Chapter 4: Timing - Delay Routines

Australia's Global University | Faculty of Engineering | School of Electrical Engineering and Telecommunications

Professor Eliathamby Ambikairajah

Head of School of Electrical Engineering and Telecommunications, UNSW, Sydney

Delay Loops

- ✓ Sometimes it may be necessary to implement a fixed delay in a PIC assembly program and it can be done using a short delay loop
- ✓ PIC instructions are executed in **one** instruction cycle (except for branching, which takes **two** instruction cycles).
- ✓ One instruction cycle = 4/clock frequency = 4/4MHz = 1μs
- ✓ Time delays are generated using delay loops.
  - Normally a memory location is set up as a counter and the counter is loaded with a number.
  - This number is decremented repeatedly in a loop until the counter reaches zero.
- ✓ The delay loop is normally written as a **subroutine**

Delay Routine

- **An example of a delay loop 'T_L ms' is shown below:**

```
counter1 EQU     H'20'       ;Memory location H'20' is used as a counter1

;First subroutine starts here

delay     movlw   D'255'        ;load w register with Decimal D1 = 255 i.e H'FF'
          movwf   counter1      ;moving  D'255' into the memory location H'20'

delayms   nop                   ;1 instruction cycle
          nop                   ;1 instruction cycle
          decfsz  counter1,1    ;1 instruction cycle (when not branching)
          goto    delayms       ;2 instruction cycles
          nop                   ;1 instruction cycle
          return                ;2 instruction cycles
```

00H	Indirect address
03H	STATUS
04H	FSR
05H	PORTA
20H	counter1
: 7FH	GPR 96 Bytes
Bank 0	

When the counter1 is zero, next line (goto) is skipped and "nop" is executed.

Delay (T_L)= [1+1+(D1 -1)(1+1+1+2) +1+1+1+2+2] cycles = [4+ 5D1] *1μs

D1= 255 provides a delay of 1.279 ms
D1 =200 provides a delay of 1.004 ms

Group Exercise: Delay routines

- **Find the number of instruction cycles taken for the delay routine below:**

```
; assume that the counter content is 'k'
delay      decfsz   counter          ;1 instruction cycle (when not branching)
            goto    delay            ; 2 instruction cycles
```

Answer: 2 + 3 (k-1) instruction cycles

- **Find the number of instruction cycles taken for the delay routine below:**

```
;Delay Routine
delayms   movlw    D'250'      ; 1 instruction cycle
          movwf    counter1    ; 1 instruction cycle  (counter1 is memory location H'20')
loop_ms   nop                  ; 1 instruction cycle
          decfsz   counter1,1  ;1 instruction cycle (when not branching)
          goto      loop_ms    ; 2 instruction cycles
          return               ;2 instruction cycles
```

Answer: ? instruction cycles

- **Find the number of instruction cycles taken for the delay routine below:**

```
;Delay Routine
delay     movlw    D'153'     ;1cycle
          movwf    counter1   ;1 cycle
delay1:   decfsz   counter,1  ;if zero skip the next instruction
          goto     delay1     ;not zero goto delay1
delay2:   decfsz   counter,1  ;1 cycle when not branching
          goto     delay2     ;2 cycles
delay3:   decfsz   counter,1  ;1 cycle when not branching
          goto     delay2     ;2 cycles
          return              ;2 cycles
```

Answer: 1998 instruction cycles

Nested Delay Loops

- ✓ The delay loop program shown in the previous slide (slide 2) provides only a short delay
- ✓ There are many occasions where we need a longer delay.
- ✓ One way to obtain this, is by using a nested delay loop where a second delay subroutine calls the first delay loop within its loop. See the example below:

```
        counter   equ       H'21'   ;Memory location H'21' is used as a counter

;Second subroutine starts here
delay   movlw     D'100'            ;load w register with Decimal 100
        movwf     counter           ;Load the counter with k = 100
dloop2  call      delay_ms          ;delay =1.004 ms (=1004 cycles) 2 instruction cycle
        decfsz    counter,1         ;1 instruction cycle (when not branching)
        goto      dloop2            ;2 instruction cycles
        return                      ;2 instruction cycles
```

