ENGR 270 LAB #4 – Interrupts

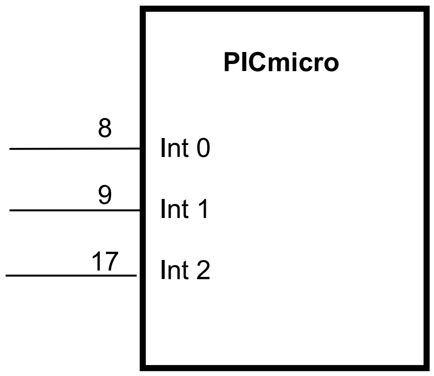
Objective
Utilizing interrupts to handle unscheduled events while the PICmicro is executing the main code.

Related Principles
- ❖ Computer Organization and Design
- ❖ Microprocessors
- ❖ Hardware and Software Interface
- ❖ Digital Design
- ❖ Assembly language

Equipment
- ❖ Windows-based PC with MPLAB Simulation Solutions Software
- ❖ USB hard disk or other removable drives
- ❖ Microchip PICKit programmer
- ❖ EDbot V7.0 Platform

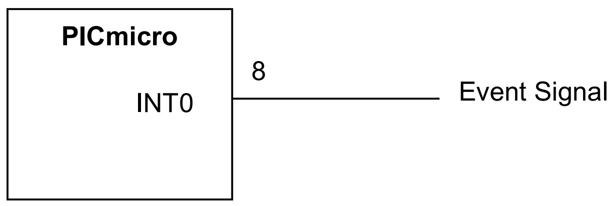
Supplies
- ❖ None

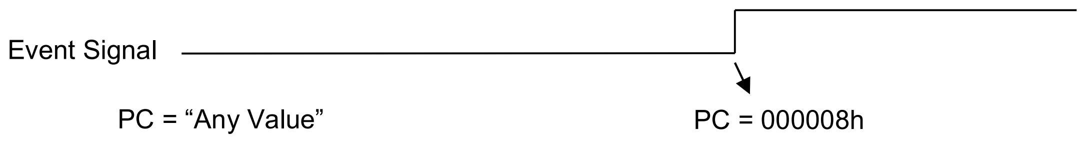
Preparation/Background
In additional to material covered in earlier labs, this lab requires knowledge of PICmicro interrupts handling. It is recommended to review the material in the course text as well as using the PICmicro data sheet as a reference. The remainder of this section provides a brief overview of PICmicro's three external or peripheral interrupt pins and their uses.



The high priority interrupt vector is at 000008h program memory location and the low priority interrupt vector is at 000018h program memory location. Interrupt vector is the location that PC will be set to after an interrupt has occurred and has been acknowledged.

There are three external interrupts available on PICmicro (INT0-Pin 8, INT1-Pin9 and INT2-Pin 17). Below is an example of connecting interrupt INT0 to Event Signal. Anytime, Event Signal goes from low to high which causes a high priority interrupt and sets PC to 000008h.





In general, each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred.
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set.
- Priority bit to select high priority or low priority (INT0 has no priority bit and is always high priority)

The following four SFR registers are used to control interrupt operations:

- RCON Register

Bit 7 Bit0

IPEN	—	—	RI'	TO'	PD'	POR'	BOR'	RCON

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **$\overline{\text{RI}}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed (set by firmware only)
0 = The RESET instruction was executed causing a device Reset
(must be set in software after a Brown-out Reset occurs)

bit 3 **$\overline{\text{TO}}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{\text{PD}}$:** Power-down Detection Flag bit
1 = Set by power-up or by the CLRWDT instruction
0 = Cleared by execution of the SLEEP instruction

bit 1 **$\overline{\text{POR}}$:** Power-on Reset Status bit
1 = A Power-on Reset has not occurred (set by firmware only)
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{\text{BOR}}$:** Brown-out Reset Status bit
1 = A Brown-out Reset has not occurred (set by firmware only)
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

