EC-310 Microprocessor and Microcontroller Based Design

Chapter - 4

I/O Port Programming-Bit Addressing

Nazar Abbas Saqib

nazar.abbas@ceme.nust.edu.pk

Outline

1. I/O Port Programming in PIC18
2. I/O Bit Manipulation Programming

I/O Bit Manipulation Programming

- Earlier we studied I/O Port programming. Now we'll go through bit manipulation.
- Sometimes there is need to access only 1 or 2 bits of the port instead of entire 8 bits.

Table 4-9: Single-Bit Addressability of Ports for PIC18F458/4580

PORT	PORTB	PORTC	PORTD	PORTE	Port Bit
RA0	RB0	RC0	RD0	RE0	D0
RA1	RB1	RC1	RD1	RE1	D1
RA2	RB2	RC2	RD2	RE2	D2
RA3	RB3	RC3	RD3		D3
RA4	RB4	RC4	RD4		D4
RA5	RB5	RC5	RD5		D5
	RB6	RC6	RD6		D6
	RB7	RC7	RD7		D7

I/O Bit Manipulation Programming

- Following instructions are used for bit manipulation operations:

Table 4-8: Single-Bit (Bit-Oriented) Instructions for PIC18

Instruction	Function
BSF fileReg,bit	Bit Set fileReg (set the bit: bit = 1)
BCF fileReg,bit	Bit Clear fileReg (clear the bit: bit = 0)
BTG fileReg,bit	Bit Toggle fileReg (complement the bit)
BTFSC fileReg,bit	Bit test fileReg, skip if clear (skip next instruction if bit = 0)
BTFSS fileReg,bit	Bit test fileReg, skip if set (skip next instruction if bit = 1)

BSF (bit set fileReg)

- BSF is used to set HIGH a single bit of a given fileReg.
- Syntax of BSF is as follows:

```
BSF    fileReg,    bit_num
BSF    PORTB,      0        (sets 0th bit of port B)
BSF    PORTA,      5        (sets 5th bit of port A)
```

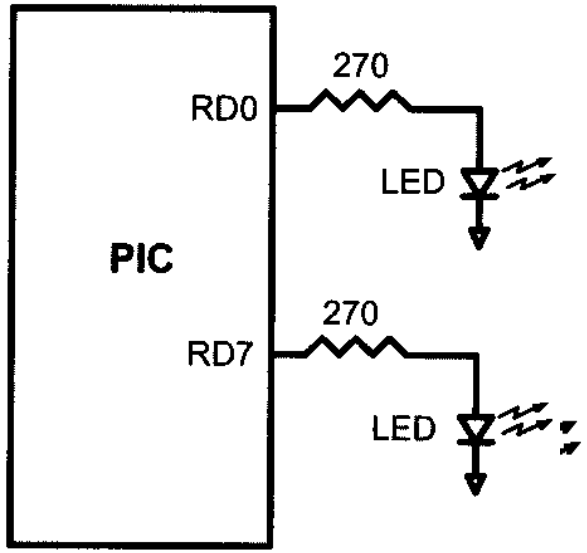
- BSF can be used to set any bit of any file register but mostly it is used for I/O port programming in embedded systems

Example 4-2

An LED is connected to each pin of Port D. Write a program to turn on each LED from pin D0 to pin D7. Call a delay module before turning on the next LED.

Solution:

```
CLRF   TRISD        ;make PORTD an output port
BSF    PORTD,0      ;bit set turns on RD0
CALL   DELAY        ;delay before next one
BSF    PORTD,1      ;turn on RD1
CALL   DELAY        ;delay before next one
BSF    PORTD,2
CALL   DELAY
BSF    PORTD,3
CALL   DELAY
BSF    PORTD,4
CALL   DELAY
BSF    PORTD,5
CALL   DELAY
BSF    PORTD,6
CALL   DELAY
BSF    PORTD,7
CALL   DELAY
```



BCF (bit clear fileReg)

- BCF is used to set LOW a single bit of a given fileReg.
- Syntax of BCF is as follows

BCF fileReg, bit_num

BCF PORTB, 0 **(clears 0th bit of port B)**

BCF PORTA, 5 **(clears 5th bit of port A)**

- BCF can be used to set any bit of any file register but mostly it is used for I/O port programming in embedded systems.

