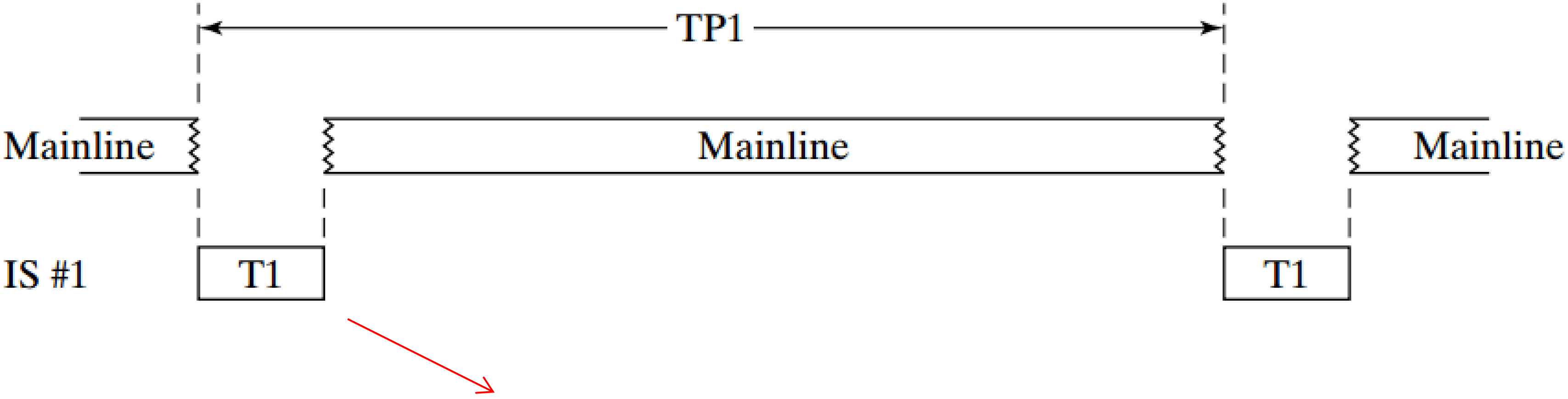
INTERRUPTS

Interrupts

An interrupt is an asynchronous signal indicating the need for attention.

