PIC18 Interrupt

Hsiao-Lung Chan
Dept Electrical Engineering
Chang Gung University, Taiwan
chanhl@mail.cgu.edu.tw

Objectives

- Difference between polling and interrupt
- Interrupt handling
- Interrupts in P18
- Timer interrupts
- External hardware interrupts
- Serial communication interrupts
- Interrupt priority

Interrupts vs. polling

- Polling
  - Continuously monitor the status of a given device
  - Waste much of microcontroller's time, e.g.

```
AGAIN   BTFSS    INTCON, TMR0IF    ; monitor Timer0 interrupt flag
        BRA      AGAIN
```

- Interrupt
  - Upon receiving an interrupt signal, microcontroller stops current instruction stream and serves the device

Interrupt service routine (ISR)

- When an interrupt is invoked, the microcontroller runs the ISR
- Interrupt vector table for PIC18 (ROM location)

Power-on reset	0x0000
High priority interrupt	0x0008
Low priority interrupt	0x0018

Steps in executing an interrupt

- Finish the executing instruction, and save the address of the next instruction (program counter) on the stack
- Jump to a fixed location by interrupt vector table which directs the address of ISR
- Execute the ISR until reach RETFIE (return from interrupt exit)
- Get PC from stack and return to the place where it was interrupted

Interrupt programming

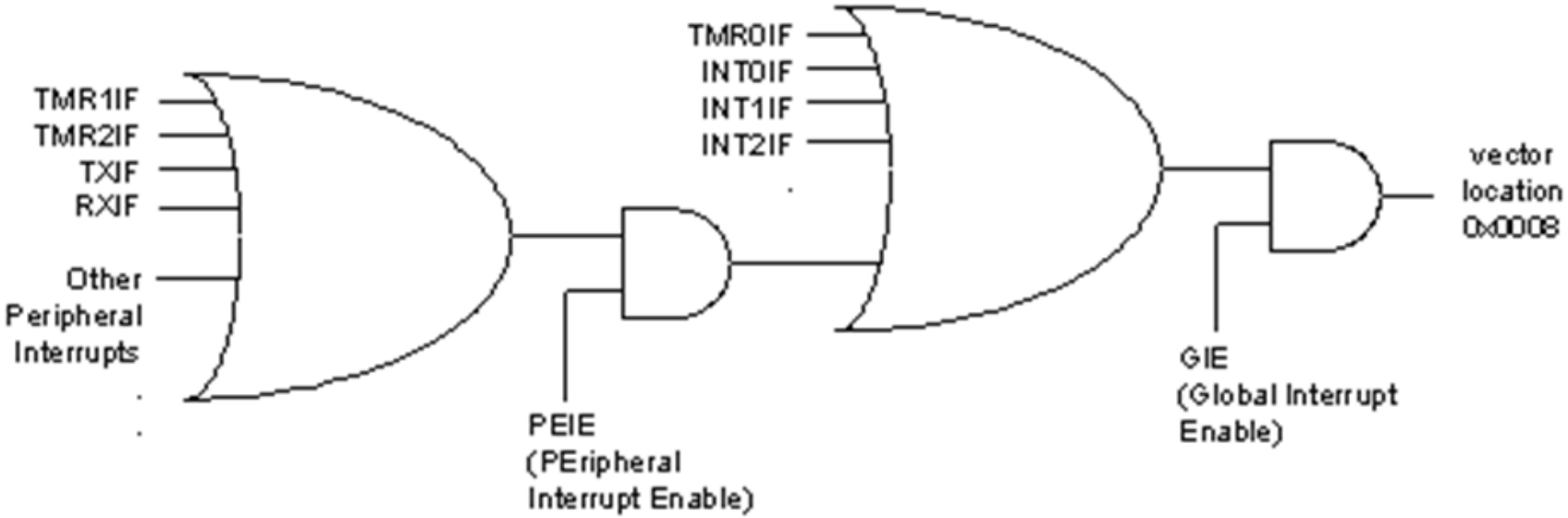
```
        org     0
        goto    main

        org     ox08
        ...             ; interrupt service for high priority interrupts
        retfie
        org     0x18
        ...             ; interrupt service for low priority interrupts
        retfie
main
        ...             ; main program
        end
```

