

- 2

- 3

```

; *****
; Lesson 3 - "Rotate"
;
; This lesson will introduce shifting instructions as well as bit-oriented skip operations to
; move the LED display.
;
; LEDs rotate from right to left at a rate of 1.5s
;
;
; PIC: 16F1829
; Assembler: MPASM v5.43
; IDE: MPLABX v1.10
;
; Board: PICKit 3 Low Pin Count Demo Board
; Date: 6.1.2012
;
; *****
; * See Low Pin Count Demo Board User's Guide for Lesson Information*
; *****

#include <p16F1829.inc>
    _CONFIG _CONFIG1, (_FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF & _MCLRE_OFF & _CP_OFF & _CPD_OFF &
    _BOREN_ON & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_OFF);
    _CONFIG _CONFIG2, (_WRT_OFF & _PLLEN_OFF & _STVREN_OFF & _LVP_OFF);

    errorlevel -302                      ;supress the 'not in bank0' warning
    cblock 0x70                          ;shared memory location that is accessible from all banks
Delay1                                  ;define two file registers for the delay loop in shared memory
Delay2
    endc

; -----LATC-----
; Bit#:  -7---6---5---4---3---2---1---0---
; LED:   -----|DS4|DS3|DS2|DS1|-
; -----

ORG 0                      ;start of code
Start:
    banksel    OSCCON      ;bank1
    movlw      b'00111000' ;set cpu clock speed of 500KHz
    movwf      OSCCON      ;move contents of the working register into OSCCON
    clrf       TRISC       ;make all of PORTC an output
    banksel    LATC        ;select the bank where LATC is (bank2)
    movlw      b'00001000' ;start the rotation by setting DS4 ON
    movwf      LATC        ;write contents of the working register to the latch
MainLoop:
OndelayLoop:
    decfsz     Delay1,f     ;Waste time.
    goto       OndelayLoop ;The Inner loop takes 3 instructions per loop * 256 loopss = 768
instructions
    decfsz     Delay2,f     ;The outer loop takes an additional 3 instructions per lap * 256
loops
    goto       OndelayLoop ;(768+3) * 256 = 197376 instructions / 125K instructions per second
    = 1.579 sec.

Rotate:
    lsrwf      LATC,F       ;shift the LEDs and turn on the next LED to the right
    btfsc      STATUS,C     ;did the bit rotate into the carry (i.e. was DS1 just lit?)
    bsf        LATC, 3      ;yes, it did and now start the sequence over again by turning on
DS4
    goto       MainLoop     ;repeat this program forever

end                          ;end code section

```

```

/**
*****
* Lesson 3 - "Rotate"
*
* This lesson will introduce shifting instructions as well as bit-oriented skip operations to
* move the LED display.
*
* LEDs rotate from right to left at a rate of 1.5s
*
* PIC: 16F1829
* Compiler: XC8 v1.00
* IDE: MPLABX v1.10
*
* Board: PICkit 3 Low Pin Count Demo Board
* Date: 6.1.2012
*
* *****
* See Low Pin Count Demo Board User's Guide for Lesson Information*
* *****
*/

#include <htc.h>                                //PIC hardware mapping
#define _XTAL_FREQ 500000                      //Used by the XC8 delay_ms(x) macro

//config bits that are part-specific for the PIC16F1829
__CONFIG(FOSC_INTOSC & WDTE_OFF & PWRTE_OFF & MCLRE_OFF & CP_OFF & CPD_OFF & BOREN_ON & CLKOUTEN_OFF &
    IESO_OFF & FCMEN_OFF);
__CONFIG(WRT_OFF & PLLEN_OFF & STVREN_OFF & LVP_OFF);

/* -----LATC-----
* Bit#:  -7---6---5---4---3---2---1---0---
* LED:    -----|DS4|DS3|DS2|DS1|-
* -----
*/

void main(void) {
    TRISC = 0;                                //all pins are outputs
    OSCCON = 0b00111000;                      //500KHz clock speed
    LATC = 0b0001000;                          //start the rotation by setting DS4 ON - rotate
    from the right to left

    while (1) {
        __delay_ms(500);                      //delay 500ms
        LATC >> = 1;                          //shift to the right by 1
        if(STATUSbits.C)                      //when the last LED is lit, restart the pattern
            LATCbits.LATC3 = 1;
    }
}