BCF (bit clear fileReg)

- **Example: The following code toggles pin RB2 continuously.**

```
      BCF   TRISB, 2          ;bit = 0, make RB2 an output pin
AGAIN BSF   PORTB, 2          ;bit set (RB2 = high)
      CALL  DELAY
      BCF   PORTB, 2          ;bit clear(RB2 = low)
      CALL  DELAY
      BRA   AGAIN
```

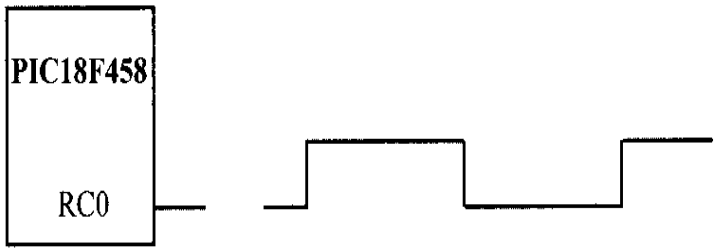
Example 4-3

Write the following programs:
(a) Create a square wave of 50% duty cycle on bit 0 of Port C.
(b) Create a square wave of 66% duty cycle on bit 3 of Port C.

Solution:

(a) The 50% duty cycle means that the "on" and "off" states (or the high and low portions of the pulse) have the same length. Therefore, we toggle RC0 with a time delay between each state.

```
        BCF    TRISC,0      ;clear TRIS bit for RC0 = out
HERE    BSF    PORTC,0      ;set to HIGH RC0 (RC0 = 1)
        CALL   DELAY        ;call the delay subroutine
        BCF    PORTC,0      ;RC0 = 0
        CALL   DELAY
        BRA    HERE         ;keep doing it
```



Another way to write the above program is:

```
        BCF    TRISC,0      ;make RC0 = out
HERE    BTG    PORTC,0      ;complement bit 0 of PORTC
        CALL   DELAY        ;call the delay subroutine
        BRA    HERE         ;keep doing it
```

(b) A 66% duty cycle means that the "on" state is twice the "off" state.

```
      BCF   TRISC,3     ;clear TRISC3 bit for output
BACK  BSF   PORTC,3     ;RC3 = 1
      CALL  DELAY       ;call the delay subroutine
      CALL  DELAY       ;twice for 66%
      BCF   PORTC,3     ;RC3 = 0
      CALL  DELAY       ;call delay once for 33%
      BRA   BACK        ;keep doing it
```

BTG (bit toggle fileReg)

- BTG is used to toggle a single bit of a given fileReg.
- Syntax of BTG is as follows
- ***BTG fileReg, bit_num***
  ***BTG PORTB, 0* (toggles 0th bit of port B)**
  ***BTG PORTA, 5* (toggles 5th bit of port A)**
- BTG toggles the specified bit of specified port.

```
        BCF   TRISB, 2          ;make RB2 an output pin
BACK    BTG   PORTB, 2          ;toggle pin RB2 only
        CALL  DELAY
        BRA   BACK
```

Checking an input pin

- To make decisions based on the status of a given bit in the file register we make use of instructions
  - BTFSC (bit test fileReg skip if clear)
  - BTFSS (bit test fileReg skip if set)
- BTFSC and BTFSS can be used to monitor the status of any bit of any fileReg or SFR.
- To monitor the status of the single bit for HIGH, we use BTFSS instruction.
- This instruction tests the bit and skips the next instruction if it is HIGH.

BTFSS (bit test fileReg skip if set)

Example 4-4

Write a program to perform the following:
(a) Keep monitoring the RB2 bit until it becomes HIGH;
(b) When RB2 becomes HIGH, write value 45H to Port C, and also send a HIGH-to-LOW pulse to RD3.

