\,\mu\text{sec}}$$

$$= 4000 \text{ Cycles}$$

First methods

```
        movlw   250,
        movwf   Counter
Loop    NOP;
        NOP;
        Nop;
        Nop;
        NOP; NOP; NOP; NOP;
        Nop; NOP; NOP; NOP; NOP;
        decfsz  Counter; 249x1+2 Cycles
        goto    Loop;    249x2 Cycles.
```

13 NOP operations

Total Cycles = 249x1+2+248x2
+ 13x250
= 4000 Cycles