

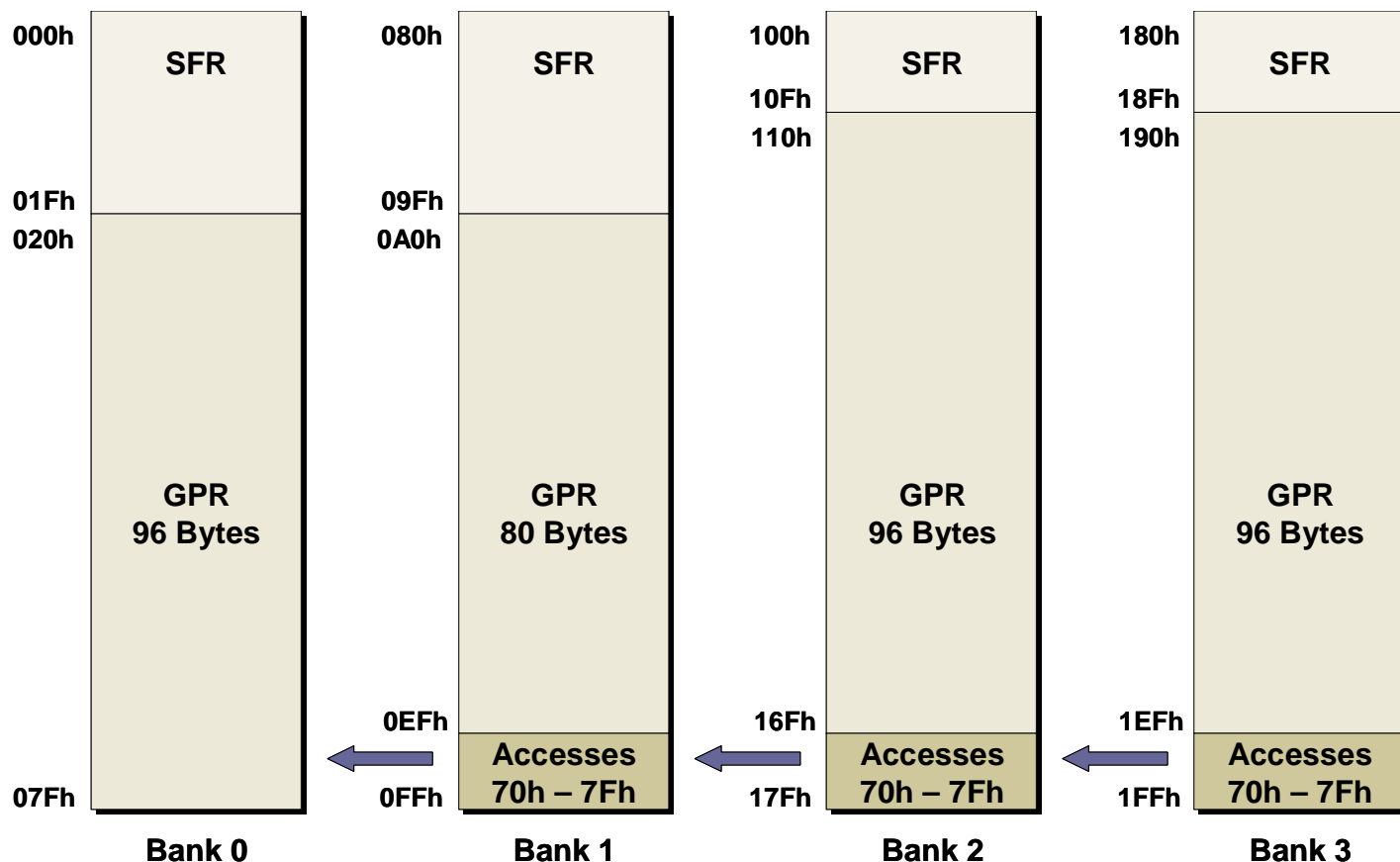


# 1003 GS3

Introduction to PIC16  
Architecture, Instruction Set and  
Assembly Language Programming

