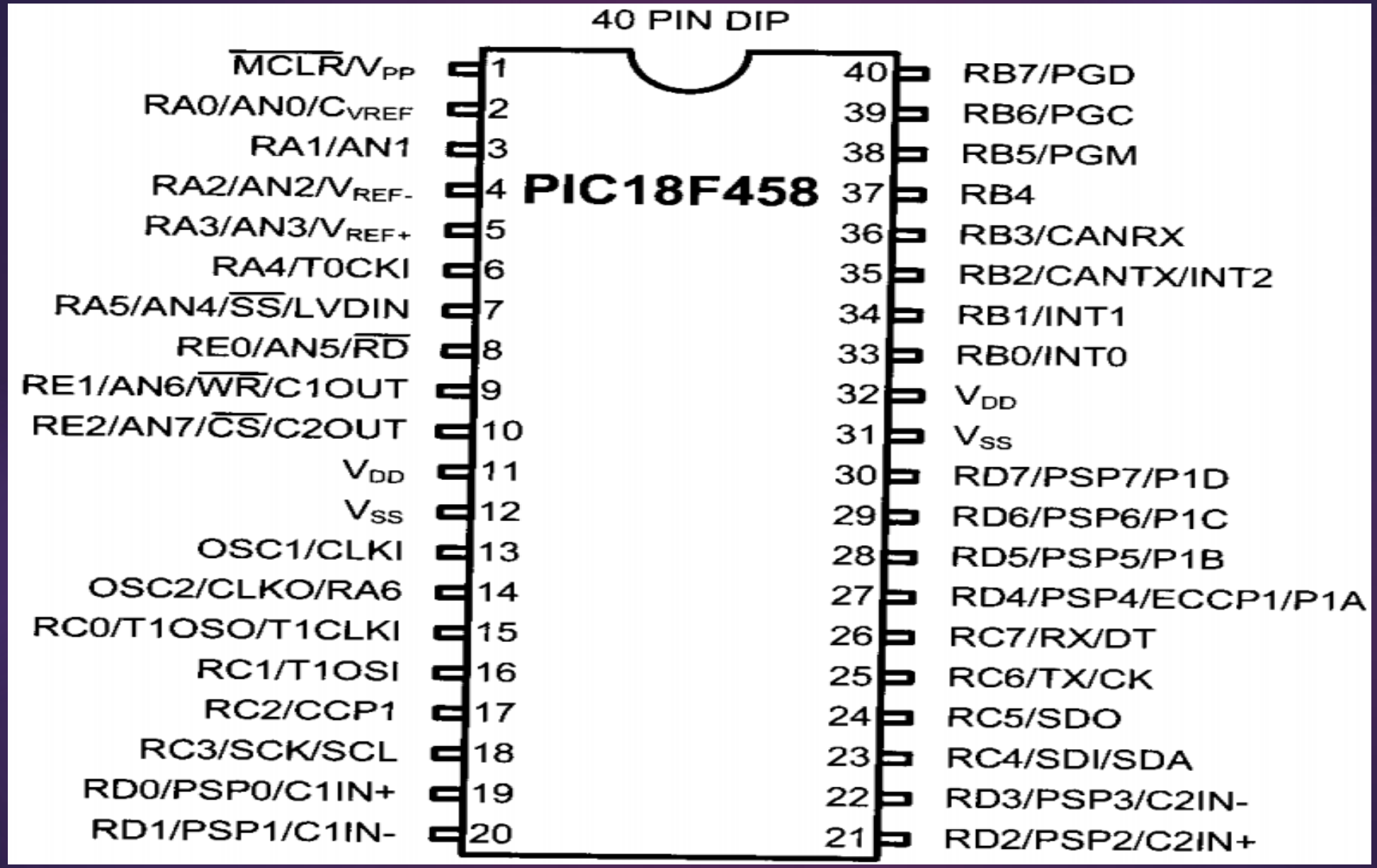
Microprocessor Interfacing & Programming

LECTURE 13 &14

I/O Port Programming in PIC18

- PIC18F458 is a 40-pin chip.
- Total 33 pins are set aside for the five ports. PORTA, PORTB, PORTC, PORTD and PORTE.
- Several dedicated pins for power, clock, reset, and special features.
- To use any of the above ports as an input or output port, it must be programmed.
- In addition to being used for simple I/O, each port has some other functions such as ADC, timers, interrupts, and serial communication pins.


