Australia's Global University Faculty of Engineering School of Electrical Engineering and Telecommunications

Chapter 5a: Interrupts

Professor Eliathamby Ambikairajah

Head of School of Electrical Engineering and Telecommunications, UNSW, Sydney

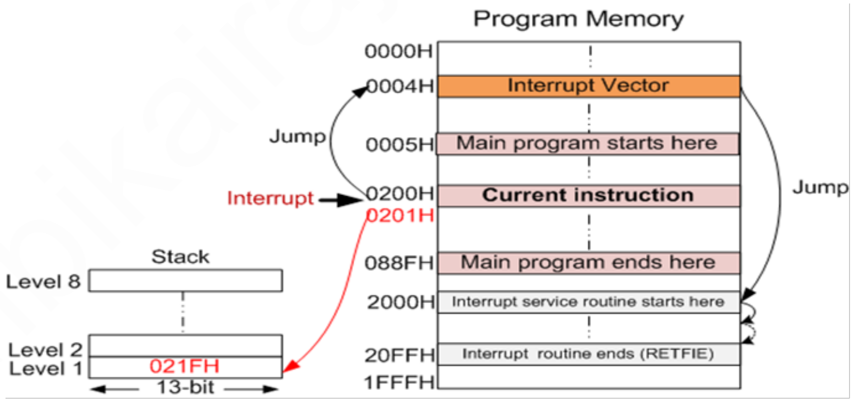
PIC Interrupts

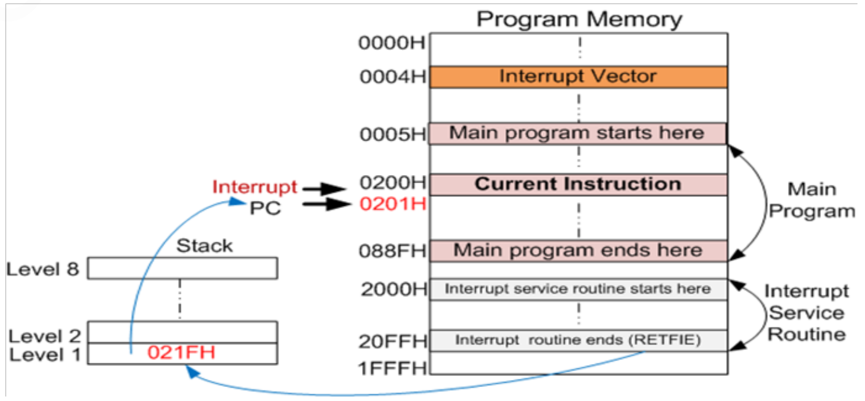
✓An interrupt is an event that forces the microcontroller to execute the current instruction, stops the main program execution and calls an interrupt service routine.

✓On interrupt, the return address of the main program is automatically pushed onto the stack and program control is redirected to the interrupt service routine via the interrupt vector address 0004H. The user must place the interrupt service routine address at this location (0004H).

✓It is important to note that apart from the return address (eg:021FH, see diagram below) no registers (eg: W or STATUS registers) are saved onto the stack. So it is important to save these registers in the RAM, before servicing the interrupt service routine.

✓A special instruction known as RETFIE is used to return from an interrupt service routine to the main program routine. Restore the W and STATUS registers before the RETFIE instruction.





The 16f886 Interrupt structure

✓ The PIC16F886 has multiple interrupt sources:

- External interrupt (RB0/INT) – pin 21
- Timer1 overflow interrupt
- PortB change interrupts
- A/D interrupt
- Fail-safe clock monitor interrupt
- EUSART receive and transmit interrupts
- Ultra low-power wake-up interrupt

- Timer0 interrupt
- Timer 2 Match interrupt
- 2 comparator interrupts
- EEPROM data write interrupt
- Enhanced CCP interrupt
- MSSP interrupt

✓ Each of the above has an individual Interrupt Flag bits and they are set automatically when an interrupt is activated.