Delay (T_L)= [1+1+(k -1)(1004+2+1+2) +1004+2+2+2] cycles = [3+ 1009 k] *1μs

k= 100 provides a delay of 100.9 ms

- ✓ A longer delay can be also obtained by using a loop within a loop principle within a single subroutine

Group Exercise: Nested Delay routines

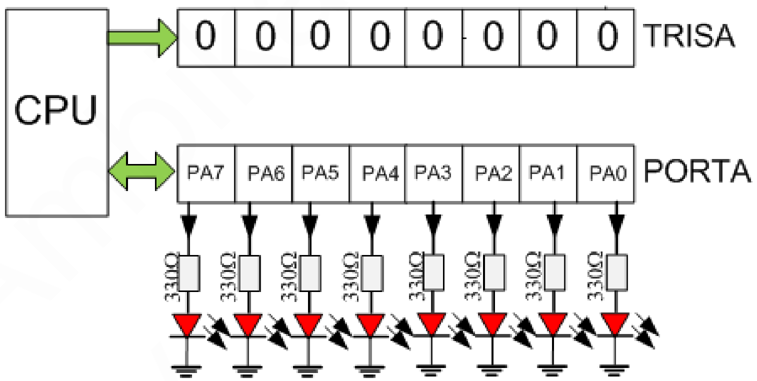
- **Determine the number of instruction cycles taken for the delay routine below:**

```
;Delay Routine

delay           movlw   D'120'
                movwf   counter1  ;counter1 is memory location H'20')
;
delay_Psec:     movlw   D'250'
                movwf   counter2  ;(counter2 is memory location H'21')
;
delay_Nms:      movlw   D'250'
                movwf   counter3  ;(counter3 is memory location H'22')
;
delay_Mms:      nop
                decfsz  counter3,1
                goto    delay_Mms
;
                decfsz  counter2,1
                goto    delay_Nms
;
                decfsz  counter,1
                goto    delay_Psec
;
                return
```

Answer: ? instruction cycles

Example: LED chaser

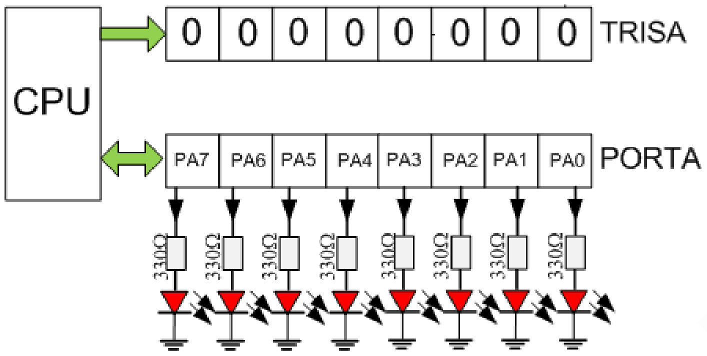
- **Eight LEDS are connected to Port A as shown in the diagram. Write down a program to turn on/off one LED after another, moving from left to right (i.e. LED walks) and repeat the sequence. Use a delay loop of 250 ms in your program.**

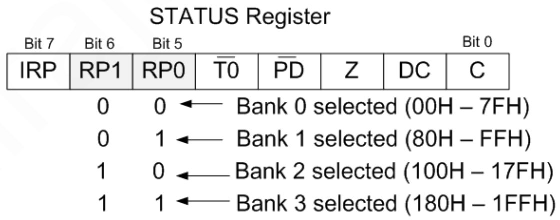



```
ORG         H'00'
            goto        init

;Intialisation routine starts here

init        BANKSEL  ANSEL
            clrw
            movwf       ANSEL
;
            movwf       ANSELH
            bcf         STATUS,6
            bsf         STATUS,5        ;select memory bank 1

            clrf        TRISA           ;set port A as output (TRISA = H'00')

            bcf         STATUS,5        ;select memory bank 0 (or BANKSEL POARTA)
            clrf        PORTA           ;all LEDs are turned off
;
```

80H	Indirect address
83H	STATUS
84H	FSR
85H	TRISA
86H	**TRISB**
87H	TRISC
A0H-EFH	(GPR) 80 Bytes
F0H - FFH	Accesses 70H -7FH
Bank 1	