40 PIN DIP
PIC18F458
MCLR/VPP 1
RA0/AN0/CVREF 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS/LVDIN 7
RE0/AN5/RD 8
RE1/AN6/WR/C1OUT 9
RE2/AN7/CS/C2OUT 10
VDD 11
VSS 12
OSC1/CLKI 13
OSC2/CLKO/RA6 14
RC0/T1OSO/T1CLKI 15
RC1/T1OSI 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0/C1IN+ 19
RD1/PSP1/C1IN- 20
40 RB7/PGD
39 RB6/PGC
38 RB5/PGM
37 RB4
36 RB3/CANRX
35 RB2/CANTX/INT2
34 RB1/INT1
33 RB0/INT0
32 VDD
31 VSS
30 RD7/PSP7/P1D
29 RD6/PSP6/P1C
28 RD5/PSP5/P1B
27 RD4/PSP4/ECCP1/P1A
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3/C2IN-
21 RD2/PSP2/C2IN+

Pins	18-pin	28-pin	40-pin	64-pin	80-pin
Chip	**PIC18F1220**	**PIC18F2220**	**PIC18F458**	**PIC18F6525**	**PIC18F8525**
Port A	X	X	X	X	X
Port B	X	X	X	X	X
Port C		X	X	X	X
Port D			X	X	X
Port E			X	X	X
Port F				X	X
Port G				X	X
Port H				X	X
Port J				X	X
Port K					X
Port L					X

Note: X indicates that the port is available.

- Each port has **three** SFRs associated with it.

PORTx , LATx and TRISx

- Like for Port B we have, PORTB,

TRISB and LATB.

TRIS ---------- TRIState

LAT ---------- LATch

Table 4-2: Ports' SFR Addresses for PIC18F458

Port	Address
PORTA	F80H
PORTB	F81H
PORTC	F82H
PORTD	F83H
PORTE	F84H
LATA	F89H
LATB	F8AH
LATC	F8BH
LATD	F8CH
LATE	F8DH
TRISA	F92H
TRISB	F93H
TRISC	F94H
TRISD	F95H
TRISE	F96H

F80h	PORTA	FA0h	PIE2	FC0h	----	FE0h	BSR
F81h	PORTB	FA1h	PIR2	FC1h	ADCON1	FE1h	FSR1L
F82h	PORTC	FA2h	IPR2	FC2h	ADCON0	FE2h	FSR1H
F83h	PORTD	FA3h	----	FC3h	ADRESL	FE3h	PLUSW1 *
F84h	PORTE	FA4h	----	FC4h	ADRESH	FE4h	PREINC1 *
F85h	----	FA5h	----	FC5h	SSPCON2	FE5h	POSTDEC1 *
F86h	----	FA6h	----	FC6h	SSPCON1	FE6h	POSTINC1 *
F87h	----	FA7h	----	FC7h	SSPSTAT	FE7h	INDF1 *
F88h	----	FA8h	----	FC8h	SSPADD	FE8h	WREG
F89h	LATA	FA9h	----	FC9h	SSPBUF	FE9h	FSR0L
F8Ah	LATB	FAAh	----	FCAh	T2CON	FEAh	FSR0H
F8Bh	LATC	FABh	RCSTA	FCBh	PR2	FEBh	PLUSW0 *
F8Ch	LATD	FACh	TXSTA	FCCh	TMR2	FECh	PREINC0 *
F8Dh	LATE	FADh	TXREG	FCDh	T1CON	FEDh	POSTDEC0 *
F8Eh	----	FAEh	RCREG	FCEh	TMR1L	FEEh	POSTINC0 *
F8Fh	----	FAFh	SPBRG	FCFh	TMR1H	FEFh	INDF0 *
F90h	----	FB0h	----	FD0h	RCON	FF0h	INTCON3
F91h	----	FB1h	T3CON	FD1h	WDTCON	FF1h	INTCON2
F92h	TRISA	FB2h	TMR3L	FD2h	LVDCON	FF2h	INTCON
F93h	TRISB	FB3h	TMR3H	FD3h	OSCCON	FF3h	PRODL
F94h	TRISC	FB4h	----	FD4h	----	FF4h	PRODH
F95h	TRISD	FB5h	----	FD5h	T0CON	FF5h	TABLAT
F96h	TRISE	FB6h	----	FD6h	TMR0L	FF6h	TBLPTRL
F97h	----	FB7h	----	FD7h	TMR0H	FF7h	TBLPTRH
F98h	----	FB8h	----	FD8h	STATUS	FF8h	TBLPTRU
F99h	----	FB9h	----	FD9h	FSR2L	FF9h	PCL
F9Ah	----	FBAh	CCP2CON	FDAh	FSR2H	FFAh	PCLATH
F9Bh	----	FBBh	CCPR2L	FDBh	PLUSW2 *	FFBh	PCLATU
F9Ch	----	FBCh	CCPR2H	FDCh	PREINC2 *	FFCh	STKPTR
F9Dh	PIE1	FBDh	CCP1CON	FDDh	POSTDEC2 *	FFDh	TOSL
F9Eh	PIR1	FBEh	CCPR1L	FDEh	POSTINC2 *	FFEh	TOSH
F9Fh	IPR1	FBFh	CCPR1H	FDFh	INDF2 *	FFFh	TOSU