✓ The Interrupt Flags must be searched in your program to find out which interrupt flag caused the interrupt.

✓ Once the interrupt has been serviced, the Interrupt Flag bit must be reset by the software before returning from the interrupt service routine to the main program.

EUSART: Enhanced Universal Synchronous/Asynchronous Receiver/Transmitter
CCP: Capture, Compare, PWM; **MSSP**: Master Synchronous Serial Port

Interrupt System Registers

- ✓ The Interrupt Control Register (INTCON) and Peripheral Interrupt Request Register 1 (PIR1) record individual interrupt requests in flag bits
- ✓ The INTCON register also has individual and global interrupt enable bits
- ✓ The following interrupt flags are contained in the INTCON register:

  (a) INT pin interrupt (b) PORTB change interrupts (c) Timer0 overflow interrupt

00H	(INDF)
01H	**TMR0**
03H	**STATUS**
04H	**FSR**
0BH	**INTCON**
0CH	**PIR1**
0DH	**PIR2**
0EH	TMR1L
20H	GPR
Bank 0	

INTCON Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(x)

R/W Read/Writable bit; (0) After rest, bit is cleared; (x) After reset, bit is unknown

GIE – Global Interrupt Enable bit
1 – Enables all unmasked interrupts;
0 – Disables all interrupts

PEIE – Peripheral interrupt enable bit (This is similar to GIE, but controls interrupts enabled by peripherals;**)** **1** – Enables all unmasked peripheral interrupts; **0** – Disables all peripheral interrupts

T0IE – TMR0 overflow interrupt enable bit
1 – Enables TMR0 interrupt; **0**–Disables TMR0 interrupts

INTE– RB0/INT external interrupt enable bit
1 – Enables the RB0/INT external interrupt; **0** – Disables RB0/INT external interrupt

RBIE– PortB change interrupt enable bit
(after configuring as input, PortB pins may cause interrupt for high to low transition or vice versa) **1** – Enables the PortB change interrupt; **0** – Disables the PortB change interrupt

T0IF – TMR0 Overflow interrupt flag bit (This registers TMR0 register overflow) **1** – TMR0 register has overflowed and the bit must be cleared in software; **0** – TMR0 register did not overflow

INTF– INT external interrupt flag bit (registers change of logic state on the RB0/INT pin); **1** - The RB0/INT external interrupt and must be cleared in software; **0** - The RB0/INT external interrupt did not occur

RBIF– PortB change interrupt flag bit; **1** – At least one of the PortB I/O pins changed state (must be cleared in software); 0- None of the PortB I/O pins have changed state

Peripheral Interrupt Request Registers

- ✓ The peripheral interrupt flags are contained in the PIR1 and PIR2 registers (SFR-Data memory Bank 0).
- ✓ The corresponding interrupt enable bits are contained in PIE1 and PIE2 registers (SFR-Data memory Bank 1).

PIC16F886: Special Function Registers (SFRs) - Data memory bank 0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
0CH	**PIR1**	-	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
0DH	**PIR2**	OSFIF	C2IF	C1IF	EEIF	BCLIF	ULPWUIF	-	CCP2IF

PIC16F886: Special Function Registers (SFRs) - Data memory bank 1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
8CH	**PIE1**	-	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
8DH	**PIE2**	OSFIE	C2IE	C1IE	EEIE	BCLIE	ULPWUIE	-	CCP2IE

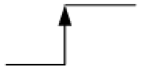
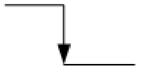
The following interrupt flags are contained in the PIR1 register:

- A/D interrupt
- EUSART receive and transmit interrupts
- Timer0 interrupt
- Timer1 overflow interrupt
- Timer 2 Match interrupt
- Enhanced CCP interrupt
- MSSP interrupt

The following interrupt flags are contained in the PIR2 register:

- Fail-safe clock monitor interrupt
- 2 comparator interrupts
- EEPROM data write interrupt
- Ultra low-power wake-up interrupt
- Enhanced CCP interrupt