```



```
banksel    LATC                ;bank2
lshf       LATC, f
btfsc      STATUS,C           ;did the bit rotate into the carry?
bsf        LATC,3             ;yes, put light DS4 back up

bra        MainLoop           ;continue forever

end
```

```

/**
*****
* Lesson 9 - "Timer0"
*
* Timer0 is a counter implemented in the processor. It may be used to count instruction
* cycles or external events, that occur at or below the instruction cycle rate.
* In the PIC18, Timer0 can be used as either an 8-bit or 16-bit counter, or timer. The
* enhanced mid-range core implements only an 8-bit counter.
* This lesson configures Timer0 to count instruction cycles and to set a flag when it rolls
* over. This frees up the processor to do meaningful work rather than wasting instruction
* cycles in a timing loop.
* Using a counter provides a convenient method of measuring time or delay loops as it
* allows the processor to work on other tasks rather than counting instruction cycles.
*
*
* LEDs rotate from right to left, similar to Lesson 3, at a rate of ~.5 seconds.
*
* PIC: 16F1829
* Compiler: XC8 v1.00
* IDE: MPLABX v1.10
*
* Board: PICkit 3 Low Pin Count Demo Board
* Date: 6.1.2012
*
* *****
* See Low Pin Count Demo Board User's Guide for Lesson Information*
* *****
*/

#include <htc.h>                                //PIC hardware mapping
#define _XTAL_FREQ 500000                      //Used by the XC8 delay_ms(x) macro

//config bits that are part-specific for the PIC16F1829
__CONFIG(FOSC_INTOSC & WDTE_OFF & PWRTE_OFF & MCLRE_OFF & CP_OFF & CPD_OFF & BOREN_ON & CLKOUTEN_OFF &
    IESO_OFF & FCMEN_OFF);
__CONFIG(WRT_OFF & PLLEN_OFF & STVREN_OFF & LVP_OFF);

/* -----LATC-----
* Bit#:  -7---6---5---4---3---2---1---0---
* LED:    -----|DS4|DS3|DS2|DS1|-
*-----
*/

void main(void) {
    OSCCON = 0b00111000;                        //500KHz clock speed
    TRISC = 0;                                  //all LED pins are outputs
    LATC = 0;
    OPTION_REG = 0b00000111;                    //1:256 prescaler for a delay of: (instruction-cycle *
    256-counts)*prescaler = ((8uS * 256)*256) =~ 524mS
    LATCbits.LATC4 = 1;                          //start with DS4 lit

    while (1) {
        while (!INTCONbits.TMR0IF) continue;    //you can let the PIC do work here, but for now we will
        wait for the flag
        INTCONbits.T0IF = 0;                    //flag MUST be cleared in software
        LATC >> = 1;                             //rotate the LEDs
        if (STATUSbits.C)                        //when the last LED is lit, restart the pattern
            LATCbits.LATC3 = 1;

    }
}

```



```

; *****
; Lesson 10 - Interrupts and Pull-ups
;
; This lesson will introduce interrupts and how they are useful. It will
; also introduce internal weak pull-ups that are available on most PICs.
;
; It should be noted that this lesson is more efficient than the last
; one, "Timer0". Notice how the processor is no longer waiting for
; Timer0 to roll over. Instead, we let the hardware modules do the work,
; freeing the CPU to do other things in the main loop
;
; The switch is no longer continuously polled for a button press. Instead,
; an interrupt will occur which will automatically place the program counter
; inside of the ISR where we can change directions outside of normal code execution
;
; LEDs rotate at a constant speed and the switch reverses their direction
;
; PIC: 16F1829
; Assembler: MPASM v5.43
; IDE: MPLABX v1.10
;
; Board: PICkit 3 Low Pin Count Demo Board
; Date: 6.1.2012
;
; *****
; * See Low Pin Count Demo Board User's Guide for Lesson Information*
; *****

#include <p16F1829.inc>
    __CONFIG __CONFIG1, (_FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF & _MCLRE_OFF & _CP_OFF & _CPD_OFF &
    _BOREN_ON & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_OFF);
    __CONFIG __CONFIG2, (_WRT_OFF & _PLLEN_OFF & _STVREN_OFF & _LVP_OFF);

    errorlevel -302                                ;suppress the 'not in bank0' warning

#define SWITCH PORTA, 2                            ;pin where SW1 is connected..NOTE: always READ from the PORT and
    WRITE to the LATCH

#define PULL_UPS                                    ;if this is uncommented, JP5 can be pulled out

#define LED_RIGHT 0xFF                             ;keep track of LED direction
#define LED_LEFT 0x00

    cblock 0x70                                     ;shared memory location that is accessible from all banks
    Direction
    Delay1
    endc

; -----LATC-----
; Bit#:  -7---6---5---4---3---2---1---0---
; LED:  -----|DS4|DS3|DS2|DS1|-
; -----

    Org 0x0                                         ;Reset Vector starts at 0x0000
    bra Start                                       ;main code execution
    Org 0x0004                                     ;Interrupt Vector starts at address 0x0004
    goto ISR

Start:
;Setup main init
    banksel OSCCON                                ;bank1
    movlw b'00111000'                             ;set cpu clock speed FO 500KHz
    movwf OSCCON                                  ;move contents of the working register into OSCCON

    bsf TRISA, RA2                                ;switch as input
    banksel ANSELA                                ;bank3