TRISx register role

- TRISx SFR dictates direction of pin --------- Input or Output.
- To make a port an output, we write 0s to the TRISx register.
- To output data to any of the pins of the Port B, we first put 0s into the TRISB register to make it an output port, and then send the data to the Port B SFR itself.

Example

```
        MOVLW 0x0          ;WREG = 00
        MOVWF TRISB        ;make Port B an output port 0000 0000
L1      MOVLW 0x55         ;WREG = 55h
        MOVWF PORTB        ;put 55h on port B pins
        CALL  DELAY
        MOVLW 0xAA         ;WREG = AAh
        MOVWF PORTB        ;put AAh on port B pins
        CALL  DELAY
        GOTO  L1
```

TRIS register for inputting data

Get the data at the pins of Port C and send it infinitely to Port B after adding the value 5 in it.

```
        MOVLW B'00000000' ;WREG = 0000000 (binary)
        MOVWF TRISB       ;Port B an output port(0 for O)
        MOVLW B'11111111' ;WREG = 11111111 (binary)
        MOVWF TRISC       ;Port C an input port (1 for I)
L2      MOVF  PORTC,W     ;move data from Port C to WREG
        ADDLW 5           ;add some value to it
        MOVWF PORTB       ;send it to Port B
        GOTO  L2          ;continue forever
```

Efficient approach

```
        CLRF  TRISB       ;clear TRISB (Port B an output port)
        SETF  TRISC       ;set TRISC (Port C an input port)
L2      MOVF  PORTC,W           ;get data from port C
        ADDLW 5           ;add some value
        MOVWF PORTB       ;send it to port B
        BRA   L2
```

Port A

```
        ;toggle all bits of PORTA

        MOVLW B'00000000' ;WREG = 00000000 (binary)
        MOVWF TRISA       ;make Port A an output port (0 for Out)
L1      MOVLW 0x55        ;WREG = 55h
        MOVWF PORTA       ;put 55h on Port A pins
        CALL  DELAY
        MOVLW 0xAA        ;WREG = AAh
        MOVWF PORTA       ;put AAh on Port A pins
        CALL  DELAY
        GOTO  L1
```

It must be noted that 55H (01010101) when complemented becomes AAH (10101010). Although by sending 55H and AAH to Port A continuously, we toggle all 8 bits of the Port A register, only 6 pins (RA0–RA5) will show the toggling data.

Port A as input

```
MYREG EQU   0X20  ;save it here

MOVLW B'11111111' ;WREG = 11111111 (binary)
MOVWF TRISA       ;make Port A an input port (1 for In)
MOVF  PORTA,W     ;move from fileReg of Port A to WREG
MOVWF MYREG       ;save it in fileReg of MYREG
```

Dual role of Ports A and B

Table 4-3: Port A Alternate Functions

Bit	Function
RA0	AN0/CVREF
RA1	AN1
RA2	AN2/VREF-
RA3	AN3/VREF+
RA4	T0CKI
RA5	AN4/SS/LVDIN
RA6	OSC2/CLKO

Table 4-4: Port B Alternate Functions

Bit	Function
RB0	INT0
RB1	INT1
RB2	INT2/CANTX
RB3	CANRX
RB4	
RB5	PGM
RB6	PGC
RB7	PGD

Dual role of Ports C and D

Table 4-5: Port C Alternate Functions

Bit	Function
RC0	T1OSO/T1CKI
RC1	T1OSI
RC2	CCP1
RC3	SCK/SCL
RC4	SDI/SDA
RC5	SDO
RC6	TX/CK
RC7	RX/DT

Table 4-6: Port D Alternate Functions

Bit	Function
RD0	PSP0/C1IN+
RD1	PSP1/C1IN-
RD2	PSP2/C2IN+
RD3	PSP3/C2IN-
RD4	PSP4/ECCP1/P1A
RD5	PSP5/P1B
RD6	PSP6/P1C
RD7	PSP7/P1D

