MECE336– Microprocessors I

Interrupts

Associate Prof. Dr. Klaus Schmidt

Department of Mechatronics Engineering – Çankaya University

Compulsory Course in Mechatronics Engineering
Credits (3/2/4)

Course Webpage: http://MECE336.cankaya.edu.tr

Interrupts: Explanation

Up to Now

- Programs are written in a precise and predictable fashion
- Instructions are followed in a clear order and the update of the program counter is well-defined at each step of the program

Interrupt Idea

- Function of an interrupt is to alert the CPU that some significant event has happened
- Interrupt can occur at any time
- Effect of an interrupt is generally to stop the CPU from doing its current task and to force it to respond to the interrupt cause
- Interrupt disturbs predictable execution of a program
- There can be different interrupt sources on a microcontroller

Interrupts: Sources on PIC16F84A

External Interrupt

- This is the only external interrupt input on PIC16F84A. It shares a pin with Port B, bit 0. It is edge triggered.

Timer Overflow

- This is an interrupt caused by the `Timer0` module. It occurs when the timers 8-bit counter overflows.

`Port B` interrupt on change

- This interrupt occurs when a change is detected on any of the higher four bits of Port B.

EEPROM write complete

- This interrupt occurs when a write instruction to the EEPROM memory is completed (see later lectures).

Interrupts: INTCON Register

Special Function Register: INTCON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

Explanation of Flags

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE:** EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT 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

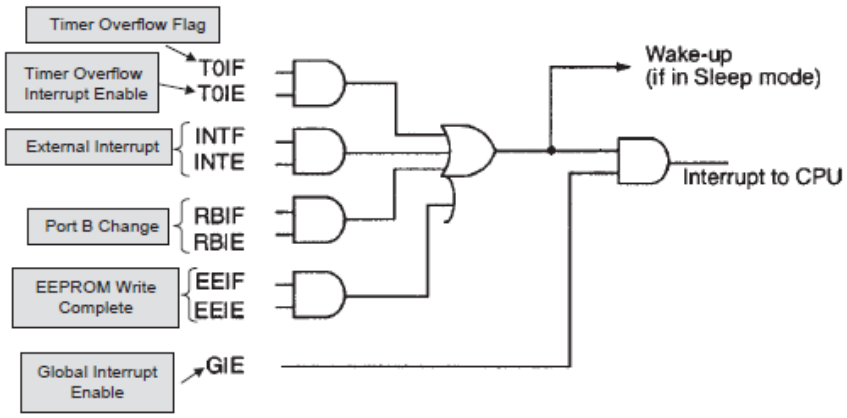
bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT 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

Interrupts: Structure

Interrupt Logic



Global Interrupt Enable

- `GIE` is 1 if any interrupt should be enabled
  $\rightarrow$ all interrupts are disabled if `GIE` is 0

Interrupts: Structure

Timer Interrupt

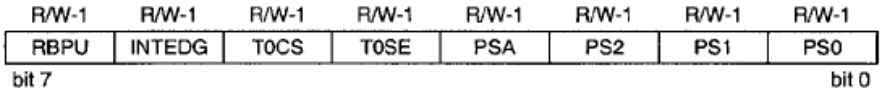
- `Timer0`: `TOIF` indicates if timer overflow happens; `TOIE` is 1 if `Timer0` interrupt is enabled
  $\rightarrow$ `Timer0` interrupt is not used if `TOIE` is 0

Example

Interrupts: Explanation

External Interrupt at INT pin (PORTB)

- INT: INTF indicates if external interrupt happens; INTE is 1 if external interrupt is enabled
  → external interrupt is not used if INTE is 0
- The external interrupt is edge triggered. The type of the edge is controlled by the INTEDG bit of the OPTION register.



  - Rising edge: INTEDG is 1
  - Falling edge: INTEDG is 0

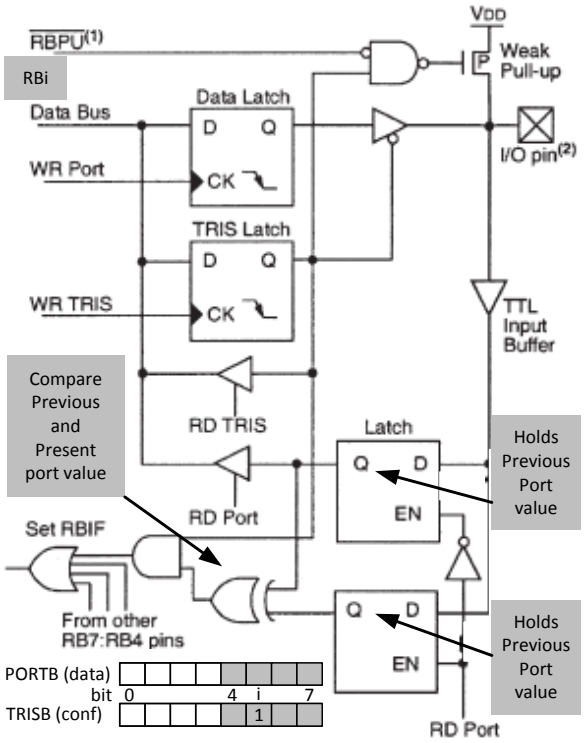
Interrupts: Explanation

External Interrupt at INT pin (PORTB): Examples

Interrupts: Explanation

PORT B **Interrupt on Change**

- Recall additional functionality of pins RB4 to RB7 of PORTB: Data value from previous read operation is stored in second latch
  $\Rightarrow$ An interrupt can be generated if the data value changes
- RBIF indicates if PORTB pin change happens; RBIE is 1 if PORT B pin change interrupt is enabled
  $\rightarrow$ interrupt is not used if RBIE is 0