- INTCON Register

Bit 7

GIE/ GIEH	PEIE/ GIEL	TMR0 IE	INT0 IE	RBIE	TMR0 IF	INT0 IF	RB IF	INTCON

bit 7	**GIE/GIEH:** Global Interrupt Enable bit When IPEN = 0: 1 = Enables all unmasked interrupts 0 = Disables all interrupts When IPEN = 1: 1 = Enables all high priority interrupts 0 = Disables all interrupts
bit 6	**PEIE/GIEL:** Peripheral Interrupt Enable bit When IPEN = 0: 1 = Enables all unmasked peripheral interrupts 0 = Disables all peripheral interrupts When IPEN = 1: 1 = Enables all low priority peripheral interrupts 0 = Disables all low priority peripheral interrupts
bit 5	**TMR0IE:** TMR0 Overflow Interrupt Enable bit 1 = Enables the TMR0 overflow interrupt 0 = Disables the TMR0 overflow interrupt
bit 4	**INT0IE:** INT0 External Interrupt Enable bit 1 = Enables the INT0 external interrupt 0 = Disables the INT0 external interrupt
bit 3	**RBIE:** RB Port Change Interrupt Enable bit 1 = Enables the RB port change interrupt 0 = Disables the RB port change interrupt
bit 2	**TMR0IF:** TMR0 Overflow Interrupt Flag bit 1 = TMR0 register has overflowed (must be cleared in software) 0 = TMR0 register did not overflow
bit 1	**INT0IF:** INT0 External Interrupt Flag bit 1 = The INT0 external interrupt occurred (must be cleared in software) 0 = The INT0 external interrupt did not occur
bit 0	**RBIF:** RB Port Change Interrupt Flag bit 1 = At least one of the RB7:RB4 pins changed state (must be cleared in software) 0 = None of the RB7:RB4 pins have changed state **Note:** A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

- INTCON2 Register

Bit 7							Bit0	
RBPU'	INTE DG0	INTE DG1	INTE DG2	—	TMR0 IP	—	RBIP	INTCON2

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 3 **Unimplemented:** Read as '0'

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as '0'

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

- INTCON3

Bit 7							Bit0	
INT2 IP	INT1 IP	—	INT2 IE	INT1 IE	—	INT2 IF	INT1 IF	INTCON3

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **Unimplemented:** Read as '0'

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **Unimplemented:** Read as '0'

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits that enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled. With this setting, the interrupts are compatible with PICmicro mid-range devices. In compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit, which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit, which enables/disables all interrupt sources. All interrupts branch to address 000008h in compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

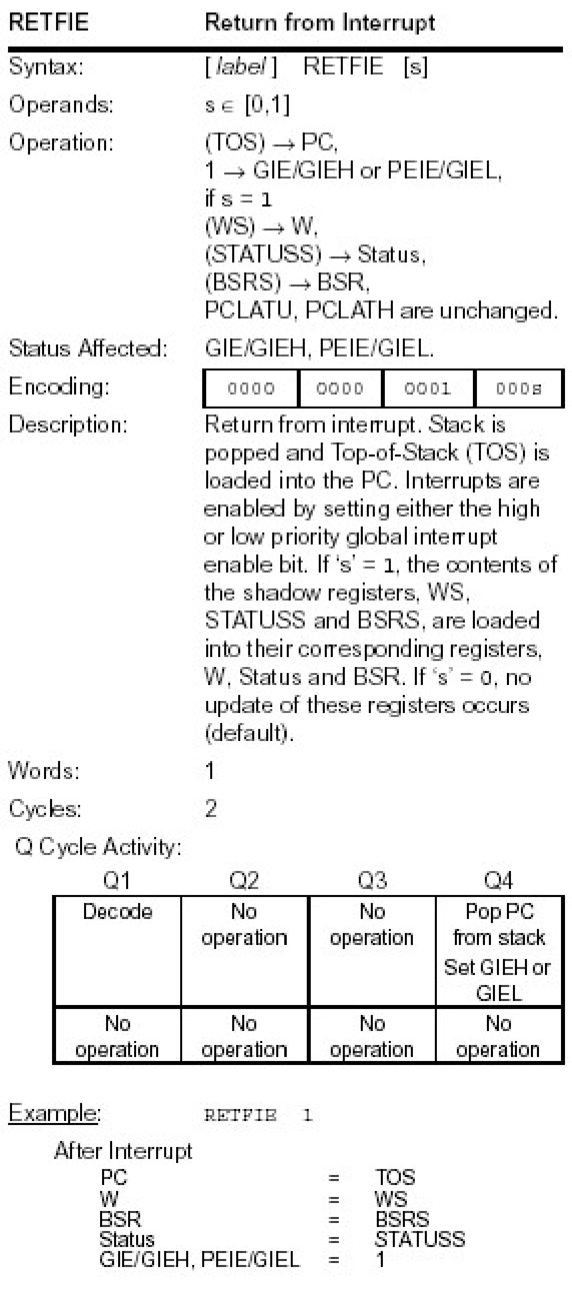
Upon interrupt, the return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts. The “return from interrupt” instruction, RETFIE, exits the interrupt routine and sets the GIE bit (GIEH or GIEL, if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency may be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding enable bit or the GIE bit.