Different ways of accessing the entire 8 bits

```
        ;toggle all bits of PORTB

        MOVLW 0x0           ;WREG = 00
        MOVWF TRISB         ;make Port B an output port
L1      MOVLW 0x55          ;WREG = 55h
        MOVWF PORTB         ;put 55h on Port B pins
        CALL  DELAY
        MOVLW 0xAA          ;WREG = AAh
        MOVWF PORTB         ;put AAh on Port B pins
        CALL  DELAY
        GOTO  L1
```

```
        CLRF  TRISB         ;make Port B an output port
L1      MOVLW 0x55          ;WREG = 55h
        MOVWF PORTB         ;put 55h on Port B pins
        CALL  DELAY
        MOVLW 0xAA          ;WREG = AAh
        MOVWF PORTB         ;put AAh on Port B pins
        CALL  DELAY
        GOTO  L1
```

read-modify-write technique →

```
        CLRF  TRISB         ;make Port B an output port
        MOVLW 0x55          ;WREG = 55h
        MOVWF PORTB         ;put 55h on Port B pins
L2      COMF  PORTB, F      ;toggle bits of Port B
        CALL  DELAY
        BRA   L2
```

Data Dependency in CPU Design

- Due to the timing issue, we must be careful not to have two I/O operations one right after the other.

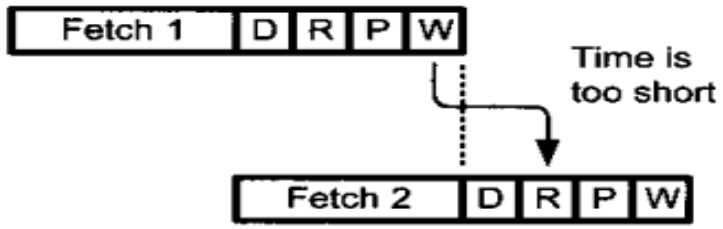
```
      CLRF  TRISB ;clear TRISB to make PORTB an output port
      SETF  TRISC       ;set TRISC all 1s (Port C as Input)
L4    MOVF  PORTC,W     ;get data from Port C into WREG
      NOP               ;NEED some NOP to ensure data is in WREG
      MOVWF PORTB       ;before it is sent to Port B
      BRA   L4          ;keep doing it
```

NOTE: We need a NOP (or some other instruction) to make sure that the data is written into WREG before it is read for outputting to Port B.

Solution of the problem

- This type of data dependency is refered as RAW (Read-After-Write).
- The NOP will introduce a bubble into the pipeline to remove data dependency due to RAW.
- One way to avoid this problem is to use the MOVFF instruction.

```
      CLRF  TRISB       ;make Port B an output port
      SETF  TRISC       ;TRISC = FFh (Port C Input)
L5    MOVFF PORTC,PORTB ;get from Port C and send to PORTB
      BRA   L5          ;keep doing it
```

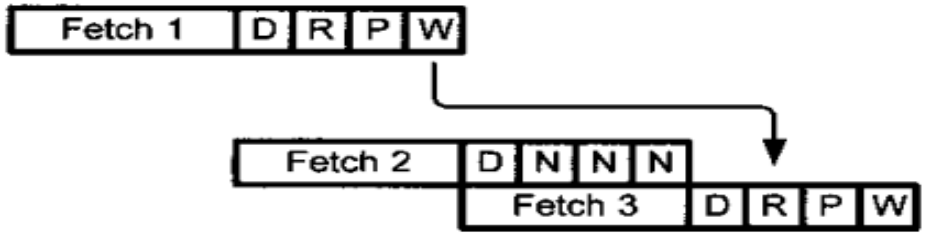


INSTRUCTION

MOVF PORTC,W ;Read PORTC into WREG

MOVWF PORTB ;Write WREG to PORTB

The RAW (Read – After – Write) for two consecutive instructions .



INSTRUCTION

MOVF PORTC,W

NOP ;Bubble in Pipeline

MOVWF PORTB

N = No Operation

D = Decode the instruction

R = Read the operand

P = Process

W = Write the result to destination register

Ports Status upon Reset

Register	Reset Value (Binary)
TRISA	11111111
TRISB	11111111
TRISC	11111111
TRISD	11111111

Note: All ports are input ports upon reset.