Solution:

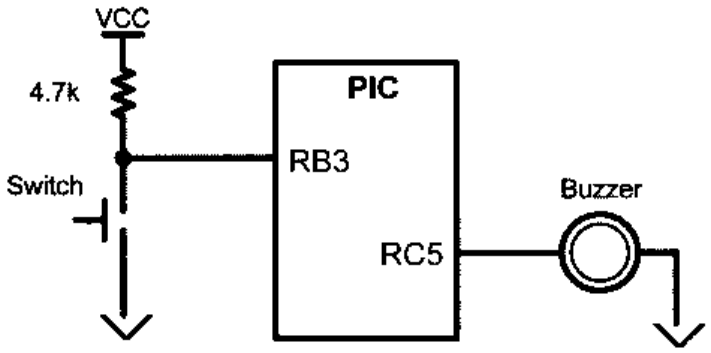
```
        BSF    TRISB,2      ;make RB2 an input
        CLRF   TRISC        ;make PORTC an output port
        BCF    PORTD,3      ;make RD3 an output
        MOVLW  0x45         ;WREG = 45h
AGAIN   BTFSS  PORTB,2      ;bit test RB2 for HIGH
        BRA    AGAIN        ;keep checking if LOW
        MOVWF  PORTC        ;issue WREG to Port C
        BSF    PORTD,3      ;bit set fileReg RD3 (H-to-L)
        BCF    PORTD,3      ;bit clear fileReg RD3 (L)
```

In this program, instruction "BTFSS PORTB,2" stays in the loop as long as RB2 is LOW. When RB2 becomes HIGH, it skips the branch instruction to get out of the loop, and writes the value 45H to Port C. It also sends a HIGH-to-LOW pulse to RD3.

BTFSC (bit test fileReg skip if clear)

Example 4-5

Assume that bit RB3 is an input and represents the condition of a door alarm. If it goes LOW, it means that the door is open. Monitor the bit continuously. Whenever it goes LOW, send a HIGH-to-LOW pulse to port RC5 to turn on a buzzer.



Solution:

```
        BSF   TRISB,3      ;make RB3 an input
        BCF   TRISC,5      ;make RC5 an output
HERE    BTFSC PORTB,3      ;keep monitoring RB3 for HIGH
        BRA   HERE         ;stay in the loop
        BSF   PORTC,5      ;make RC5 HIGH
        BCF   PORTC,5      ;make RC5 LOW for H-to-L
        BRA   HERE
```

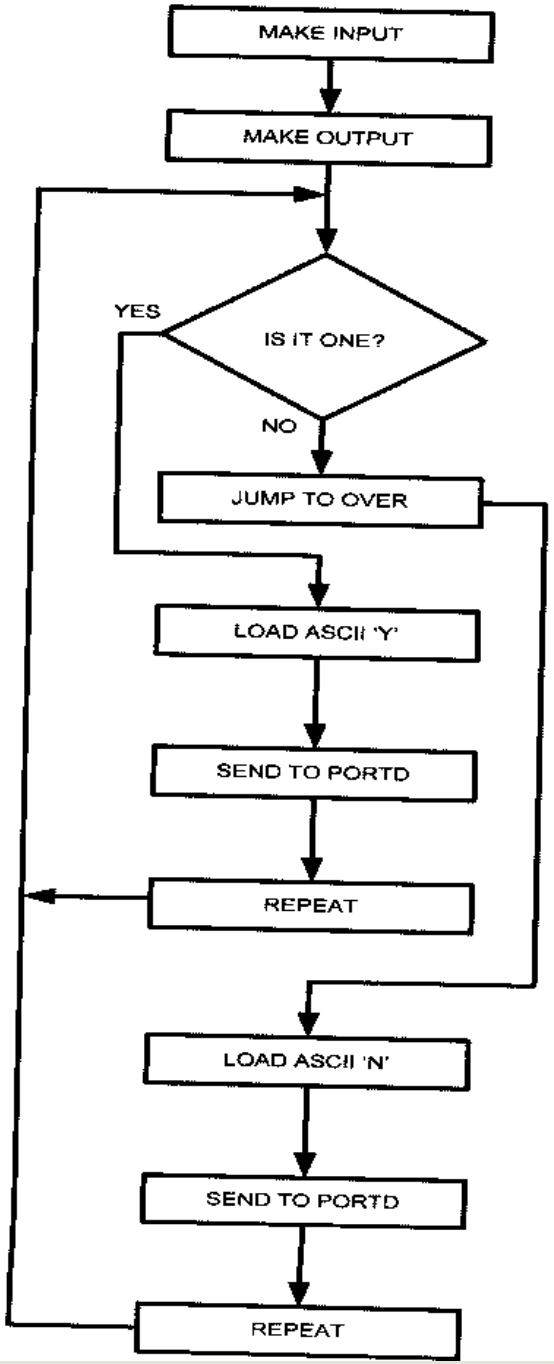
Monitoring a single bit

Example 4-6

A switch is connected to pin RB2. Write a program to check the status of SW and perform the following:
(a) If SW = 0, send the letter 'N' to PORTD.
(b) If SW = 1, send the letter 'Y' to PORTD.