Notes for Hands-on Exercises

Mnemonic, Operands	Description	Cycles	14-bit Opcode		Status Affected	Notes	
			MSb	LSb			
<b>Byte-Oriented File Register Operations</b>							
ADDWF	f,d	Add W and f	1	00 0111	dfff ffff	C,DC,Z	1,2
ANDWF	f,d	AND W and f	1	00 0101	dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001	1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001	0xxx xxxx	Z	
COMF	f,d	Complement f	1	00 1001	dfff ffff	Z	1,2
DECF	f,d	Decrement f	1	00 0011	dfff ffff	Z	1,2
DECFSZ	f,d	Decrement f, Skip if 0	1 (2)	00 1011	dfff ffff		1,2,3
INCF	f,d	Increment f	1	00 1010	dfff ffff	Z	1,2
INCFSZ	f,d	Increment f, Skip if 0	1 (2)	00 1111	dfff ffff		1,2,3
IORWF	f,d	Inclusive OR W with f	1	00 0100	dfff ffff	Z	1,2
MOVF	f,d	Move f	1	00 1000	dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000	1fff ffff		
NOP		No Operation	1	00 0000	0xx0 0000		
RLF	f,d	Rotate Left f through Carry	1	00 1101	dfff ffff	C	1,2
RRF	f,d	Rotate Right f through Carry	1	00 1100	dfff ffff	C	1,2
SUBWF	f,d	Subtract W from f	1	00 0010	dfff ffff	C,DC,Z	1,2
SWAPF	f,d	Swap nibbles in f	1	00 0010	dfff ffff		1,2
XORWF	f,d	Exclusive OR W with f	1	00 0110	dfff ffff	Z	1,2
<b>Bit-Oriented File Register Operations</b>							
BCF	f,b	Bit Clear f	1	01 00bb	bfff ffff		1,2
BSF	f,b	Bit Set f	1	01 01bb	bfff ffff		1,2
BTFSC	f,b	Bit Test f, Skip if Clear	1 (2)	01 10bb	bfff ffff		3
BTFSS	f,b	Bit Test f, Skip if Set	1 (2)	01 11bb	bfff ffff		3
<b>Literal and Control Operations</b>							
ADDLW	k	Add literal to W	1	11 111x	kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001	kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk	kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000	0110 0100	$\overline{TO}, \overline{PD}$	
GOTO	k	Go to address	2	10 1kkk	kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000	kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx	kkkk kkkk		
RETFIE		Return from Interrupt	2	00 0000	0000 1001		
RETLW	k	Return with literal in W	2	11 01xx	kkkk kkkk		
RETURN		Return from Subroutine	2	00 0000	0000 1000		
SLEEP	-	Go into Standby mode	1	00 0000	0110 0011	$\overline{TO}, \overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x	kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010	kkkk kkkk	Z	
Note:	1: When an I/O register is modified as a function of itself (e.g. MOVF PORTB, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back as a '0'. 2: If this instruction is executed on the TMR0 register (and, where applicable, d=1), the prescaler will be cleared if assigned to the Timer0 module. 3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.						



Note: Your program may use any RAM in the General Purpose RAM (GPR) sections for its own purposes. RAM in the Special Function Register (SFR) sections contains the configuration and control registers used to work with the processor and its peripherals.

By default, you will be pointing to bank 0 so that all RAM accesses will occur in this bank. You may use any RAM above address 0x20 for your own needs.

To switch banks, bits RP0 and RP1 in the STATUS register must be set or cleared as appropriate. Note that the STATUS register is available in all four banks.

Example:

```
bsf    STATUS,RP0    ; Switch to bank 1 when already in bank 0
```

A special feature of some PIC16 devices is the section of “unbanked” memory from 0x6F to 0x7F. This area of RAM in bank 0 is accessible from all the upper banks by reading or writing the last 16 locations in each bank (F0-FF, 170-17F and 1F0-1FF). Be careful when using this if you migrate your code to other PIC16 devices since they don't all have this feature in their memory map.

## STATUS Register (Address 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
<b>IRP</b>	<b>RP1</b>	<b>RP0</b>	<b><math>\overline{\text{TO}}</math></b>	<b><math>\overline{\text{PD}}</math></b>	<b>Z</b>	<b>DC</b>	<b>C</b>
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)

1 = Bank 2 and 3 (100h – 1FFh)

0 = Bank 0 and 1 (00h – FFh)

bit 6-5 **RP1:RP0:** Register Bank Select bits (used for direct addressing)

11 = Bank 3 (180h-1FFh)

10 = Bank 2 (100h-17Fh)

01 = Bank 1 (080h-0FFh)

00 = Bank 0 (000h-07Fh)

Hint: To switch banks, use something like the following:

```
bsf STATUS, RP0
```

Since the bank bits default to 0, it is not necessary to set or clear RP1 unless you want to access bank 2 or 3.

bit 4  **$\overline{\text{TO}}$ :** Time-out bit

1 = After power-up, **CLRWDT** instruction, or **SLEEP** instruction

0 = A WDT time-out occurred

bit 3  **$\overline{\text{PD}}$ :** Power-down bit

1 = After power-up or by the **CLRWDT** instruction

0 = By execution of the **SLEEP** instruction

bit 2 **Z:** Zero bit

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/Borrow bit

1 = A carry out from the 4<sup>th</sup> low order bit of the result occurred

0 = No carry out from the 4<sup>th</sup> low order bit of the result occurred

bit 0 **C:** Carry/Borrow bit

1 = A carry out from the most significant bit of the result occurred

0 = No carry out from the most significant bit of the result occurred

### Legend:

R = Readable Bit

W = Writable Bit

U = Unimplemented bit (Read as '0')