Option Register – Interrupt Edge Select Bit

- ✓ RB0/INT interrupt is affected by rising and falling edge (transition) on PORTB, bit 0.
- ✓ This transition can be configured from OPTION register INTEDG bit (bit 6)
- ✓ Setting INTEDG bit to 1 will activate interrupt on a rising edge 
- ✓ Setting INTEDG bit to 0 will activate interrupt on a falling edge 

OPTION Register – Data memory bank 1 and 3

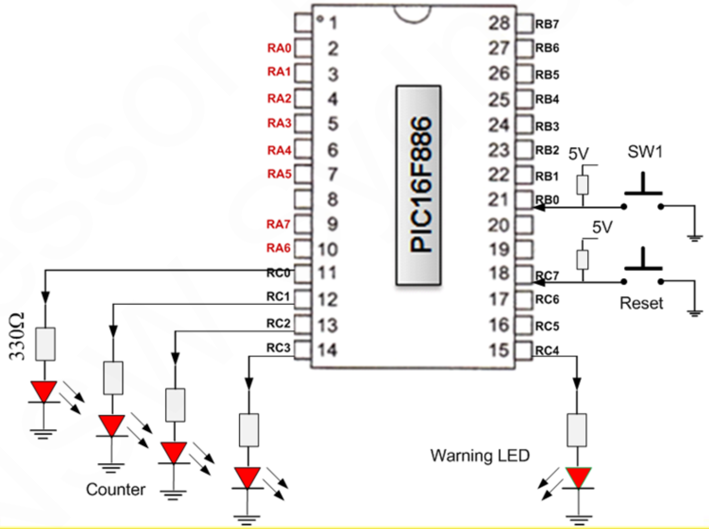
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

R/W Read/Writable bit; (1) After rest, bit is set;

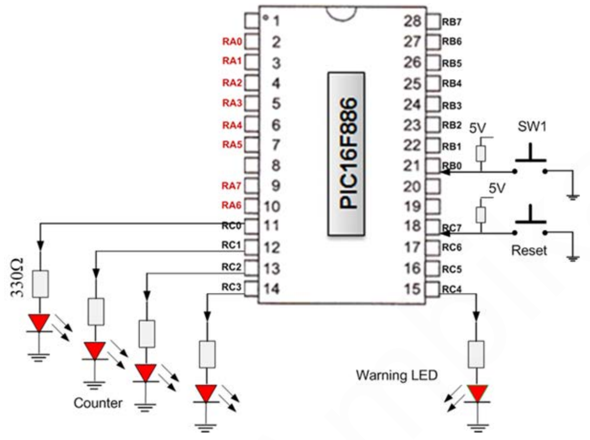
INTEDG – Interrupt Edge Select bit
1 – Interrupt on rising edge of INT pin;
0 – Interrupt on falling edge of INT pin

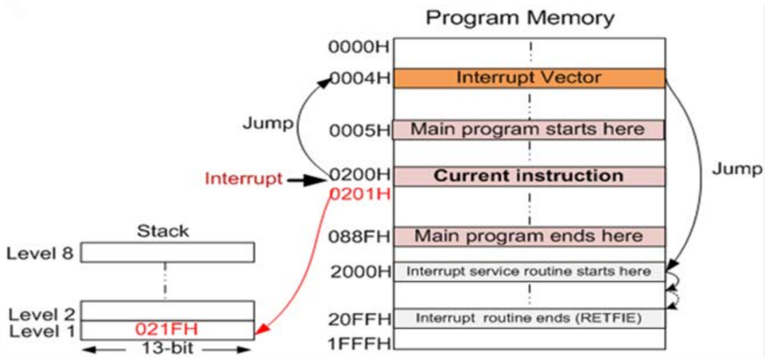
Example: Hardware interrupt

- Write an assembly language program that counts the number of people passing through a door.
  - You can assume that only one person can go through the door at any given time.
  - When the total number of people pass through the door reaches a value of 9 , the warning LED is lit indicating the door is locked and the switch (SW1) used to detect the people passing through the door is disabled (any further press will not be detected).
  - The number of people going through the door must be displayed using a set of 4 LEDs or using a 7-segment display.
  - When the reset button (reset) is pressed, the counter is cleared and the switch (SW1) is enabled.
  - Note that the switch SW1 is connected to RB0 (PortB) as an **interrupt**.
  - Note that you are not required to include any switch de-bouncing routine.