Example: LED chaser ……



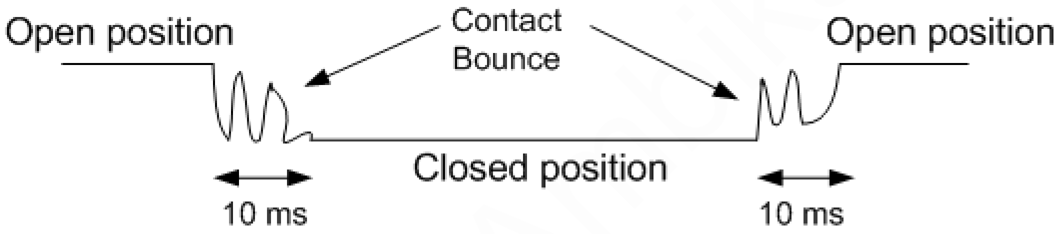



```
;Main program starts here
;
start        movlw    H'80'        ; load register W with H'80'
             movwf    PORTA        ;LED connected to the MSB will be on
;
shift_right  bcf      STATUS,0     ;carry bit =0;make sure carry is cleared (zero)
             call     delay        ;call the delay routine of 250ms
;
             rrf      PORTA,1      ;rotate right contents of PORTA through carry
             btfss    PORTA,0      ;test if we have reached PA0
             goto     shift_right ;
             goto     start
;
             END                   ;end of program
```

Write a delay subroutine of approx: 250 ms

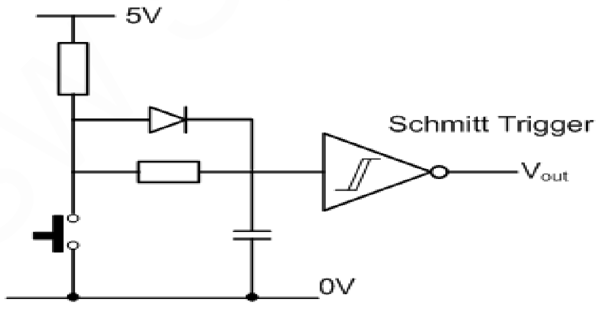
Eliminating Switch Bounce

- ✓ All mechanical switches have a bouncing property where the switch contacts open and close when a switch is pushed. The switch contacts normally bounce for about 10 – 15 ms before staying together(see diagram below). This is also true when the switch opens.



- ✓ The microcontroller may register many of theses contact bounces instead of registering one push. Therefore, a software solution is to:
  - Note the first detection of change in switch position
  - Wait for about 10 -15 ms (a delay loop)
  - Check the switch again to see if it is still pressed

- ✓ Hardware techniques based on latches and Schmitt triggers are also available.

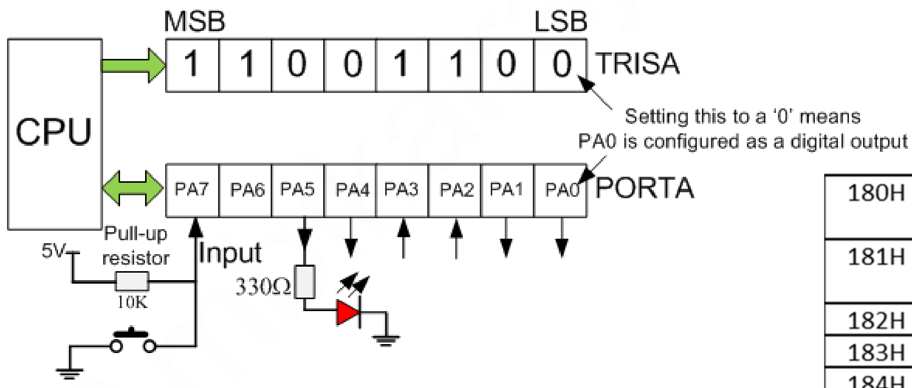


A switch debouncing circuit

Exercise: Draw a switch debouncing circuit using latches

Example: Switch de-bouncing using a delay loop

- **This program lights the LED if the switch is pressed and switch de-bouncing is included**