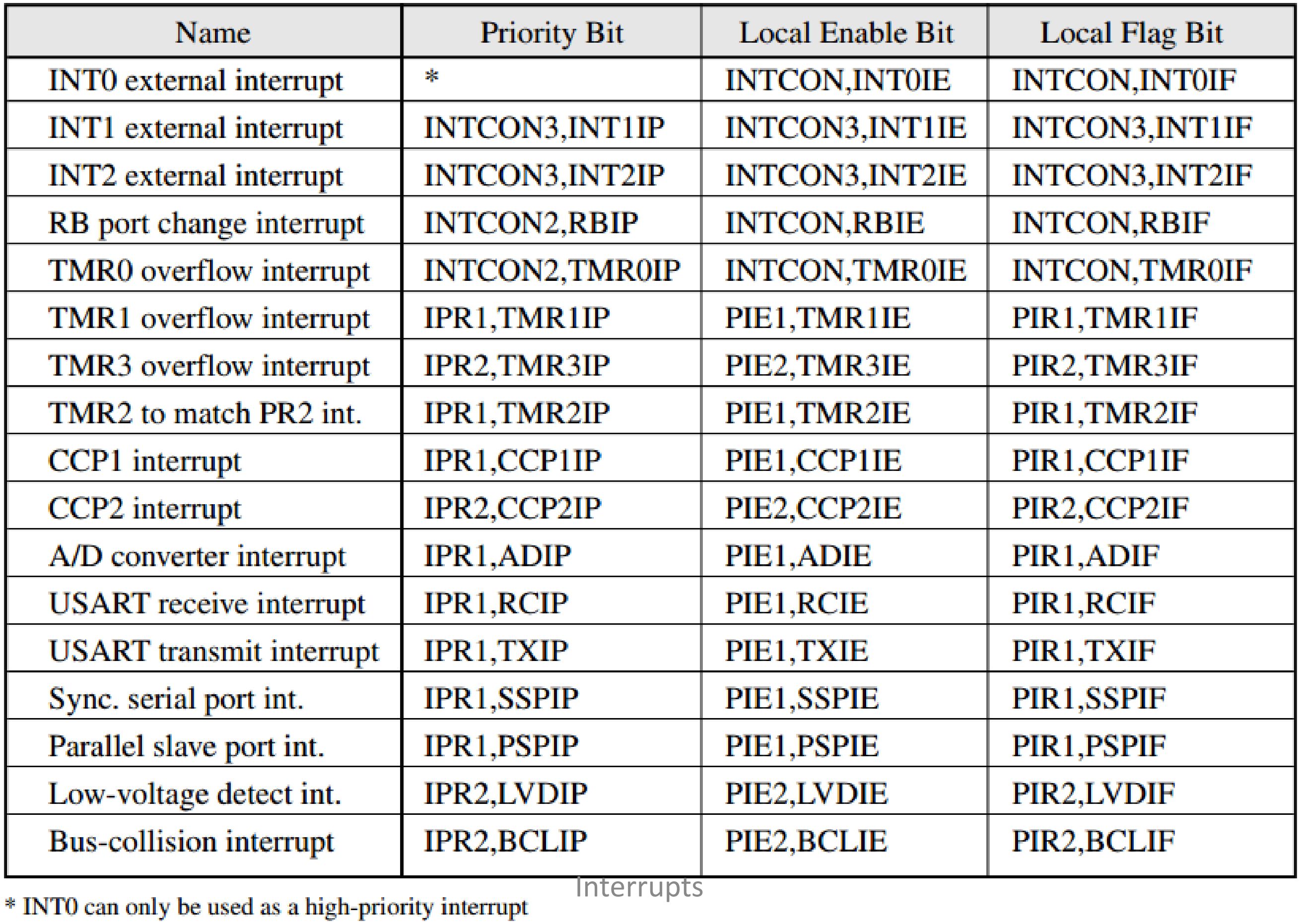
- To handle tasks that require a fast response



Interrupt Service Routine (ISR)

Interrupt Priority

- By default all the interrupts sources are set with high-priority level
- The Interrupt-Priority (IP) bit for each interrupt source to be assigned to the low-priority must be cleared.
- 17 classes of interrupts:



Name	Priority Bit	Local Enable Bit	Local Flag Bit
INT0 external interrupt	*	INTCON,INT0IE	INTCON,INT0IF
INT1 external interrupt	INTCON3,INT1IP	INTCON3,INT1IE	INTCON3,INT1IF
INT2 external interrupt	INTCON3,INT2IP	INTCON3,INT2IE	INTCON3,INT2IF
RB port change interrupt	INTCON2,RBIP	INTCON,RBIE	INTCON,RBIF
TMR0 overflow interrupt	INTCON2,TMR0IP	INTCON,TMR0IE	INTCON,TMR0IF
TMR1 overflow interrupt	IPR1,TMR1IP	PIE1,TMR1IE	PIR1,TMR1IF
TMR3 overflow interrupt	IPR2,TMR3IP	PIE2,TMR3IE	PIR2,TMR3IF
TMR2 to match PR2 int.	IPR1,TMR2IP	PIE1,TMR2IE	PIR1,TMR2IF
CCP1 interrupt	IPR1,CCP1IP	PIE1,CCP1IE	PIR1,CCP1IF
CCP2 interrupt	IPR2,CCP2IP	PIE2,CCP2IE	PIR2,CCP2IF
A/D converter interrupt	IPR1,ADIP	PIE1,ADIE	PIR1,ADIF
USART receive interrupt	IPR1,RCIP	PIE1,RCIE	PIR1,RCIF
USART transmit interrupt	IPR1,TXIP	PIE1,TXIE	PIR1,TXIF
Sync. serial port int.	IPR1,SSPIP	PIE1,SSPIE	PIR1,SSPIF
Parallel slave port int.	IPR1,PSPIP	PIE1,PSPIE	PIR1,PSPIF
Low-voltage detect int.	IPR2,LVDIP	PIE2,LVDIE	PIR2,LVDIF
Bus-collision interrupt	IPR2,BCLIP	PIE2,BCLIE	PIR2,BCLIF

