EECE.3170: Microprocessor Systems Design I

Summer 2016

Lecture 13: Key Questions
June 20, 2016

1. What is an interrupt? What is an exception?

2. For what purposes are interrupts useful?

3. Describe the basic steps in interrupt processing.

4. What is an interrupt or exception vector? Describe briefly how an interrupt vector table functions.

5. Explain how interrupts can be set up and managed in the PIC microcontrollers.

6. Explain the operation of the programs used to rotate the LEDs using interrupts (interrupt.asm and interrupt.c).

7. Explain how the analog to digital converter module is configured in PIC microcontrollers.

8. Explain the operation of the programs used to test the ADC (a2d.asm and a2d.c).

9. Explain the programs that use the ADC result to vary the speed of rotation (vs_rotate.asm and vs_rotate.c).

```
; *****************************************************************
; 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 efficent 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 automically place the program counter
; inside of the ISR where we can change directions outisde 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                     ;surpress 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)

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

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

                                        ;Setup interrupt-on-change for the switch
     bsf             INTCON, IOCIE      ;must set this global enable flag to allow any interrupt-on-change  ↙
    flags to cause an interrupt
     banksel         IOCAN              ;bank7
     bsf             IOCAN,  IOCAN2     ;when SW1 is pressed, enter the ISR (Note, this is set when a       ↙
    FALLING EDGE is detected)
     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:
                                        ;delay for approximatly 5ms
    movlw           d'209'              ;(1/(500KHz/4))*209*3 = 5.016mS
    movwf           Delay1
DebounceLoop:
    decfsz          Delay1, f           ;1 instruction to decrement,unless if branching (ie Delay1 = 0)
    bra             DebounceLoop        ;2 instructions to branch
    return

RotateRight:
     lsrf           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:
     lslf           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 occured
                                        ;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,
                                    ;the following 3 lines mask out only the known changed bits and don ↙
    't
                                    ;interfere with the others. A simple clrf would work, but this ↙
    current
                                    ;method is good practice
    movlw           0xFF
    xorwf           IOCAF, w
    andwf           IOCAF, f            ;MUST ALWAYS clear this in software or else stuck in the ISR ↙
    forever
                                    ;clearing this will clear the   INTCON, IOCIF   bit as well

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

    banksel         PORTA               ;bank0
    btfsc           SWITCH              ;is it still held down?
    retfie                              ;nope, exit the ISR back to the main code

    movlw           0xFF
    xorwf           Direction, f        ;toggle the direction state and save it back
    retfie                              ;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 efficent 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 automically place the program counter
 *  inside of the ISR where we can change directions outisde 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 interupts


    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;

        }
    }


}
```

```
; ********************************************************************
; Lesson 4 - "Analog to Digital"
;
; This shows how to read the A2D converter and display the
; High order parts on the 4 bit LED display.
; The pot on the Low Pin Count Demo board varies the voltage
; coming in on in A0.
;
; The A2D is referenced to the same Vdd as the device, which
; is nominally is 5V.  The A2D returns the ratio of the voltage
; on Pin RA0 to 5V.  The A2D has a resolution of 10 bits, with 1024
; representing 5V and 0 representing 0V.
;
;
; The top four MSbs of the ADC are mirrored onto the LEDs. Rotate the potentiometer
; to change the display.
;
;
; 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

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


    ORG 0                               ;start of code at address 0x0000
Start:
                                        ;Setup main init
     banksel        OSCCON              ;bank1
     movlw          b'00111000'         ;set cpu clock speed
     movwf          OSCCON              ;move contents of the working register into OSCCON