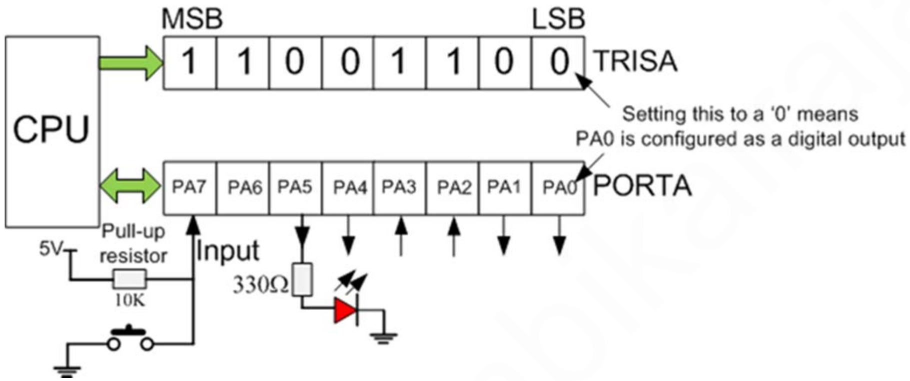

```
;
;
;
;
;Configure microcontroller
            list        p=16f886
            #include    <pic16f886inc>
;Configuration data for microcontroller
            …….         ……
            …….         ……
            ORG         H'00'       ;force program to start at reset vector
            goto        start       ;Go to the beginning of the initialisation program
;
;Intialisation
start       BANKSEL  ANSEL  ; select Bank 3
            clrf                    ANSEL
            BANKSEL  TRISA   ;select memory bank 1
            movlw                   B'11001100'
            movwf                   TRISA       ;program port A according
                                                ;to the above bit pattern
            BANKSEL  PORTA;
```

180H	Indirect address
181H	Option_REG
182H	PCL
183H	STATUS
184H	FSR
185H	SRCON
186H	TRISB
187H	BAUDCTL
188H	**ANSEL**
189H	ANSELH
18AH	PCLATH
190H - 1FFH	GPR 16 bytes
Bank 3	

00H	Indirect address
03H	**STATUS**
04H	FSR
05H	**PORTA**
06H	PORTB
07H	PORTC
20H : 7FH	(GPR) 96 Bytes
Bank 0	

80H	Indirect address
83H	**STATUS**
84H	FSR
85H	**TRISA**
86H	TRISB
87H	TRISC
A0H-EFH	(GPR) 80 Bytes
F0H - FFH	Accesses 70H -7FH
Bank 1	

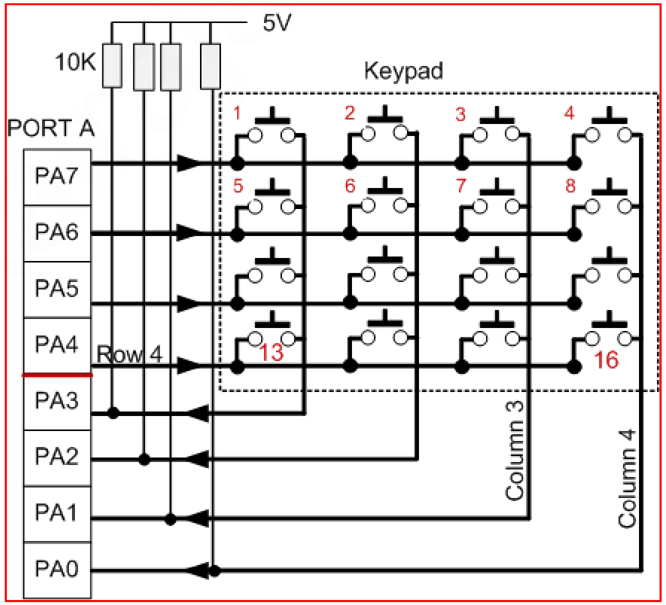
Example: Switch de-bouncing using a delay loop