Programming with Single Interrupt





```
;Programming with single interrupt
w_temp       equu         H'20'
status_temp equ          H'21'
counter      equ          H'22'
             org          H'00'
             goto         main
             org          H'04'
             goto         int_routine  ;go to Interrupt service routine
;Initialisation
Main         BANKSEL      ANSEL        ;select memory bank containing ANSEL register
             clrf         ANSEL        ;set PORTC to digital by clearing ANSEL register
             clrf         ANSELH       ;set PORTB to digital
             BANKSEL      TRISC        ;select memory bank containing TRISC register
             movlw        B'10000000'
             movwf        TRISC        ;make PC0 to PC6 output and PC7 as input
             movlw        B'00000001'
             movwf        TRISB        ;set PB0 as input ; PB1 to PB7 as output (not used here)
             BANKSEL      PORTC        ;select memory bank containing PORTC registers
             clrf         PORTC        ;turn off all LEDs
             clrf         counter      ;clear the counter
```

Enable External Interrupt

OPTION Register – Data memory bank 1 and 3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

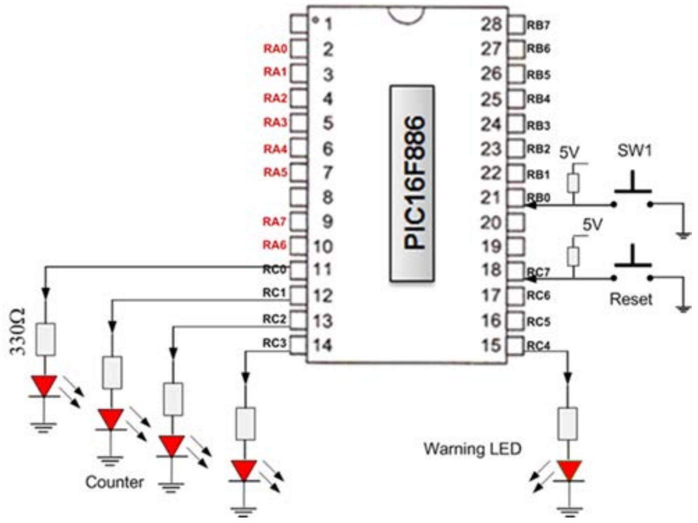
R/W Read/Writable bit; (1) After rest, bit is set;

INTCON Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(x)

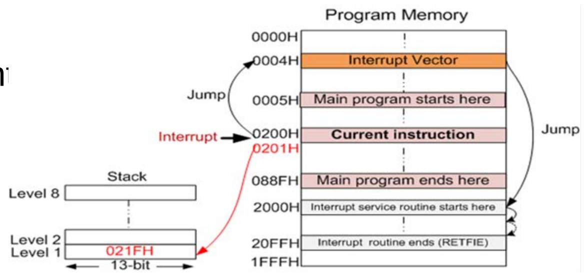
R/W Read/Writable bit; (0) After rest, bit is cleared; (x) After reset, bit is unknown

```
;Enable external interrupt
            BANKSEL  OPTION_REG           ; select Option register  to configure rising or  falling transition
            bcf      OPTION_REG,6 ;select interrupt to trigger on falling edge (0-falling edge, 1-rising edge)
            BANKSEL  INTCON               ;Select memory bank containing INTCON register
            bsf      INTCON,4  ;External interrupt (RB0) enable; i.e  INTE = 1 to enable the external interrupt
            bcf      INTCON,1   ;clear the flag bit (INTF=0); 0 - The RB0/INT external interrupt did not occur
            bsf      INTCON,7             ;Global interrupt enable (GIE) is set to 1 for enabling the interrupts
;
wait_loop   movf     counter,w   ;Move the content of counter to W register
            xorlw    H'09'       ;check if the counter content is D'09'
            btfss    STATUS,Z    ;skip next instruction if z=1
            goto     wait_loop
            clrf     PORTC       ; turn off all lights
            bsf      PORTC,4     ; turn on the warning light
            bcf      INTCON,4    ;disable interrupt (INTE =0)

reset_loop  btfsc    PORTC,7     ;test bit 7 to see if key pressed;
                                 ; branch if PC7 is 0 ( pressed)
            goto     reset_loop
            clrf     counter
            bcf      PORTC,4
            bsf      INTCON,4    ; External interrupt (RB0) enable
            goto     wait_loop
```