Note: Do not use the MOVFF instruction to modify any of the interrupt control registers while any interrupt is enabled. Doing so may cause erratic microcontroller behavior.

❖ Returning from interrupt handling code
Upon interrupt, the value of PC+2 (pointer to the next instruction) is pushed on the stack. This allows the interrupt handling code to return to the next instruction before interrupt by popping the stack and using the top of stack value as the PC.

The Instruction RETFIE when executed will automatically return the instruction execution back to the next instruction before the interrupt.



RETFIE **Return from Interrupt**

Syntax:	[*label*] RETFIE [s]
Operands:	s ∈ [0,1]
Operation:	(TOS) → PC, 1 → GIE/GIEH or PEIE/GIEL, if s = 1 (WS) → W, (STATUSS) → Status, (BSRS) → BSR, PCLATU, PCLATH are unchanged.
Status Affected:	GIE/GIEH, PEIE/GIEL.
Encoding:	0000 0000 0001 000s
Description:	Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	Pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example: `RETFIE 1`

After Interrupt

```
PC                  =  TOS
W                   =  WS
BSR                 =  BSRS
Status              =  STATUSS
GIE/GIEH, PEIE/GIEL =  1
```

Notes:

- Example – High priority interrupts and returns code.

  Solution:

Address	**Content**
0x008	MVLW 23
0x00A	ADDWF 0x90, 1, 0
0x00C	CLRF 0x89
0x00E	RETFIE
…	
0x126	MVLW 23
0x128	ADDWF 0x90, 1, 0
0x12A	CLRF 0x89

A high Priority Interrupt occurs when instruction at location 0x128 is being executed. Where PC is equal to 0x12A.

❖ NOP Delay Loop Example
The following code generate delay equal to approximately 100x4 cycles:

```
        MOVLW   155     ; start the count
        MOVWF   0x84
Delay:  NOP             ; 1 cycle
        INCF    0x84    ; 1 cycle
        BNC     Delay   ; 2 cycle when jumps to Delay
```

Using f_{osc} = 31.5 Khz, internal clock frequency, means that One clock cycle period is $T_{osc} = 1/f_{osc}$ = 32 usec. Tcyc, instruction cycle, is 4 times the clock frequency which means Tcyc = 4*32 = 128 usec. Therefore, the above NOP delay loop generated a delay equal to 400x128 usec or approximately 51 msec.

❖ Interrupt Usage Example
INT0, Pin 8 (high priority) is connected to DIP switch #3 and INT1, Pin 9 (low or high priority) is connected to DIP switch #2 on EDbot. The following code is written to demonstrate the use of low and high priority interrupts.

Each INT0 occurrence increments the Wreg value by 5 which causes the LED blinking on and off time to increase by 0.5 seconds.