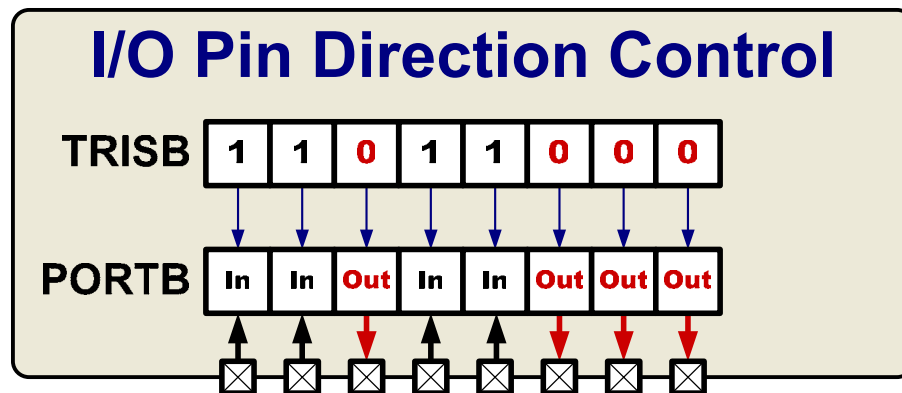
- n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

'x' = Bit is unknown

## How TRISB relates to PORTB



- To make bit x of PORTB an input, set bit x in TRISB.
- To make bit x of PORTB an output, clear bit x in TRISB.

Remember: TRISB is in Bank 1 and PORTB is in BANK0, so bank switching will be required to modify TRISB.

Example initialization of PORTB and TRISB:

```

clrf    PORTB           ; Ensure output latches are clear
bsf     STATUS, RP0    ; Switch to bank 1
movlw   b'11110000'    ; Make the lower 4 bits of PORTB outputs
movwf   TRISB
bcf     STATUS, RP0    ; Switch to bank 0
bsf     PORTB, 0       ; Set bit 0 of PORTB

```

**The Task:**

Turn on the LED connected to bit 0 of PORTB (RB0).

A project template has been created for you at:  
**C:\Masters\1003\Lab1\Lab1.mcp**

The project template has all the essential program elements except for 6 instructions. The comments will guide you as to what you need to write to make the program work.

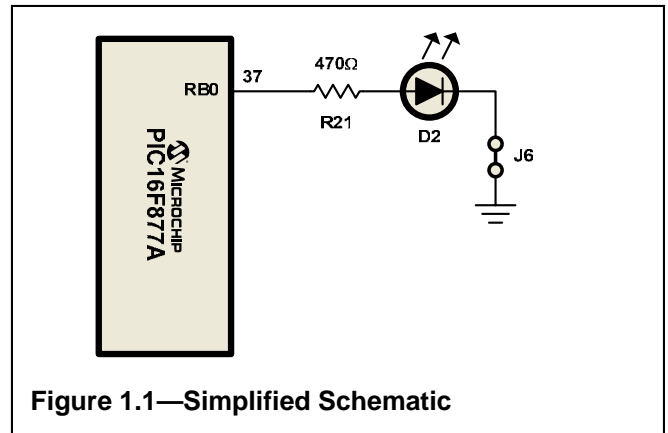
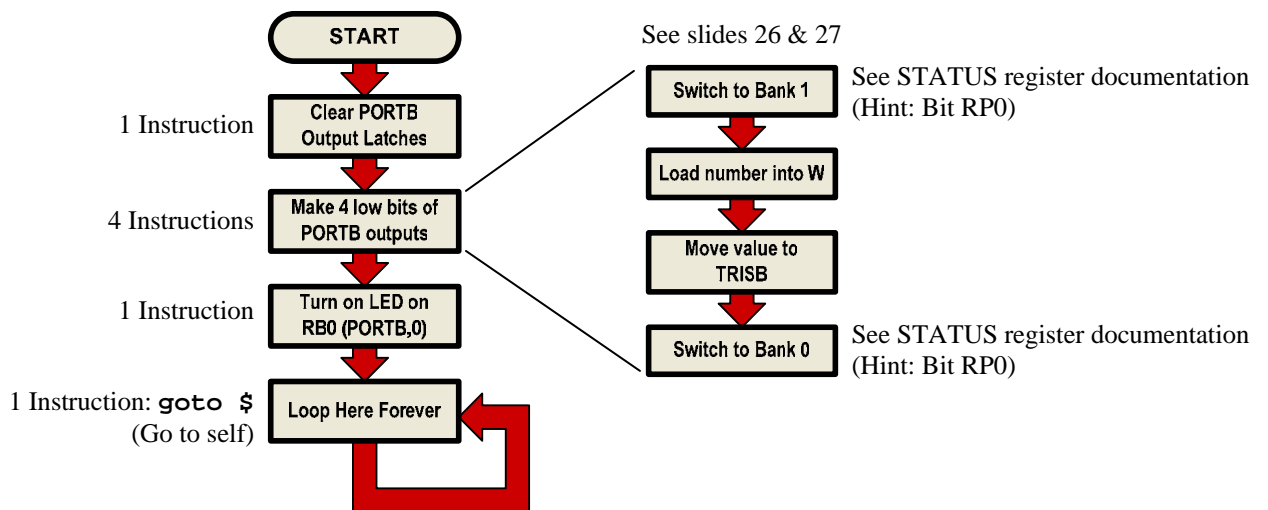