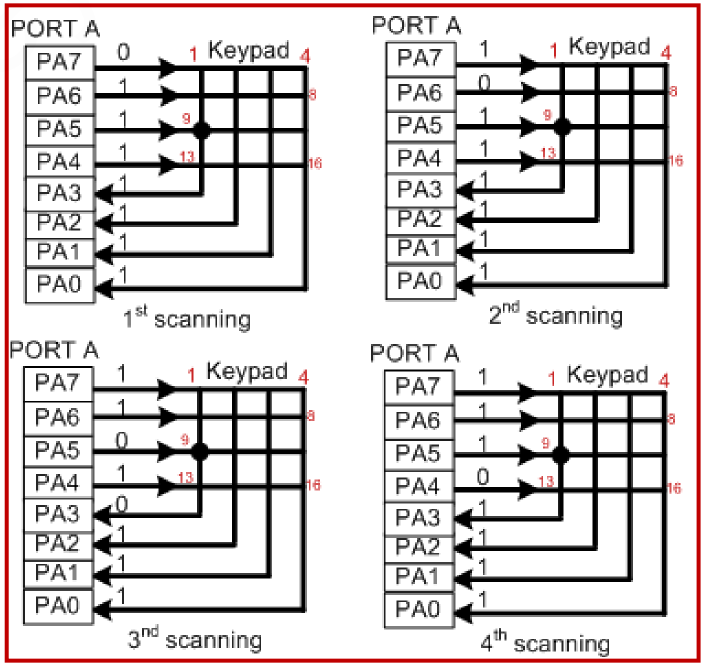



```
;Main program starts here
            bcf     PORTA,5    ;clear PA5 in port A, i.e LED off
wait4keyp   btfsc   PORTA,7    ;Test  bit 7 to see if key pressed ; branch if PA7= 0(pressed)
            goto    wait4keyp  ; keep checking the key
            call    delay15ms  ; call a delay routine of 15 ms
            btfss   PORTA,7    ;see if key still pressed; branch if PA7= 1( not pressed)
            bsf     PORTA,5    ;light the LED (set PA5=1). i.e. button is pressed.
;
wait4keyr   btfss   PORTA,7    ;Test  bit 7 to see if key pressed ; branch if PA7= 1(not pressed)
            goto    wait4keyr  ; keep checking the key to be released
            call    delay15ms  ; call a delay routine of 15 ms
            btfsc   PORTA,7    ; branch if PA7= 0(pressed)
            bcf     PORTA,5    ;turnoff  the LED (set PA5=1). i.e. button is released
            goto    wait4keyp  ;keep reading the status of switch
            END                ;end of  program
```

Keyboard Scanning

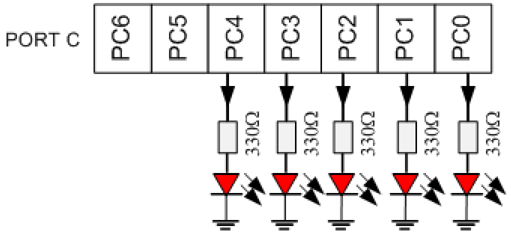
- ✓ A keyboard allows numeric or alphanumeric information to be entered and is widely used in photocopiers, central heating controllers etc
- ✓ A keyboard consists of touch-activated switches arranged in a matrix fashion (Rows and columns) as shown in the diagram
- ✓ When a key is pressed, it connects its row to its column
- ✓ The "Row-Scanning" technique is utilised in order to identify the pressed key
- ✓ Send a "walking-zero" pattern on the output lines (rows) and read the input lines (columns) and look for a zero

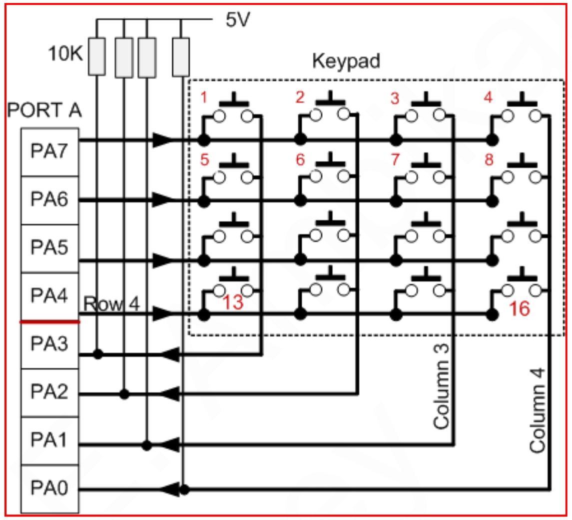




Example :Keyboard Scanning program

➢ **This program scans a 4x4 matrix keyboard connected to PORT A and when a key is pressed, the value of the key is coded and displayed by LEDS connected to PORT C.**