Sources of interrupts in the PIC18

- Timers 0, 1, 2, 3
- Pins RB0, RB1, RB2 for external hardware interrupts INT0, INT1, INT2
- PORTB-Change interrupt (any one of the upper four Port B pins)
- Serial communication's USART interrupts: receive and transmit, respectively
- ADC (analog-to-digital converter)
- CCP (compare capture pulse-width-modulation)

Enable and disable an interrupt



(Figure 11-2)

INTCON (interrupt control) register

GIE (Global Interrupt Enable)
GIE = 0 Disables all interrupts. If GIE = 0, no interrupt is acknowledged, even if they are enabled individually.
If GIE = 1, interrupts are allowed to happen. Each interrupt source is enabled by setting the corresponding interrupt enable bit.

TMR0IE Timer0 interrupt enable
= 0 Disables Timer0 overflow interrupt
= 1 Enables Timer0 overflow interrupt

INT0IE Enables or disables external interrupt 0
= 0 Disables external interrupt 0
= 1 Enables external interrupt 0

These bits, along with the GIE, must be set high for an interrupt to be responded to. Upon activation of the interrupt, the GIE bit is cleared by the PIC18 itself to make sure another interrupt cannot interrupt the microcontroller while it is servicing the current one. At the end of the ISR, the RETFIE instruction will make GIE = 1 to allow another interrupt to come in.

PEIE (PEripheral Interrupt Enable)
For many of the peripherals, such as Timers 1, 2, .. and the serial port, we must enable this bit in addition to the GIE bit. (See Figure 11-2.)

(Figure 11-3)

Programming timer interrupts

- Programming ISR
- Enable interrupt
- Recognize interrupt

Table 11-2: Timer Interrupt Flag Bits and Associated Registers

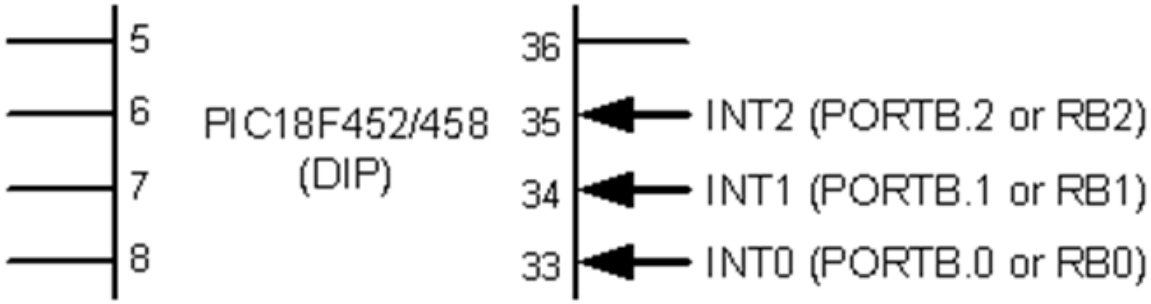
Interrupt	Flag Bit	Register	Enable Bit	Register
Timer0	TMR0IF	INTCON	TMR0IE	INTCON
Timer1	TMR1IF	PIR1	TMR1IE	PIE1
Timer2	TMR2IF	PIR1	TMR2IE	PIE1
Timer3	TMR3IF	PIR3	TMR3IE	PIE2

Examples for Timer interrupt

- Assembly programming
  - Generate a square wave based on an interrupt (Program 11-1)
  - Programming two interrupts (Program 11-3)
- C programming
  - Programming two timer interrupts (Program 11-2C)