Figure 1.1—Simplified Schematic

**Program Flow:**



**Program Template:**

```

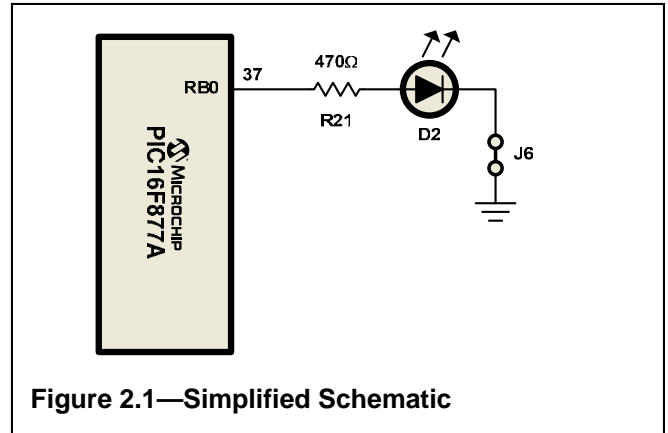
Lab 1: "Hello, world!" for Microcontrollers
1  LIST p=16f877a
2
3  #include <p16f877a.inc>
4
5  org    0x0000
6  RESET_v  goto    START      ;Reset Vector
7
8  START    {1st Instruction}   ;Clear PORTB output latches
9           {2nd Instruction}   ;Switch to bank 1
10          {3rd Instruction}   ;Load value to make lower 4 bits outputs
11          {4th Instruction}   ;Move value to TRISB
12          {5th Instruction}   ;Switch to bank 0
13          {6th Instruction}   ;Turn on LED on RB0
14
15          goto $              ;Loop here forever
16
17  END
    
```

**The Task:**

Make the LED connected to bit 0 of PORTB (RB0) blink at a rate that is slow enough to be seen by the human eye.

A project template has been created for you at:  
**C:\Masters\1003\Lab2\Lab2.mcp**

The project template has all the essential program elements except for 6 instructions. The comments will guide you as to what you need to write to make the program work.

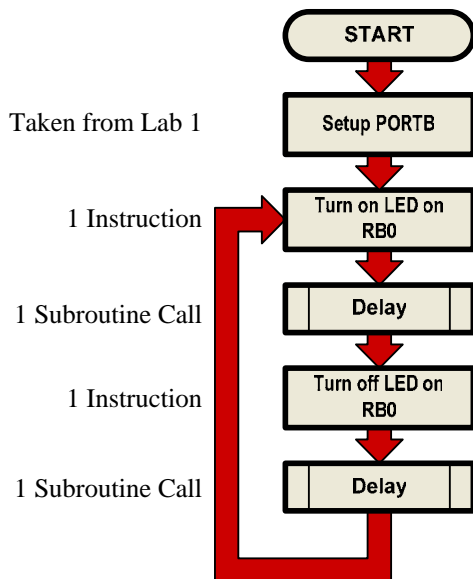


**Figure 2.1—Simplified Schematic**

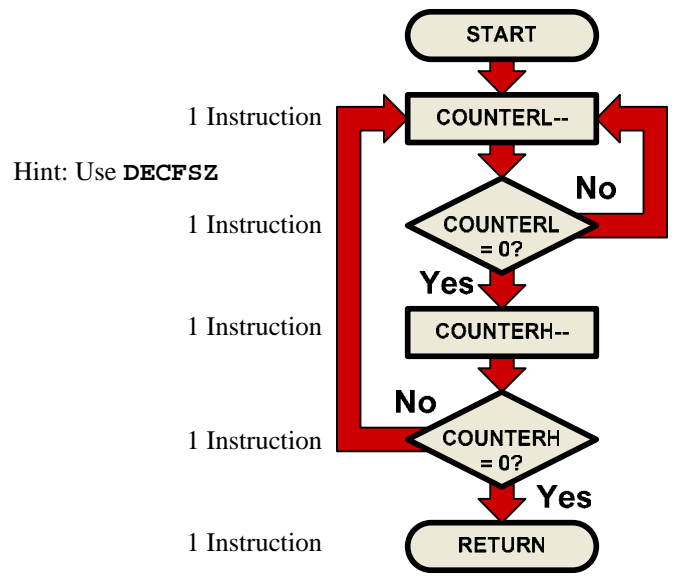
**Why do we need a delay?**

With the PICmicro running at 4MHz, each instruction takes only 1µs to execute. Because of this, it will be necessary to slow down the main program loop which would only take 4µs to execute without any delays. If we don't include the delay, the LED will blink so fast that it will appear to be on constantly, though at a level slightly less than full brightness. The delay routine consists of two registers used to implement a 16-bit counter. In the delay loop, this counter will be decremented through each pass until it reaches zero. If you are curious as to how long this delay lasts, it is fairly easy to calculate if you remember that all instructions execute in a single cycle (1µs) except for program branches which take two cycles (2µs).