Interrupts: Explanation

PORT B **Interrupt on Change: Examples**

Interrupts: Configuration Examples

Disable Interrupts

Enable Timer Interrupt

Enable External Interrupt and Interrupt on Change

Interrupts: CPU Response to an Interrupt

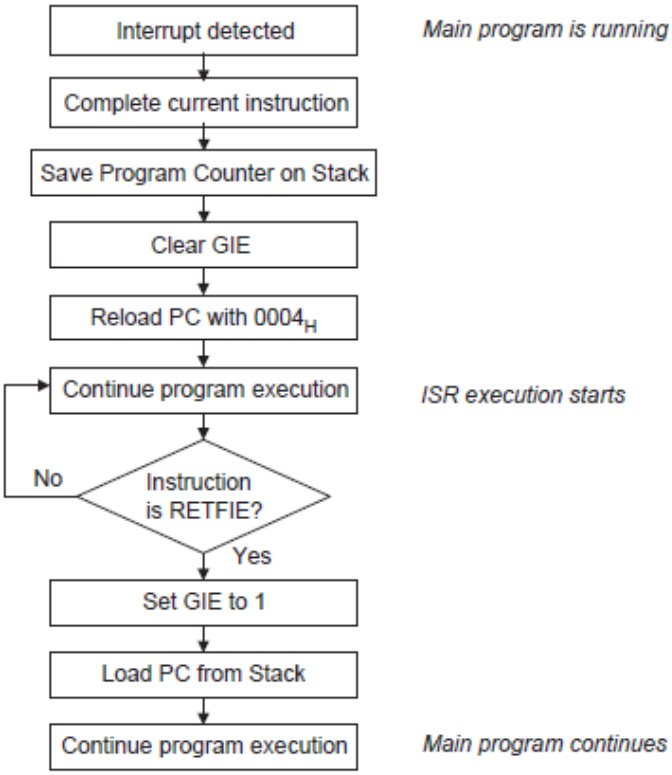
Assumption

- `GIE` is 1 (enabled) and the interrupt signal is detected by the CPU

Stepwise Procedure

- CPU completes the current instruction and saves the value of the program counter to the stack
- CPU clears `GIE` to avoid further interrupts
- The program counter is loaded with the program memory location 0004. The program continues with the *interrupt subroutine* from that location after the interrupt
- The interrupt subroutine ends with the `retfie` instruction
- After `retfie`, the CPU sets `GIE` to 1 and loads the program counter with the top of the stack
- The program continues from that program memory location

Interrupts: Explanation



Interrupts: General Program Structure

Program with Interrupt Subroutine Call

Interrupt Subroutine: General Idea

Considerations

- The interrupt subroutine is called from the program memory location 0004 using a goto instruction
```
org   0x04
goto  interrupt_subroutine
```
- To disable further interrupts during the interrupt subroutine, the interrupt enable flag should be reset
`bcf   INTCON,INTE;` (for external interrupt)
- The interrupt flag should be reset at the end of the subroutine
`bcf   INTCON,INTF;` (for external interrupt)
- The content of the W register and STATUS register should be saved
```
movwf   TEMP_W;
swapf   STATUS,0;
movwf   TEMP_S;
```


Interrupt Subroutine: swapf Instruction

Instruction

- `swapf  f,d`: swap the nibbles of the content in file register `f`. Write the result to
  - Working register W if d is 0
  - File register f if d is 1

Example

Interrupt Subroutine: General Structure

Interrupt Subroutine that Protects `W` and `STATUS`

Interrupt Subroutine: General Structure

Note: Using `movf` changes the zero flag in the `STATUS` register, whereas `swapf` does not change the `STATUS` register
⇒ Use `swapf` to save register content in interrupt subroutine

Interrupt Subroutine: Example

Task

Write a program using an interrupt subroutine such that a counter is incremented whenever a falling edge is detected at the `RB0/INT` pin.

Considerations

Interrupt Subroutine: Example

Program

Interrupt Subroutine: Example

Program

Multiple Interrupts: Explanation

Situation

- Consider that multiple different interrupts can occur in a program

Procedure

- An interrupt is detected by the CPU (the CPU does not know which interrupt occurred)
- Program counter is moved to the stack and is then reset to 0004
- The program must check which interrupt occurred by looking at possible interrupt flags `TOIF`, `INTF`, `RBIF` in the `INTCON` register
- The program calls a different subroutine depending on which interrupt flag is 1
- The program counter is set to the value at the top of the stack after the `retfie` instruction

Multiple Interrupts: Example Program

Task

Multiple Interrupts: Example Program

Program

Multiple Interrupts: Example Program

Program