Solution:

```
        BSF    TRISB,2      ;make RB2 an input
        CLRF   TRISD        ;make PORTD an output port
AGAIN   BTFSS  PORTB,2      ;bit test RB2 for HIGH
        BRA    OVER         ;it must be LOW
        MOVLW  A'Y'         ;WREG = 'Y' ASCII letter Y
        MOVWF  PORTD        ;issue WREG to PORTD
        GOTO   AGAIN        ;we can use BRA too
OVER    MOVLW  A'N'         ;WREG = 'N' ASCII  letter N
        MOVWF  PORTD        ;issue WREG to PORTD
        GOTO   AGAIN        ;we can use BRA too
```



INSTRUCTIONS

	BSF TRISB, 2
	CLRF TRISD
AGAIN	BTFSS PORTB, 2
	BRA OVER
	MOVLW A'Y'
	MOVWF PORTD
	GOTO AGAIN
OVER	MOVLW A'N'
	MOVWF PORTD
	GOTO AGAIN

Monitoring a single bit

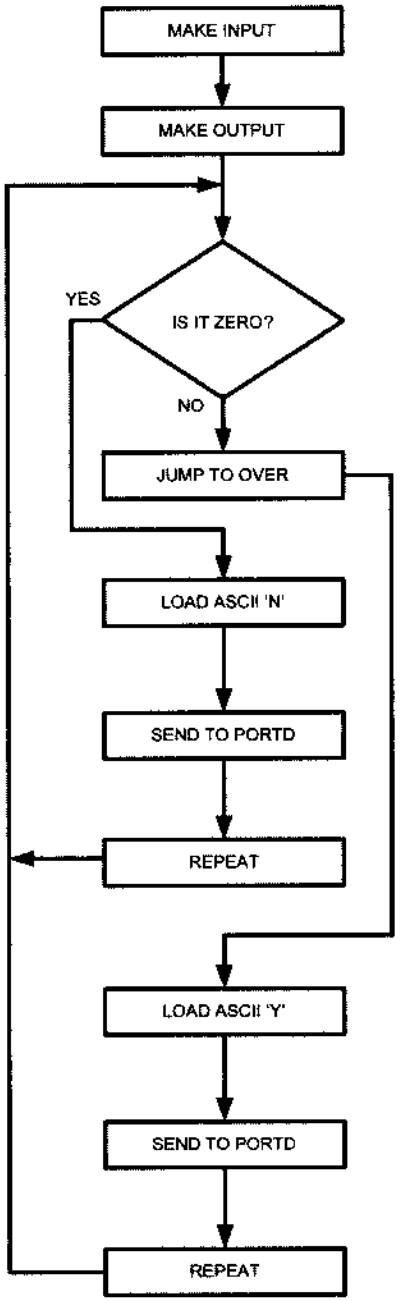
Example 4-7

A switch is connected to pin RB2. Write a program to check the status of SW and perform the following:
(a) If SW = 0, send letter 'N' to PORTD.
(b) If SW = 1, send letter 'Y' to PORTD.

Use the BTFSC instruction to check the SW status. This is another version of Example 4-6 using the BTFSC instruction instead of BTFSS.