Programming external hardware interrupt

- Upon power-on reset, positive edge-triggered interrupts



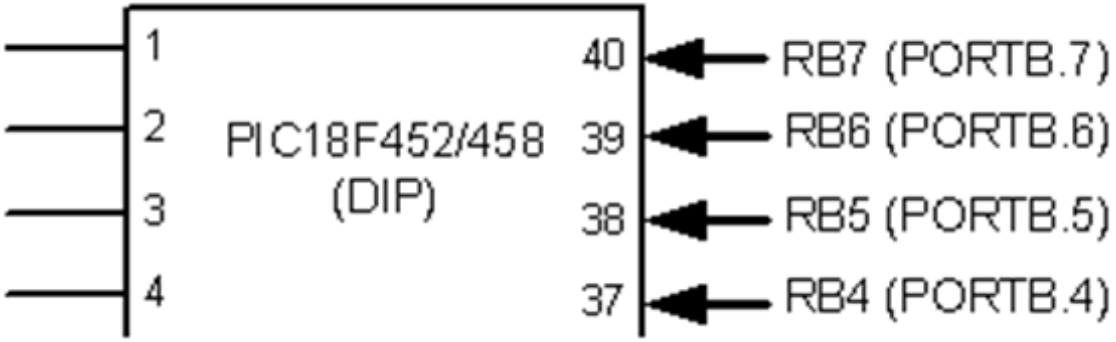
(Figure 11-7)

Table 11-3: Hardware Interrupt Flag Bits and Associated Registers

Interrupt (Pin)	Flag bit	Register	Enable bit	Register
INT0 (RB0)	INT0IF	INTCON	INT0IE	INTCON
INT1 (RB1)	INT1IF	INTCON3	INT1IE	INTCON3
INT2 (RB2)	INT2IF	INTCON3	INT2IE	INTCON3

PORTB-change interrupt

- Change status from HIGH to LOW, or LOW to HIGH
- RBIF (interrupt flag), RBIE (interrupt enable) in INTCON