                                        ;Configure the ADC/Potentimator
                                        ;already in bank1
     bsf            TRISA, 4            ;Potentimator is connected to RA4....set as input
     movlw          b'00001101'         ;select RA4 as source of ADC and enable the module (carefull, this ↙
    is actually AN3)
     movwf          ADCON0
     movlw          b'00010000'         ;left justified - Fosc/8 speed - vref is Vdd
     movwf          ADCON1
     banksel        ANSELA              ;bank3
     bsf            ANSELA, 4           ;analog for ADC

                                        ;Configure the LEDs
     banksel        TRISC               ;bank1
     clrf           TRISC               ;make all of PORTC an output
     banksel        LATC                ;select the bank where LATC is
     movlw          b'00001000'         ;start the rotation by setting DS1 ON
     movwf          LATC                ;write contents of the working register to the latch
```

```
MainLoop:
                                    ;Start the ADC
    nop                             ;requried ADC delay of 8uS => (1/(Fosc/4)) = (1/(500KHz/4)) = 8uS
    banksel         ADCON0
    bsf             ADCON0, GO      ;start the ADC
    btfsc           ADCON0, GO      ;this bit will be cleared when the conversion is complete
    goto            $-1             ;keep checking the above line until GO bit is clear

                                    ;Grab Results and write to the LEDs
    swapf           ADRESH, w       ;Get the top 4 MSbs (remember that the ADC result is LEFT justified↙
    !)
    banksel         LATC
    movwf           LATC            ;move into the LEDs
    bra             MainLoop

    end                             ;end code
```

```
/**
 ********************************************************************
 * Lesson 4 - "Analog to Digital"
 *
 * This shows how to read the A2D converter and display the
 * High order parts on the 4 bit LED display.
 * The pot on the Low Pin Count Demo board varies the voltage
 * coming in on in A0.
 *
 * The A2D is referenced to the same Vdd as the device, which
 * is nominally is 5V.  The A2D returns the ratio of the voltage
 * on Pin RA0 to 5V.  The A2D has a resolution of 10 bits, with 1023
 * representing 5V and 0 representing 0V.
 *
 *
 * The top four MSbs of the ADC are mirrored onto the LEDs. Rotate the potentiometer
 * to change the display.
 *
 *  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

                                                    //setup ADC
    TRISAbits.TRISA4 = 1;                           //Potentiamtor is connected to RA4...set as input
    ANSELAbits.ANSA4 = 1;                           //analog
    ADCON0 = 0b00001101;                            //select RA4 as source of ADC and enable the module     ↙
    (AN3)
    ADCON1 = 0b00010000;                            //left justified - FOSC/8 speed - Vref is Vdd

    while (1) {
        __delay_us(5);                              //wait for ADC charging cap to settle
        GO = 1;
        while (GO) continue;                        //wait for conversion to be finished
        LATC = (ADRESH >> 4);                       //grab the top 4 MSbs
    }



}
```

```
; ******************************************************************
; Lesson 5 - "Variable Speed Rotate"
;
; This lesson combines all of the previous lessons to produce a variable speed rotating
; LED display that is proportional to the ADC value. The ADC value and LED rotate
; speed are inversely proportional to each other.
;
; Rotate the POT counterclockwise to see the LEDs shift faster.
;
;
; 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:
                                        ;Setup main init
     banksel        OSCCON              ;bank1
     movlw          b'00111000'         ;set cpu clock speed
     movwf          OSCCON              ;move contents of the working register into OSCCON

                                        ;Configure the ADC/Potentimator
                                        ;already in bank1
     bsf            TRISA, 4            ;Potentimator is connected to RA4....set as input
     movlw          b'00001101'         ;select RA4 as source of ADC and enable the module (carefull, this ↙
    is actually AN3)
     movwf          ADCON0
     movlw          b'00010000'         ;left justified - Fosc/8 speed - vref is Vdd
     movwf          ADCON1
     banksel        ANSELA              ;bank3
     bsf            ANSELA, 4            ;analog for ADC

     ;Configure the LEDs
     banksel        TRISC       ;bank1
     clrf           TRISC               ;make all of PORTC an output
     banksel        LATC                ;bank2
     movlw          b'00001000'         ;start the rotation by setting DS4 ON