Write a test program for the PIC18 chip to toggle all the bits of PORTB, PORTC, and PORTD every 1/4 of a second. Assume a crystal frequency of 4 MHz.

Solution:

```
;tested with MPLAB for the PIC18F458 and XTAL = 4 MHz

        list P=PIC18F458
#include P18F458.INC

R1  equ  0x07
R2  equ  0x08

ORG 0
        CLRF   TRISB          ;make Port B an output port
        CLRF   TRISC          ;make Port C an output port
        CLRF   TRISD          ;make Port D an output port
        MOVLW  0x55           ;WREG = 55h
        MOVWF  PORTB          ;put 55h on Port B pins
        MOVWF  PORTC          ;put 55h on Port C pins
        MOVWF  PORTD          ;put 55h on Port D pins
L3      COMF   PORTB,F        ;toggle bits of Port B
        COMF   PORTC,F        ;toggle bits of Port C
        COMF   PORTD,F        ;toggle bits of Port D
        CALL   QDELAY         ;quarter of a second delay
        BRA    L3


;-------------1/4 SECOND DELAY
QDELAY
        MOVLW  D'200'
        MOVWF  R1
D1      MOVLW  D'250'
        MOVWF  R2
D2      NOP
        NOP
        DECF  R2,  F
        BNZ  D2
        DECF  R1,  F
        BNZ  D1
        RETURN
        END
```

I/O Bit Manipulation programming

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)

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

BSF (bit set fileReg) & BCF (bit clear fileReg)

- Syntax: BSF fileReg, bit_num
- Syntax: BCF fileReg, bit_num

where, fileReg can be any location in file register and bit_num is the desired bit number from 0 to 7.

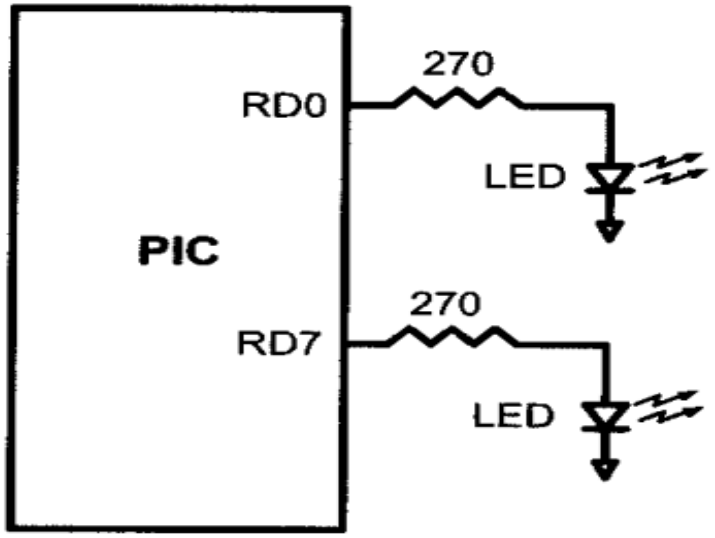
Remember: For I/O ports, we must activate the appropriate bit in the TRISx register if we want the pin to reflect the changes.

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
```

Ex:4-2



BCF (bit clear fileReg)

```
      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
```

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.

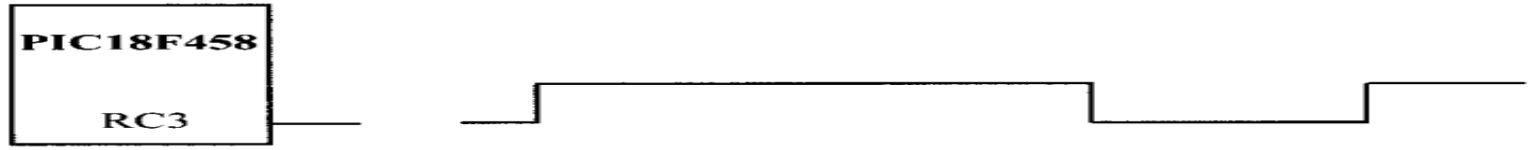
```
        BCF     TRISC,0         ;clear TRIS bit for RC0 = out
HERE    BSF     PORTC,0         ;set to HIGH RC0 (RC0 = 1)
        CALL    DELAY           ;call the delay subroutine
        BCF     PORTC,0         ;RC0 = 0
        CALL    DELAY
        BRA     HERE            ;keep doing it
```

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)

- Syntax: BTG fileReg , bit_number

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

Notice: RB2 is the third bit of Port B (the first bit is RB0, the second bit is RB1)