**Program Flow:**



**Delay Subroutine:**



## Lab 2: Blinking LED

```

1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11     RESET_V  goto      START      ;Reset Vector
12
13     START    clrf      PORTB      ;Clear PORTB output latches
14             bsf      STATUS,RP0   ;Switch to bank 1
15             movlw   b'11110000'  ;Load value to make lower 4 bits outputs
16             movwf  TRISB        ;Move value to TRISB
17             bcf      STATUS,RP0   ;Switch to bank 0
18
19     LOOP     {1st Instruction}    ;Turn on LED on RB0
20             {2nd Instruction}    ;Call delay routine
21             {3rd Instruction}    ;Turn off LED on RB0
22             {4th Instruction}    ;Call delay routine
23             {5th Instruction}    ;Repeat main loop
24
25     DELAY    {6th Instruction}    ;Decrement COUNTERL
26             {7th Instruction}    ;If not zero, keep decrementing COUNTERL
27             {8th Instruction}    ;Decrement COUNTERH
28             {9th Instruction}    ;If not zero, decrement COUNTERL again
29             {10th Instruction}   ;Return to main routine
30
31     END

```

Notes: COUNTERL and COUNTERH are the two registers used to implement the 16-bit counter. They are defined by us on lines 5-8 such that COUNTERL is at address 0x20 and COUNTERH is at address 0x21.

Lines 10-17 are identical to the initialization code from Lab 1.

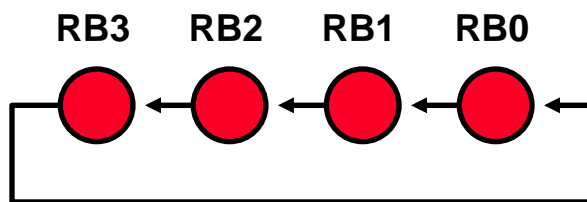
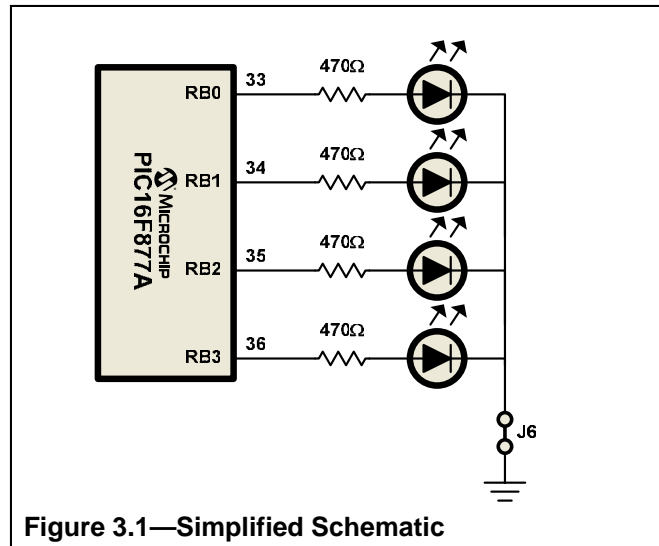
**The Task:**

Using one of the rotate instructions, make the illuminated LED “move” across the four LEDs on PORTB. When it reaches the other side, send it back to the start.

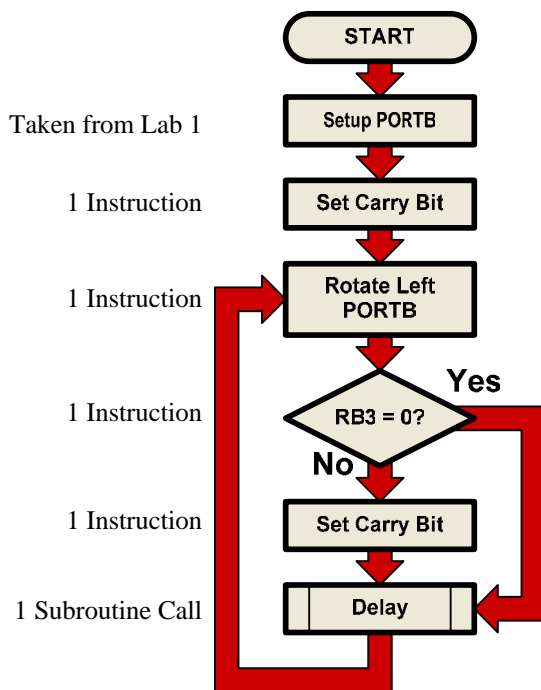
A project template has been created for you at:  
**C:\Masters\1003\Lab3\Lab3.mcp**

The project template has all the essential program elements except for 6 instructions. The comments will guide you as to what you need to write to make the program work.

The same delay routine used in LAB2 will be required here, otherwise it will appear as if all the LEDs are on at the same time because they are rotating so fast.