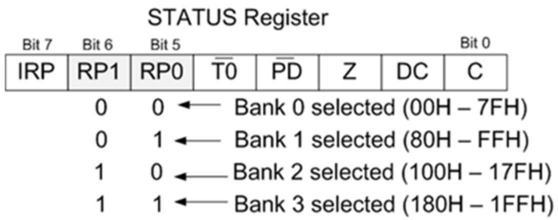


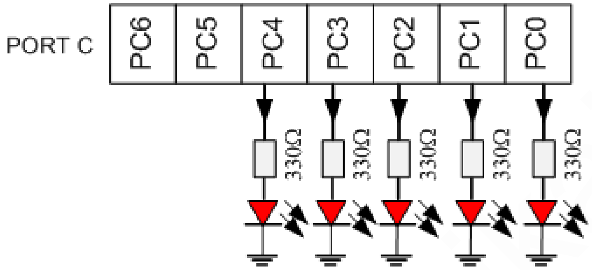
80H	Indirect address
83H	**STATUS**
84H	FSR
85H	**TRISA**
86H	TRISB
87H	TRISC
A0H-EFH	(GPR) 80 Bytes
F0H - FFH	Accesses 70H -7FH
Bank 1	

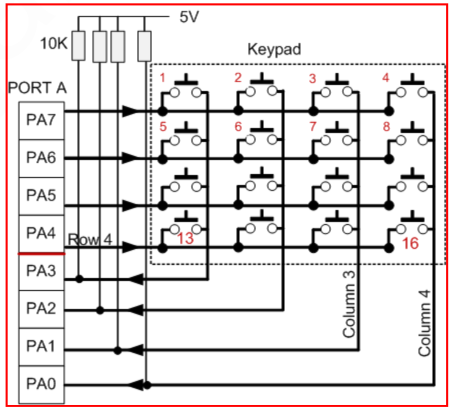
```
;Configuration data for microcontroller
store           equ         H'20'
count           equ         H'21'       …….         ……
                ORG         H'00'       ;force program to start at reset vector
                goto        start       ;Go to the beginning of the initialisation program
;Intialisation
start           BANKSEL     ANSEL       ;select memory bank containing ANSEL Register
                clrf        ANSEL       ;set PORTA to digital by clearing
                BANKSEL     TRISA       ;select memory bank containing TRISA Register
                movlw       B'00001111'
                movwf       TRISA       ;PA0-PA3 input and PA4-PA7 output
                clrf        TRISC       ;make PORTC all outputs
                BANKSEL     PORTC       ;select memory bank containing PORTA and PORTC Registers
                clrf        PORTC       ;reset PORTC (turn off all LEDS)
;
```

00H	Indirect address
03H	**STATUS**
04H	FSR
05H	**PORTA**
06H	PORTB
07H	PORTC
20H : 7FH	(GPR) 96 Bytes
Bank 0	

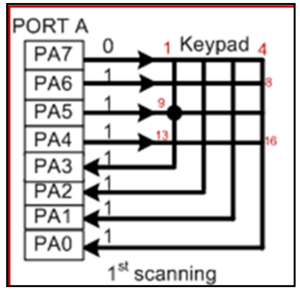
Example :Keyboard Scanning program








```
;Main program starts here
keyscan   bsf      STATUS,0    ;carry bit =1
          movlw    B'01111111'; first row pattern PA7=0
          movwf    store       ;store row pattern (location H'20')
          movlw    D'04'
          movwf    count       ;counter =4   (location H'21')
rowscan   movlw    store,0     ;move the content of store (row pattern) to W register
          movwf    PORTA       ;row  energising
          movf     PORTA,0     ;move PORTA contents to W register
          nop                  ;one cycle for PORTA contents to appear on pins
          xorwf    store,0     ;XOR operation to check key pressed
          btfsc    STATUS,2    ;if the result is zero no key pressed
          call     led_display; if key pressed find the key
          rrf      store,1     ;create next row pattern (result is put back in store location)
          decfsz   count,1     ; check all 4 rows have been energised and if so start from row 1 again.
          goto     rowscan     ;
          goto     keyscan     ;start again and keep scanning
```