Interrupt Service Routine

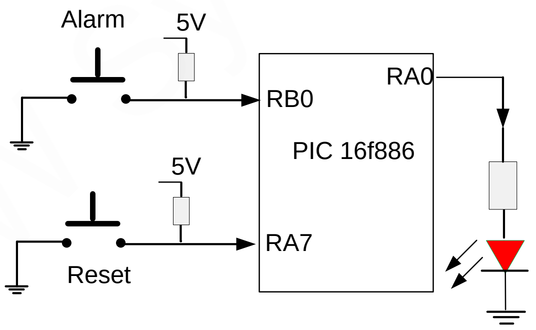
```
;Interrupt service routine  (first save all registers)
Int_routine   movwf      w_temp       ;save current contents of w register in a temporary location ‘w_temp’
              movf       STATUS,W     ;move the contents of the  status register into W register
              movwf      status_temp  ;save the contents of  STATUS register in a temporary location ‘status_temp’

;main function
              incf       counter,1    ; the counter is incremented when there is an in
              movlw      counter,w    ;Move the content of counter to W register
              andwf      H’0F’        ;Most significant 4 bits are made zero
              movwf      PORTC        ;counter contents displayed

;Restore all registers
              movf       status_temp,W ;retrieve the STATUS register content saved in temporary location
              movwf      STATUS       ; put it back in the STATUS register

;it is important that  any further  instructions , before returning from the service routine should not affect the STATUS
register. So we use swapf and movwf instructions as they do not affect the STATUS register (see instruction set sheet)

              swapf      w_temp,1     ;the upper and lower nibbles are exchanged in w-temp and result placed in w-temp
              swapf      w_temp,0     ;the upper and lower nibbles are exchanged in w-temp and result placed in W reg.

;Enable (reset the interrupt flag) the global interrupt before returning to the main program. If this is not carried out, the
microcontroller immediately cause another interrupt to occur when the it executes RETFIE instruction.
              bcf         INTCON,1    ;clear the interrupt flag  (INTF=0) (ready for the next interrupt )
              retfie                  ;return from the interrupt service routine
;
              END
```



Laboratory Activities

Activity 7: Lift control System

- The PIC16F886 microcontroller is used to control the operation of the lift in the Elec Eng building. If a passenger in the lift presses the alarm switch, the LED on the control panel flashes on and off at a frequency of 2Hz until the operator presses a reset switch (see diagram below).
- Write an assembly language program for the lift control system, that uses an interrupt service routine.
- The Interrupt Service Routine should protect the contents of the working register, makes use of a 0.25 second time delay subroutine (you must write this routine) and controls an LED attached to bit 0 of PORT A and monitors the reset switch attached to PORTA bit 7.
- Explain why it is better to have the alarm switch cause an interrupt rather than the main program to poll the alarm switch to see whether it has been pressed or not?



ELEC2117: References

1. Designing Embedded Systems with PIC Microcontrollers – Tim Wilmshurst, Elsevier, 2010
2. PIC Microcontrollers –Free online book – mikroElektronika ; http://www.mikroe.com/products/view/11/book-pic-microcontrollers/
3. PIC 16F886 Data Sheet (2007), Microchip Technology; www.microchip.com