**Program Flow:**



**Remember: The rotate instructions operate on 9-bits, with the Carry bit in the STATUS register as the 9th bit.**

## Lab 3: Rotating LED

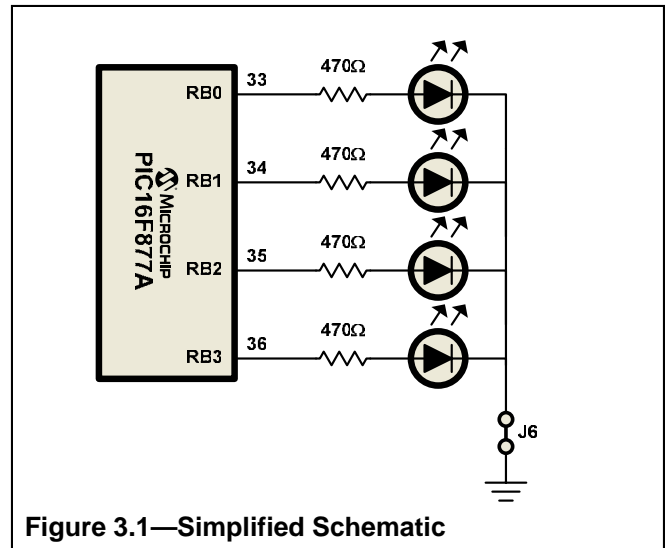
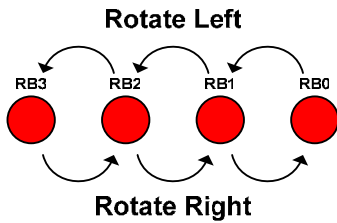
```

1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11     RESET_V  goto      START          ;Reset Vector
12
13     START    clrf      PORTB          ;Clear PORTB output latches
14             bsf      STATUS,RP0      ;Switch to bank 1
15             movlw   b'11110000'     ;Load value to make lower 4 bits outputs
16             movwf  TRISB            ;Move value to TRISB
17             bcf      STATUS,RP0      ;Switch to bank 0
18
19             {1st Instruction}        ;Set carry bit for initial rotate
20     LOOP    {2nd Instruction}        ;Rotate PORTB to left
21             {3rd Instruction}        ;Call delay routine
22             {4th Instruction}        ;Is the LED on RB3 (PORTB,3) on?
23             {5th Instruction}        ;If yes, set the Carry bit
24             {6th Instruction}        ;Repeat main loop
25
26     DELAY    decfsz   COUNTERL        ;Decrement COUNTERL
27             goto     DELAY            ;If not zero, keep decrementing COUNTERL
28             decfsz   COUNTERH        ;Decrement COUNTERH
29             goto     DELAY            ;If not zero, decrement COUNTERL again
30             return                    ;Return to main subroutine
31
32     END

```

**The Task:**

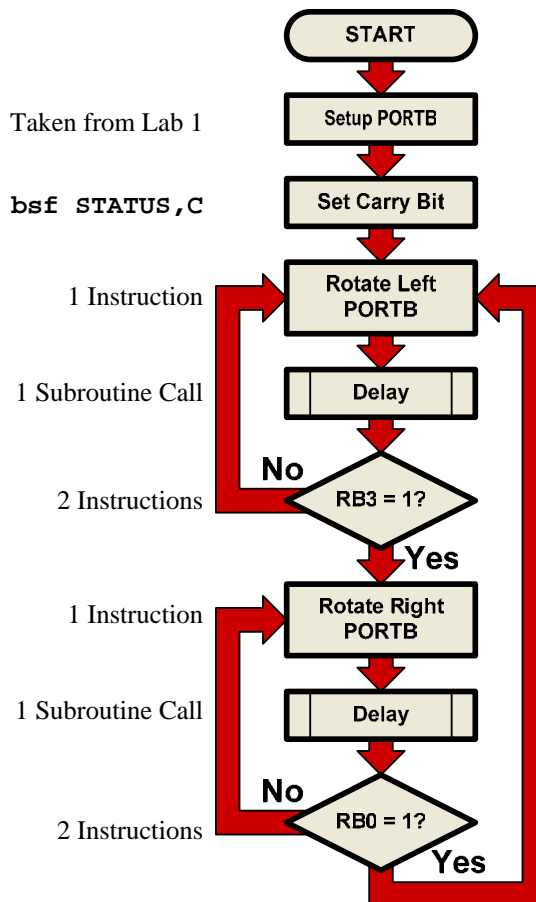
Similar to Lab 3, but this time, make the direction of rotation change whenever the illuminated LED reaches one end or the other.



A project template has been created for you at:  
**C:\Masters\1003\Lab4\Lab4.mcp**

The project template has all the essential program elements except for 6 instructions. The comments will guide you as to what you need to write to make the program work.