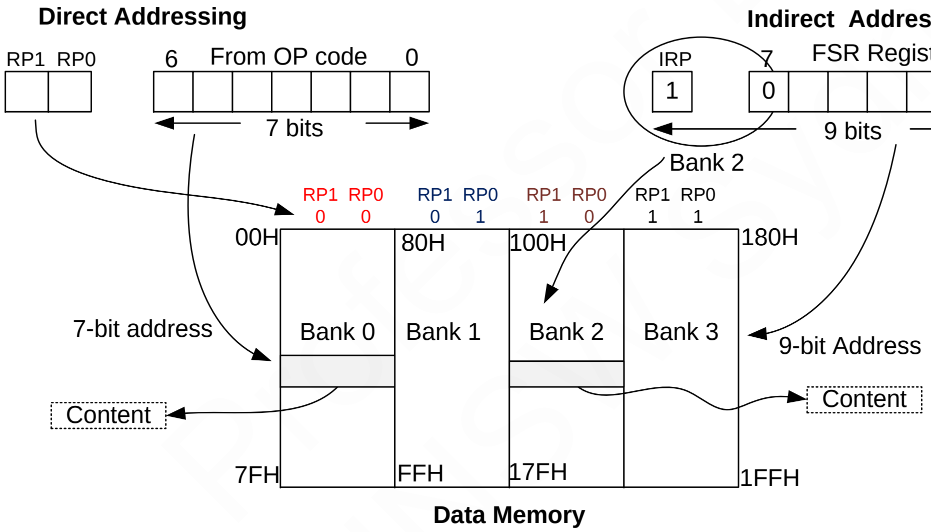


You must write the led_display subroutine for this program to work

Indirect Addressing of Data memory

- ✓ Indirect addressing is useful when creating a list of data that needs to be stored in data memory or, for example, clear RAM location 20H to 7FH
- ✓ Indirect addressing of data memory is possible by using the INDF register (See Table).
- ✓ An instruction using INDF register actually accesses the 8-bit data in the File Select Register (FSR) (See Table)and this data is used as the address for accessing the data memory. This is called the "Indirect Address". The difference between direct and indirect addressing is illustrated below:

00H	Indirect address **(INDF)**
01H	TMR0
02H	PCL
03H	STATUS
04H	**FSR**
05H	PORTA
06H	PORTB
07H	PORTC
:	
20H : : 7FH	General Purpose Registers (GPR) 96 Bytes
Bank 0 - Data Memory	



Direct and indirect addressing

An effective 9-bit address is obtained by concatenating the 8-bit FSR and the IRP bit of the STATUS register

bsf STATUS,7 ;IRP=1

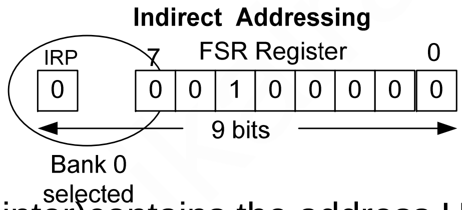
STATUS Register

Bit 7	Bit 6	Bit 5					Bit 0
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C

RP1	RP0	
0	0	Bank 0 selected (00H – 7FH)
0	1	Bank 1 selected (80H – FFH)
1	0	Bank 2 selected (100H – 17FH)
1	1	Bank 3 selected (180H – 1FFH)

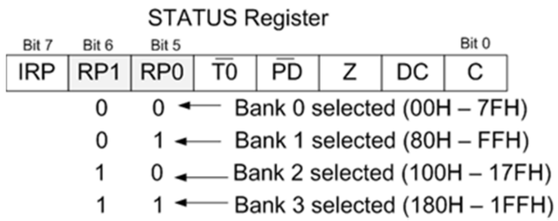
Example : Indirect addressing

➢ **This program stores the values D'01' to D'15' in memory locations H'20' to H'2F'**




```
          bcf       STATUS,7  ;IRP=0
          movlw     H'20'
          movwf     FSR       ; FSR register (Pointer)contains the address H'20'
          movlw     D'01'     ; load W register with 1
;Any instruction using the INDF register actually access the register pointed to by the
;File Select Register (FSR).
loop      movwf     INDF      ;Move W content  to memory location pointed by FSR
          addlw     1         ;increment the content of W register
          incf      FSR,1     ;increment the pointer
          btfss     FSR,4     ;when FSR = B'0010 1111'
          goto      loop      ;if bit 4 of FSR = 0 next instruction is executed
itself    goto      itself
          END
```