Each INT1 occurrence decrements the Wreg value by 5 which causes the LED blinking on and off time to decrease by by 0.5 seconds.
.

```
;-----------------------------------------------------------------------------------
; FILE: IntrExample
; DESC: Interrupt Example - Demonstrates use of interrupts
; DATE: 5-18-16
; AUTH: Class
; DEVICE: PICmicro (PIC18F1220)
;-----------------------------------------------------------------------------------
   list            p=18F1220                    ; processor type
   radix           hex                          ; default radix for data
   config          WDT=OFF, LVP=OFF, OSC = INTIO2         ; Disable Watchdog timer,  Low V. Prog, and RA6 as a clock

#include           p18f1220.inc                 ; This header file includes address and bit definitions for all SFRs

#define            countID          0x80
#define            countOD          0x81


   org 0x000                                    ; Executes after reset
   GOTO            StartL

   org 0x008                                    ; Executes after high priority interrupt
   GOTO            HPRIO

   org 0x018                                    ; Executes after low priority interrupt
   GOTO            LPRIO

   org 0x20

HPRIO:             ; high priority interrupt
    ADDLW          .5                           ; when interrupt 0 occurs
    BCF            INTCON, INT0IF               ; Clear Interrupt 0
    RETFIE                                      ; Return from interrupt

LPRIO:             ; Low priority interrupt
   BTFSC           INTCON3, INT1IF              ; Check for Interrupt 1
   BRA             Intr1
   RETFIE          ; Return from interrupt

Intr1:             ; take care of Interrupt 1
   ADDLW           0xFB                         ; W←(W-5).  {note: SUBLW .5 will not work}
   BCF             INTCON3, INT1IF               ; Clear interrupt 1 flag
   RETFIE                                       ; Return from interrupt


StartL:                                         ; Initialization code to be executed during reset
   ; Initialize all I/O ports
   CLRF        PORTA                            ; Initialize PORTA
   CLRF        PORTB                            ; Initialize PORTB
   MOVLW       0x7F                             ; Set all A\D Converter Pins as
   MOVWF       ADCON1                           ; digital I/O pins
   MOVLW       0x0D                             ; Value used to initialize data direction
   MOVWF       TRISA                            ; Set Port A direction
   MOVLW       0xC7                             ; Value used to initialize data direction
   MOVWF     TRISB                          ; Set Port B direction
   MOVLW     0x00                           ; clear Wreg

   ; Enable NT0 and INT1
   BSF           INTCON, PEIE               ; enable all peripheral interrupts
   BSF           INTCON, INT0IE             ; enable INT0
   BSF           INTCON3, INT1IE            ; enable INT1
   BCF           INTCON3, INT1IP            ; INT1 is set to low priority
   BSF           RCON, IPEN                 ; enable priority levels on interrupts
   BCF           INTCON, INT0IF              ;flags must be cleared  to allow an interrupt
   BCF           INTCON3, INT1IF             ;
   BSF           INTCON, GIE                ; enable interrupts globally

   MOVLW         .5                         ; Set starting delay to 0.5 seconds

MainL:                                      ;Main loop
   BTG            PORTB,5                   ; LED Toggle
   CALL          Delay
   BRA           MainL

;Function to delay for Wreg x 0.1 seconds
Delay:
   MOVWF     countOD
DelayOL:                                    ; delay Outer loop
   CLRF      countID
DelayIL:                                    ; Delay Inner Loop
   INCF      countID
   BNZ       DelayIL
   DECF      countOD
   BNZ       DelayOL
   RETURN                                   ; end delay function

   end   ; Interrupt Example Program
```

Experiment #1
Use the example code provided to implement a system that start with 0.5 second LED on/off. Each time INT0 occurs, the on/off time increases by one second and each time INT1 occurs, the on/off time decreases by one seconds. The delay should never be less than 0.5 second or more than 25 seconds.

Experiment #2
Write code for EDbot to blink LED on/off every 0.5 second. When INT0 occurs, EDbot should turn clockwise in circles for 2 seconds. When INT1 (Low Priority) occurs, EDbot should turns counter clockwise for 2 seconds.

This experiment requires that you review your high level design (flow chart or pseudo code) and demonstrate your system to the instructor upon completion.

Report Requirements

All reports must be computer printed (formulas and diagrams may be hand drawn) and at minimum include:

For each experiment:

a) Clear problem statement; specify items given and to be found.
b) Specific responses to each question asked in the experiment.
c) Documentation of resulting high level design, disassembled code, system diagram, schematics and any other supporting material.

For the report as a whole

a) Cover sheet with your name, course, lab title, date of completion and your teammates' name.
b) Lessons learned from this lab.
c) A new experiment and expected results which provide additional opportunity to practice the concepts in this lab.