**Program Flow:**



## Lab 4: Alternating Rotating LED

```

1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11     RESET_V  goto      START          ;Reset Vector
12
13     START    clrfsz   PORTB          ;Clear PORTB output latches
14             bsf      STATUS,RP0     ;Switch to bank 1
15             movlw   b'11110000'    ;Load value to make lower 4 bits outputs
16             movwf  TRISB          ;Move value to TRISB
17             bcf      STATUS,RP0     ;Switch to bank 0
18
19             bsf      STATUS,C        ;Set carry bit for initial rotate
20     LEFT     {1st Instruction}      ;Rotate PORTB to left
21             {2nd Instruction}      ;Call delay routine
22             {3rd Instruction}      ;Is the LED on RB3 (PORTB,3) on?
23             {4th Instruction}      ;if no, rotate left again
24
25     RIGHT    {5th Instruction}      ;Rotate PORTB to right
26             {6th Instruction}      ;Call delay routine
27             {7th Instruction}      ;Is the LED on RB0 (PORTB,0) on?
28             {8th Instruction}      ;if no, rotate right again
29             {9th Instruction}      ;if yes, rotate left
30
31     DELAY    decfsz  COUNTERL        ;Decrement COUNTERL
32             goto    DELAY           ;If not zero, keep decrementing COUNTERL
33             decfsz  COUNTERH        ;Decrement COUNTERH
34             goto    DELAY           ;If not zero, decrement COUNTERL again
35             return                  ;Return to main subroutine
36
37     END

```



## Lab 5: Lookup Table

```

1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11     RESET_V  goto    START      ;Reset Vector
12
13     START    clrfsz  PORTB      ;Clear PORTB output latches
14             bsf     STATUS,RP0  ;Switch to bank 1
15             movlw   b'11110000' ;Load value to make lower 4 bits outputs
16             movwf   TRISB      ;Move value to TRISB
17             bcf     STATUS,RP0  ;Switch to bank 0
18
19             {1st Instruction}    ;Clear index into table
20             {2nd Instruction}    ;Load W with high byte of TABLE address
21             {3rd Instruction}    ;Move W to PCLATH
22     LOOP    {4th Instruction}    ;Move INDEX to W
23             {5th Instruction}    ;Call TABLE
24             {6th Instruction}    ;Move W to PORTB
25             {7th Instruction}    ;Call delay
26             {8th Instruction}    ;Increment INDEX
27             {9th Instruction}    ;Load W with 0x07
28             {10th Instruction}   ;Subtract W from INDEX, result in W
29             {11th Instruction}   ;Is Z bit in STATUS set?
30             {12th Instruction}   ;if yes, clear INDEX
31             {13th Instruction}   ;Repeat loop
32
33     DELAY    decfsz  COUNTERL     ;Decrement COUNTERL
34             goto    DELAY        ;If not zero, keep decrementing COUNTERL
35             decfsz  COUNTERH     ;Decrement COUNTERH
36             goto    DELAY        ;If not zero, decrement COUNTERL again
37             return               ;Return to main subroutine
38
39     TABLE   addwf    PCL,f       ;Add offset to program counter
40             retlw   b'00000001'  ;Table entry 0
41             retlw   b'00000011'  ;Table entry 1
42             retlw   b'00000111'  ;Table entry 2
43             retlw   b'00001111'  ;Table entry 3
44             retlw   b'00001110'  ;Table entry 4
45             retlw   b'00001100'  ;Table entry 5
46             retlw   b'00001000'  ;Table entry 6
47
48     END

```

# Lab Solutions

---

## Lab 1: "Hello, world!" for Microcontrollers

```
1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      org      0x0000
6 RESET_V goto      START      ;Reset Vector
7
8 START      clrf    PORTB      ;Clear PORTB output latches
9           bsf     STATUS,RP0  ;Switch to bank 1
10          movlw   b'11110000' ;Load value to make lower 4 bits outputs
11          movwf  TRISB       ;Move value to TRISB
12          bcf    STATUS,RP0  ;Switch to bank 0
13          bsf    PORTB,0     ;Turn on LED on RB0
14
15          goto   $           ;Loop here forever
16
17          END
```

## Lab 2: Blinking LED

```
1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11 RESET_V    goto      START      ;Reset Vector
12
13 START      clrfsz   PORTB        ;Clear PORTB output latches
14           bsf      STATUS,RP0    ;Switch to bank 1
15           movlw   b'11110000'    ;Load value to make lower 4 bits outputs
16           movwf  TRISB          ;Move value to TRISB
17           bcf      STATUS,RP0    ;Switch to bank 0
18
19 LOOP      bsf      PORTB,0        ;Turn on LED on RB0
20           call    DELAY          ;Call delay routine
21           bcf      PORTB,0        ;Turn off LED on RB0
22           call    DELAY          ;Call delay routine
23           goto    LOOP          ;Repeat main loop
24
25 DELAY      decfsz  COUNTERL       ;Decrement COUNTERL
26           goto    DELAY          ;If not zero, keep decrementing COUNTERL
27           decfsz  COUNTERH       ;Decrement COUNTERH
28           goto    DELAY          ;If not zero, decrement COUNTERL again
29           return
30
31     END
```