00H	Indirect address **(INDF)**
01H	TMR0
02H	PCL
03H	STATUS
04H	**FSR**
05H	PORTA
:	
20H : :	
2FH	
30H	
:	
Bank 0 - Data Memory	

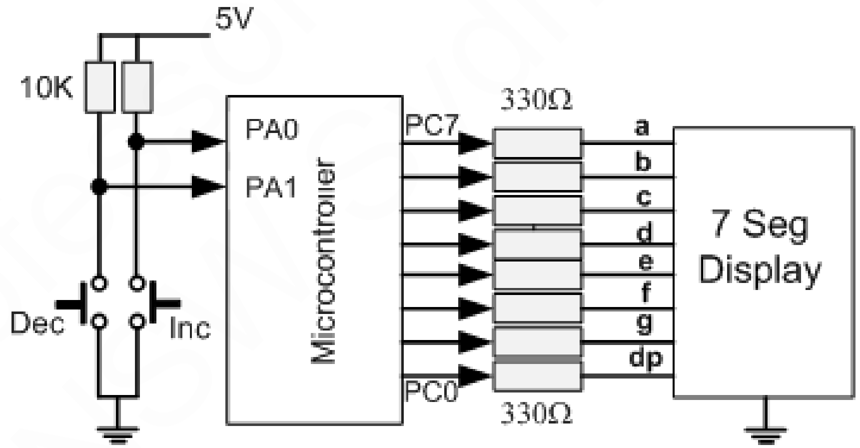


Exercise: Write a program to clear the memory locations from H'20' to H'7F' using indirect addressing

Laboratory Activities

Activity 5: Increment and decrement with two push buttons

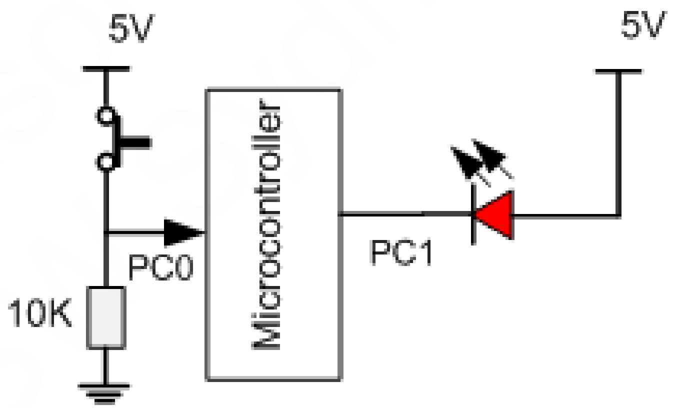
- Write an assembly language program to increment (by pressing and releasing the increment button) or decrement (by pressing and releasing the decrement button) the number displayed on the seven segment display.
- This number must be between zero and nine.
- The switches are connected to PA0 and PA1 and the 7-segment display is connected to PORT C (PC0 to PC7).
- The display must be initialised to show zero before executing the main program
- The switch debounce should be included in your program.
- Your program should include a Binary to BCD conversion table/routine



Laboratory Activities

Activity 6: Door Open or close Detector

- The switch connected to PC0 (PORT C) represents the closed or open condition of a door as shown in the diagram.
- The switch in a closed position is equivalent to the door being closed.
- Write an assembly program which continuously polls PC0 to detect if the door is open.
- When the door remains open, an LED connected to PC1 should flash at 0.5sec intervals continuously.
- You must write the delay routine for 0.5 sec assuming a clock frequency of 4MHz.



ELEC2117: References

1. Designing Embedded Systems with PIC Microcontrollers – Tim Wilmshurst, Elsevier, 2010
2. PIC Microcontrollers –Free online book – mikroElektronika ; http://www.mikroe.com/products/view/11/book-pic-microcontrollers/
3. PIC 16F886 Data Sheet (2007), Microchip Technology; www.microchip.com