Solution:

```
       BSF   TRISB,2      ;make RB2 an input
       CLRF  TRISD        ;make PORTD an output port
AGAIN  BTFSC PORTB,2      ;bit test RB2 for LOW
       BRA   OVER         ;it must be HIGH
       MOVLW A'N'         ;WREG = 'N' ASCII letter N
       MOVWF PORTD        ;issue WREG to PORTD
       BRA   AGAIN        ;we can use GOTO
OVER   MOVLW A'Y'         ;WREG = 'Y' ASCII letter Y
       MOVWF PORTD        ;issue WREG to PORTD
       BRA   AGAIN        ;we can use GOTO
```



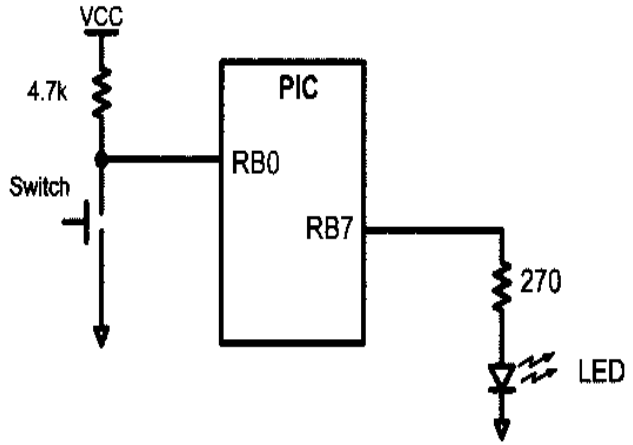
INSTRUCTIONS

	BSF TRISB, 2
	CLRF TRISD
AGAIN	BTFSC PORTB, 2
	BRA OVER
	MOVLW A'N'
	MOVWF PORTD
	BRA AGAIN
OVER	MOVLW A'Y'
	MOVWF PORTD
	BRA AGAIN

Reading a single bit

Example 4-8

A switch is connected to pin RB0 and an LED to pin RB7. Write a program to get the status of SW and send it to the LED.



Solution:

```
       BSF   TRISB,0      ;make RB0 an input
       BCF   TRISB,7      ;make RB7 an output
AGAIN  BTFSS PORTB,0      ;bit test RB0 for HIGH
       GOTO  OVER         ;it must be LOW (BRA is OK too)
       BSF   PORTB,7
       GOTO  AGAIN        ;we can use BRA too
OVER   BCF   PORTB,7
       GOTO  AGAIN        ;we can use BRA too
```

Reading a single bit

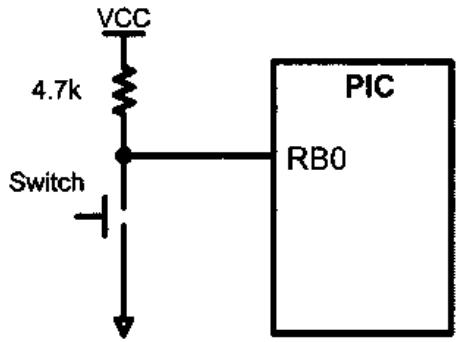
Example 4-9

A switch is connected to pin RB0. Write a program to get the status of SW and save it in D0 of fileReg location 0x20.

Solution:

```
MYBITREG EQU 0x20 ;set aside loc 0x20 reg

       BSF   TRISB,0      ;make RB0 an input
AGAIN  BTFSS PORTB,0      ;bit test RB0 for HIGH
       GOTO  OVER         ;it must be LOW (BRA is OK too)
       BSF   MYBITREG,0   ;set bit 0 of fileReg
       GOTO  AGAIN        ;we can use BRA too
OVER   BCF   MYBITREG,0   ;clear bit 0 of fileReg
       GOTO  AGAIN        ;we can use BRA too
```



Reading input pins vs LATx port

- There are two possibilities in reading a port:
  - Read the status of the input port.
  - Read the internal latch of the LAT register.
- Reading LATx for ports:
- For example, consider instruction '*COMF PORTB*'.
- The sequence of actions when such an instruction is executed is as follows:
  - The instruction reads the internal latch of the LATB and brings that data into the CPU.
  - The data is complemented.
  - The result is rewritten back to LATB latch.
  - The data on the pins are changed only if the TRISB bits are cleared to zero.

Reading input pins vs LATx port

- The LATx register associated with an I/O pin eliminates the problems that could occur with read-modify-write instructions.
- A read of the LATx register returns the values held in the port output latches, instead of the values on the I/O pins. A write to the LATx register has the same effect as a write to the PORTx register.
- The differences between the PORT and LAT registers can be summarized as follows:
  - A write to the PORTx register writes the data value to the port latch.
  - A write to the LATx register writes the data value to the port latch.
  - A read of the PORTx register reads the data value on the I/O pin.
  - A read of the LATx register reads the data value held in the port latch.