## Lab 3: Rotating LED

```
1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11 RESET_V goto      START      ;Reset Vector
12
13 START      clrf     PORTB      ;Clear PORTB output latches
14           bsf     STATUS,RP0   ;Switch to bank 1
15           movlw   b'11110000' ;Load value to make lower 4 bits outputs
16           movwf   TRISB       ;Move value to TRISB
17           bcf     STATUS,RP0   ;Switch to bank 0
18
19           bsf     STATUS,C      ;Set carry bit for initial rotate
20 LOOP      rlf     PORTB,f      ;Rotate PORTB to left
21           call    DELAY        ;Call delay routine
22           btfsc   PORTB,3      ;Is the LED on RB3 (PORTB,3) on?
23           bsf     STATUS,C      ;If yes, set the Carry bit
24           goto    LOOP        ;Repeat main loop
25
26 DELAY     decfsz  COUNTERL     ;Decrement COUNTERL
27           goto    DELAY        ;If not zero, keep decrementing COUNTERL
28           decfsz  COUNTERH     ;Decrement COUNTERH
29           goto    DELAY        ;If not zero, decrement COUNTERL again
30           return              ;Return to main subroutine
31
32     END
```

## Lab 4: Alternating Rotating LED

```

1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11     RESET_V  goto      START      ;Reset Vector
12
13     START    clrf      PORTB      ;Clear PORTB output latches
14             bsf      STATUS,RP0   ;Switch to bank 1
15             movlw   b'11110000'  ;Load value to make lower 4 bits outputs
16             movwf  TRISB        ;Move value to TRISB
17             bcf      STATUS,RP0   ;Switch to bank 0
18
19             bsf      STATUS,C      ;Set carry bit for initial rotate
20
21     LEFT     rlf      PORTB,f      ;Rotate PORTB to left
22             call    DELAY         ;Call delay routine
23             btfss   PORTB,3      ;Is the LED on RB3 (PORTB,3) on?
24             goto    LEFT         ;if no, rotate left again
25
26     RIGHT    rrf      PORTB,f      ;Rotate PORTB to right
27             call    DELAY         ;Call delay routine
28             btfss   PORTB,0      ;Is the LED on RB0 (PORTB,0) on?
29             goto    RIGHT        ;if no, rotate right again
30             goto    LEFT         ;if yes, rotate left
31
32     DELAY    decfsz   COUNTERL     ;Decrement COUNTERL
33             goto    DELAY         ;If not zero, keep decrementing COUNTERL
34             decfsz   COUNTERH     ;Decrement COUNTERH
35             goto    DELAY         ;If not zero, decrement COUNTERL again
36             return                ;Return to main subroutine
37
38     END

```

## Lab 5: Lookup Table

```

1      LIST p=16f877a
2
3      #include <p16f877a.inc>
4
5      cblock 0x020
6          COUNTERL
7          COUNTERH
8      endc
9
10     org      0x0000
11     RESET_V goto      START          ;Reset Vector
12
13     START   clrf      PORTB          ;Clear PORTB output latches
14           bsf      STATUS,RP0      ;Switch to bank 1
15           movlw   b'11110000'      ;Load value to make lower 4 bits outputs
16           movwf  TRISB            ;Move value to TRISB
17           bcf      STATUS,RP0      ;Switch to bank 0
18
19           clrf    INDEX            ;Clear index into table
20           movlw  HIGH TABLE      ;Load W with high byte of TABLE address
21           movwf  PCLATH           ;Move W to PCLATH
22     LOOP   movf    INDEX,w         ;Move INDEX to W
23           call   TABLE           ;Call TABLE
24           movwf  PORTB           ;Move W to PORTB
25           call   DELAY            ;Call delay
26           incf   INDEX,f         ;Increment INDEX
27           movlw  0x07             ;Load W with 0x07
28           subwf  INDEX,w         ;Subtract W from INDEX, result in W
29           btfsc  STATUS,Z         ;Is Z bit in STATUS set?
30           clrf   INDEX           ;if yes, clear INDEX
31           goto   LOOP            ;Repeat loop
32
33     DELAY   decfsz  COUNTERL       ;Decrement COUNTERL
34           goto   DELAY           ;If not zero, keep decrementing COUNTERL
35           decfsz  COUNTERH       ;Decrement COUNTERH
36           goto   DELAY           ;If not zero, decrement COUNTERL again
37           return                ;Return to main subroutine
38
39     TABLE addwf   PCL,f          ;Add offset to program counter
40           retlw  b'00000001'      ;Table entry 0
41           retlw  b'00000011'      ;Table entry 1
42           retlw  b'00000111'      ;Table entry 2
43           retlw  b'00001111'      ;Table entry 3
44           retlw  b'00001110'      ;Table entry 4
45           retlw  b'00001100'      ;Table entry 5
46           retlw  b'00001000'      ;Table entry 6
47
48     END

```