```

```

    bcf          ANSELA, RA2          ;digital
                                        ;can reference pins by their position in the PORT (2) or name (RA2)

    banksel     TRISC                 ;Configure the LEDs
    clrf        TRISC                 ;bank1
    banksel     LATC                  ;make all of PORTC an output
    movlw       b'00001000'          ;bank2
                                        ;start with DS4 lit

    banksel     OPTION_REG            ;Setup Timer0 as the delay
    movlw       b'00000111'          ;bank1
    prescaler = ((8uS * 256)*256) =~ 524mS ;1:256 prescaler for a delay of: (insruction-cycle * 256-counts)*
    movwf       OPTION_REG
    bsf         INTCON, TMR0IE        ;enable the rollover interrupt to occur

    bsf         INTCON, IOCIE         ;Setup interrupt-on-change for the switch
    flags to cause an interrupt      ;must set this global enable flag to allow any interrupt-on-change
    banksel     IOCAN                 ;bank7
    bsf         IOCAN, IOCAN2         ;when SW1 is pressed, enter the ISR (Note, this is set when a
    FALLING EDGE is detected)        ;if this is not set, the interrupt flags will still get set, but
    bsf         INTCON, GIE           ;must set this global to allow any interrupt to bring the program
    into the ISR                     ;if this is not set, the interrupt flags will still get set, but

    the ISR will never be entered

#ifdef PULL_UPS                      ;enter here if this is defined (not commented out)
    banksel     WPUA                  ;bank4
    bsf         WPUA, 2               ;enable the weak pull-up for the switch
    banksel     OPTION_REG            ;bank1
    bcf         OPTION_REG, NOT_WPUEN ;enable the global weak pull-up bit
    ;this bit is active HIGH, meaning it must be cleared for it to be enabled
#endif

    movlw       LED_RIGHT             ;start with LEDs shifting to the right
    movwf       Direction

    ;Clear the RAM
    clrf        Delay1

MainLoop:
    bra         MainLoop              ;can spend rest of time doing something critical here

Debounce:
    movlw       d'209'                ;delay for approximatly 5ms
    movwf       Delay1                ;(1/(500KHz/4))*209*3 = 5.016mS

DebounceLoop:
    decfsz      Delay1, f              ;1 instruction to decrement,unless if branching (ie Delay1 = 0)
    bra         DebounceLoop          ;2 instructions to branch
    return

RotateRight:
    lsr         LATC, f                ;logical shift right
    btfsc       STATUS,C               ;did the bit rotate into the carry?
    bsf         LATC,3                 ;yes, put it into bit 3.
    retfie

RotateLeft:
    lsl         LATC, f                ;logical shift left
    btfsc       LATC, 4                ;did it rotate out of the LED display?
    bsf         LATC, 0                ;yes, put in bit 0
    retfie

    ;Enter here if an interrupt has occurred
    ;First, check what caused the interrupt by checking the ISR flags

```

```

;This lesson only has 2 flags to check

ISR:
    banksel IOCAF                ;bank7
    btfsc      IOCAF, 2          ;check the interrupt-on-change flag
    bra        Service_SW1      ;switch was pressed
    bra        Service_TMR0     ;Timer0 overflowed
Service_SW1:
    ;In order to ensure that no detected edge is lost while clearing
    flags,
    't
    current
    ;the following 3 lines mask out only the known changed bits and don't
    ;interfere with the others. A simple clrxf would work, but this
    ;method is good practice
    movlw      0xFF
    xorwf      IOCAF, w
    andwf      IOCAF, f
    forever
    ;MUST ALWAYS clear this in software or else stuck in the ISR
    ;clearing this will clear the INTCON, IOCIF bit as well

    call       Debounce         ;delay for 5ms and then check the switch again

    banksel    PORTA
    btfsc      SWITCH
    retfie     ;nope, exit the ISR back to the main code

    movlw      0xFF
    xorwf      Direction, f
    retfie     ;toggle the direction state and save it back
                ;return to main code

Service_TMR0:
    bcf        INTCON, T0IF     ;MUST ALWAYS clear this in software or else stuck in the ISR
    forever
    banksel    LATC             ;change to bank2
    movlw      LED_RIGHT        ;check what direction currently in
    subwf      Direction, w     ;be sure to save in wreg so as to not corrupt 'Direction'
    btfsc      STATUS, Z
    bra        RotateRight
    bra        RotateLeft

end
;end code generation