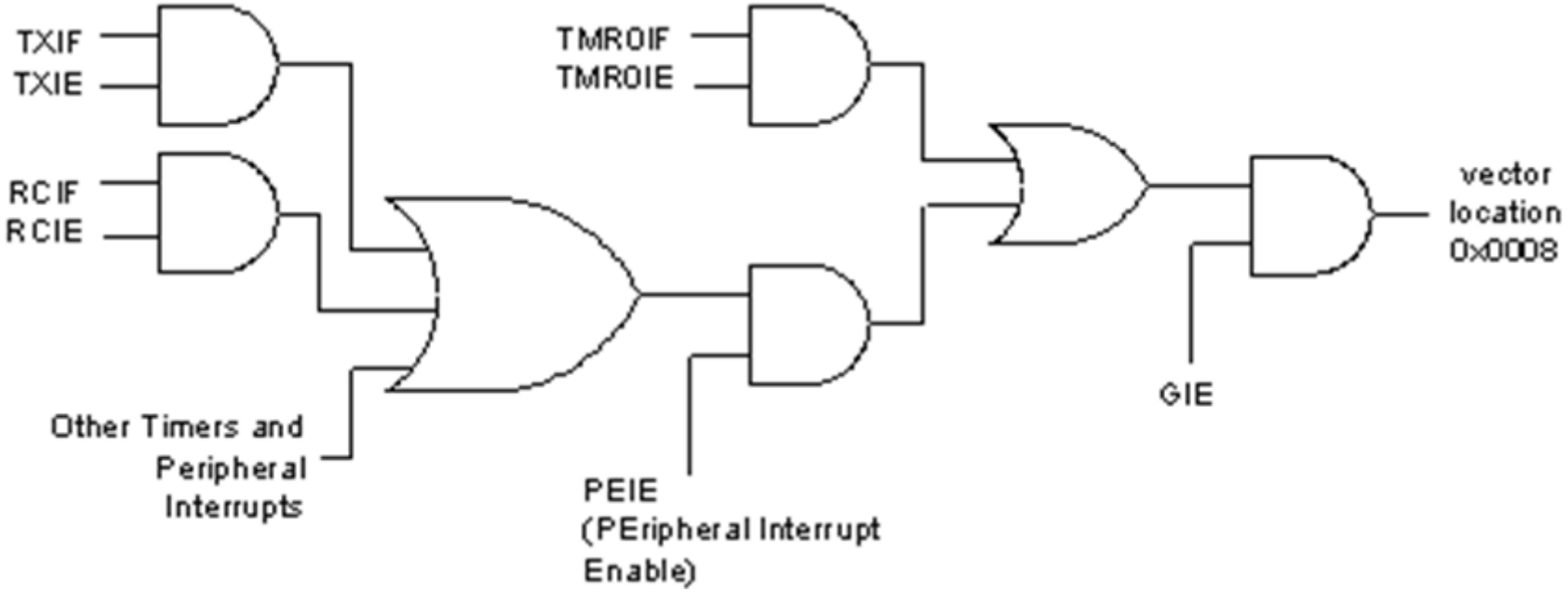


(Figure 11-15)

Programming serial communication interrupts

Table 11-4: Serial Port Interrupt Flag Bits and their Associated Registers

Interrupt	Flag bit	Register	Enable bit	Register
TXIF (Transmit)	TXIF	PIR1	TXIE	PIE1
RCIF (Receive)	RCIF	PIR1	RCIE	PIE1



Examples for interrupts

- INT0 interrupt (Program 11-4)
- Serial communication interrupt (Program 11-7)

Interrupt priority

- Interrupt vector table for PIC18
  - Power-on-reset 0x0000
  - High-priority interrupt 0x0008 (Default on power-on reset)
  - Low-priority interrupt 0x0018 (Selected with IP bit)

Fast context saving in task switching

- Context saving
  - High-priority interrupt
    - PIC18 automatically save WREG, BSR, STATUS in shadow register when a high-priority interrupt is activated.
    - Restore the original content by RETFIE 0x01 (fast context switching)
  - Low-priority interrupt
    - No automatic save, so must save these 3 register at the beginning of ISR

Registers for interrupt flag, enable and priority

Table 11-6: Interrupt Flag Bits for PIC18 Timers

Interrupt	Flag bit (Register)	Enable bit (Register)	Priority (Register)
Timer0	TMR0IF (INTCON)	TMR0IE (INTCON)	TMR0IP (INTCON2)
Timer1	TMR1IF (PIR1)	TMR1IE (PIE1)	TMR1IP (IPR1)
Timer2	TMR2IF (PIR1)	TMR2IE (PIE1)	TMR2IP (IPR1)
Timer3	TMR3IF (PIR3)	TMR3IE (PIE2)	TMR3IP (IPR2)
INT1	INT1IF (PIR1)	INT1IE (PIE1)	INT1IP (INTCON3)
INT2	INT2IF (PIR1)	INT2IE (PIE1)	INT2IP (INTCON)
TXIF	TXIF (PIR1)	TXIE (PIE1)	TXIP (IPR1)
RCIF	RCIF (PIR1)	RCIE (PIE1)	RCIP (IPR1)
RB_INT	RBIF (INTCON)	RBIE (INTCON)	RBIP (INTCON2)

Note: INT0 has only the high-level priority.

0: low priority
1: high priority

Examples for interrupt priority

- Timer0 and Timer 1 interrupts (Program 11-10)
- Two timers and serial communication interrupts (Program 11-11C)

Reference

- M.A. Mazidi, R.D. Mckinlay, D Causey, PIC Microcontroller and Embedded Systems Using Assembly and C for PIC18, Pearson Education Inc., 2008.
- Han-Way Huang, PIC Microcontroller: An Introduction to Software and Hardware Interfacing, Thomson Delmar Learning, 2005.