     movwf          LATC                ;write contents of the working register to the latch
MainLoop:
     call           A2d                 ;get the ADC result
                                        ;top 8 MSbs are now in the working register (Wreg)
     movwf          Delay2              ;move ADC result into the outer delay loop
```

```
     call           CheckIfZero         ;if ADC result is zero, load in a value of '1' or else the delay
    loop will decrement starting at 255
     call           DelayLoop           ;delay the next LED from turning ON
     call           Rotate              ;rotate the LEDs

     bra            MainLoop            ;do this forever

CheckIfZero:
     movlw          d'0'                ;load wreg with '0'
     xorwf          Delay2, w           ;XOR wreg with the ADC result and save in wreg
     btfss          STATUS, Z           ;if the ADC result is NOT '0', then simply return to MainLoop
     return                             ;return to MainLoop
     movlw          d'1'                ;ADC result IS '0'. Load delay routine with a '1' to avoid
    decrementing a rollover value of 255
     movwf          Delay2              ;move it into the delay location in shared memory (RAM)
     return                             ;return to MainLoop

A2d:
    ;Start the ADC
    nop                                 ;requried ADC delay of 8uS => (1/(Fosc/4)) = (1/(500KHz/4)) = 8uS
    banksel         ADCON0
    bsf             ADCON0, GO          ;start the ADC
    btfsc           ADCON0, GO          ;this bit will be cleared when the conversion is complete
    goto            $-1                 ;keep checking the above line until GO bit is clear
    movf            ADRESH, w           ;Get the top 8 MSbs (remember that the ADC result is LEFT justified
    !)

    return

DelayLoop:
                                        ;Delay amount is determined by the value of the ADC
    decfsz          Delay1,f            ;will always be decrementing 255 here
    goto            DelayLoop           ;The Inner loop takes 3 instructions per loop * 255 loops (required
     delay)
    decfsz          Delay2,f            ;The outer loop takes and additional 3 instructions per lap * X
    loops (X = top 8 MSbs from ADC conversion)
    goto            DelayLoop

    return

Rotate:
    banksel         LATC                ;change to Bank2
    lsrf            LATC                ;logical shift right
    btfsc           STATUS,C            ;did the bit rotate into the carry?
    bsf             LATC,3              ;yes, put it into bit 3.

    return

    end                                 ;end code
```

```
/**
 ******************************************************************
 *  Lesson 5 - "Variable Speed Rotate"
 *
 *  This lesson combines all of the previous lessons to produce a variable speed rotating
 *  LED display that is proportional to the ADC value. The ADC value and LED rotate
 *  speed are inversely proportional to each other.
 *
 *  Rotate the POT counterclockwise to see the LEDs shift faster.
 *
 *  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|-
     *----------------------------------------
     */

unsigned char adc(void);                                //prototype

void main(void) {
    unsigned char delay;


    OSCCON = 0b00111000;                                //500KHz clock speed
    TRISC = 0;                                          //all LED pins are outputs
    LATC = 0;
    LATCbits.LATC3 = 1;                                 //start sequence with DS4 lit

                                                        //setup ADC
    TRISAbits.TRISA4 = 1;                               //Potentiamtor is connected to RA4...set as input
    ANSELAbits.ANSA4 = 1;                               //analog
    ADCON0 = 0b00001101;                                //select RA4 as source of ADC and enable the module ↙
     (AN3)
    ADCON1 = 0b00010000;                                //left justified - FOSC/8 speed - Vref is Vdd

    while (1) {
       delay = adc();                                   //grab the top 8 MSbs
       __delay_ms(5);                                   //delay for AT LEAST 5ms
       while (delay-- != 0)
           __delay_ms(2);                               //decrement the 8 MSbs of the ADC and dealy 2ms for ↙
     each
       LATC >> = 1;                                     //shift to the right by 1 to light up the next LED
       if(STATUSbits.C)                                 //when the last LED is lit, restart the pattern
           LATCbits.LATC3 = 1;
    }

}
```

```
unsigned char adc(void) {
    __delay_us(5);                                  //wait for ADC charging cap to settle
    GO = 1;
    while (GO) continue;                            //wait for conversion to be finished

    return ADRESH;                                  //grab the top 8 MSbs

}
```