```

```

/**
*****
* Lesson 10 - "Interrupts and Pull-ups"
*
* This lesson will introduce interrupts and how they are useful. It will
* also introduce internal weak pull-ups that are available on most PICs.
*
* It should be noted that this lesson is more efficient than the last
* one, "Timer0". Notice how the processor is no longer waiting for
* Timer0 to roll over. Instead, we let the hardware modules do the work,
* freeing the CPU to do other things in the main loop
*
* The switch is no longer continuously polled for a button press. Instead,
* an interrupt will occur which will automatically place the program counter
* inside of the ISR where we can change directions outside of normal code execution
*
* LEDs rotate at a constant speed and the switch reverses their direction
*
* PIC: 16F1829
* Compiler: XC8 v1.00
* IDE: MPLABX v1.10
*
* Board: PICKit 3 Low Pin Count Demo Board
* Date: 6.1.2012
*
* *****
* See Low Pin Count Demo Board User's Guide for Lesson Information*
* *****
*/

#include <htc.h> //PIC hardware mapping
#define _XTAL_FREQ 500000 //Used by the XC8 delay_ms(x) macro

#define DOWN 0
#define UP 1

#define SWITCH PORTAbits.RA2

#define LED_RIGHT 1
#define LED_LEFT 0

#define PULL_UPS //if this is uncommented, the trace under JP5 can be cut
//with no affect on the output

//config bits that are part-specific for the PIC16F1829
__CONFIG(FOSC_INTOSC & WDTE_OFF & PWRTE_OFF & MCLRE_OFF & CP_OFF & CPD_OFF & BOREN_ON & CLKOUTEN_OFF &
IESO_OFF & FCMEN_OFF);
__CONFIG(WRT_OFF & PLLEN_OFF & STVREN_OFF & LVP_OFF);

/* -----LATC-----
* Bit#: -7---6---5---4---3---2---1---0---
* LED: -----|DS4|DS3|DS2|DS1|-
* -----
*/

unsigned char _direction; //a global variable
void main(void) {
    //general init
    OSCCON = 0b00111000; //500KHz clock speed
    TRISC = 0; //all LED pins are outputs
    LATC = 0; //init LEDs in OFF state

    LATCbits.LATC3 = 1; //DS4 is lit
    _direction = LED_RIGHT; //start with LEDs rotating from right to left

    //setup switch (SW1)

```

```

    TRISAbits.TRISA2 = 1;           //switch as input
    ANSELAbits.ANSA2 = 0;          //digital switch

                                   //by using the internal resistors, you can save cost by
    eleminating an external pull-up/down resistor
#ifdef PULL_UPS
    WPUA2 = 1;                     //enable the weak pull-up for the switch
    nWPUEN = 0;                   //enable the global weak pull-up bit
#endif

                                   //setup TIMER0 as the delay
                                   //1:256 prescaler for a delay of: (insruction-cycle * 256-
counts)*prescaler = ((8uS * 256)*256) =~ 524mS
OPTION_REG = 0b00000111;         //setup TIMER0
INTCONbits.TMR0IE = 1;           //enable the TMR0 rollover interrupt

                                   //setup interrupt on change for the switch
INTCONbits.IOCIE = 1;            //enable interrupt on change global
IOCANbits.IOCAN2 = 1;            //when SW1 is pressed, enter the ISR
GIE = 1;                         //enable global interrupts

while (1) {
    continue;                     //can spend rest of time doing something critical here
}

void interrupt ISR(void) {
    if (IOCAF) {                  //SW1 was just pressed
        IOCAF = 0;               //must clear the flag in software
        __delay_ms(5);           //debounce by waiting and seeing if still held down
        if (SWITCH == DOWN) {
            _direction ^= 1;     //change directions
        }
    }

    if (INTCONbits.T0IF) {
        INTCONbits.T0IF = 0;

        if (_direction == LED_RIGHT) {
            LATC >> = 1;         //rotate right
            if (STATUSbits.C == 1) //when the last LED is lit, restart the pattern
                LATCbits.LATC3 = 1;
        } else{
            LATC << = 1;         //rotate left
            if (LATCbits.LATC4 == 1) //when the last LED is lit, restart the pattern
                LATCbits.LATC0 = 1;
        }
    }
}
}

```