* INT0 can only be used as a high-priority interrupt

Low Priority and High Priority Interrupts

- **Low Priority Interrupt**
  - Control goes to Low Priority interrupt service routine (LPISR)
  - If another low priority interrupt event occurs while the LPISR is running, the new event is lost.
  - If a high priority interrupt event occurs, the LPISR is stopped, then
    - Control goes to HPISR
    - When HPISR has finished, control goes back to LPISR.
- **High Priority Interrupt**
  - Control goes to High Priority interrupt service routine (HPISR)
  - If another interrupt event occurs (low or high priority), the event is lost.
- For external interrupts
  - INT0 is always a high priority interrupt
  - INT1 and INT2 can be set as either high or low priority interrupt

Introducing interrupts in ASM file

```
;;;;;;; Vectors ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        org  0x0000          ;Reset vector
        nop
        goto  Mainline

        org  0x0008          ;High priority interrupt vector
        goto  $

        org  0x0018          ;Low priority interrupt vector
        goto  $
```

```
;;;;;;; Vectors ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        org  0x0000          ;Reset vector
        nop
        goto  Mainline

        org  0x0008          ;High priority interrupt vector
        goto  NAME_OF_HPISR

        org  0x0018          ;Low priority interrupt vector
        goto  NAME_OF_LPISR
```

Low-priority interrupts

In the low priority interrupt service routine, when an interrupt occurs, the following sequence of events takes place automatically:

- The CPU completes the execution of its present mainline instruction
- The **GIEL** bit is cleared, disabling further low-priority interrupts
- The program counter (PC) is stacked
- The program counter is set to 0x0018, the low-priority interrupt vector.

Furthermore, the content of **WREG** and **STATUS** register should set aside at the beginning of the interrupt service routine, and restored at the end using the correct order.

saved

```
movff   STATUS, STATUS_TEMP
movf     W, WREG_TEMP
```

restored

```
movf    WREG_TEMP, W
movff   STATUS_TEMP, STATUS
```

- To return from the low-priority service routine the instruction “**retfie**” needs to be executed, it re-enables the GIEL bit

High-priority Interrupts

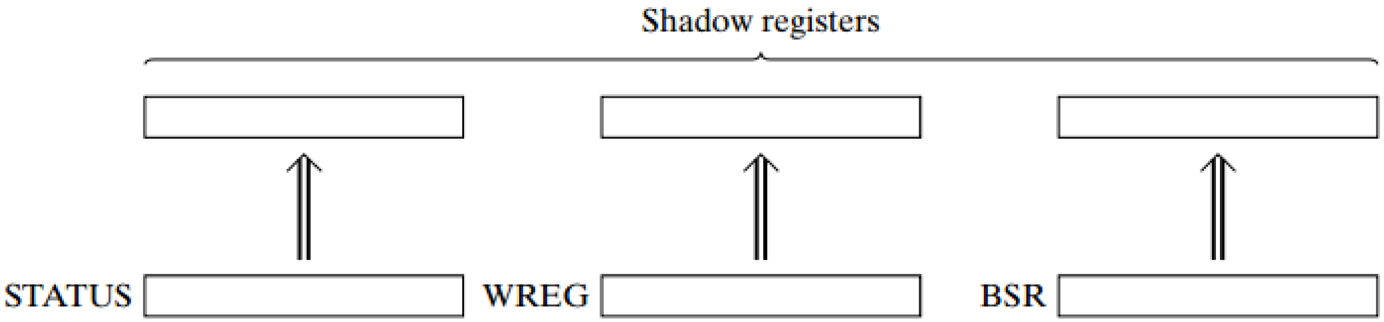
High-priority interrupts are able to suspend the execution of the mainline code and disable all low-priority interrupts and even suspend the execution of a low-priority service interrupt routine.

the following sequence of events takes place automatically:

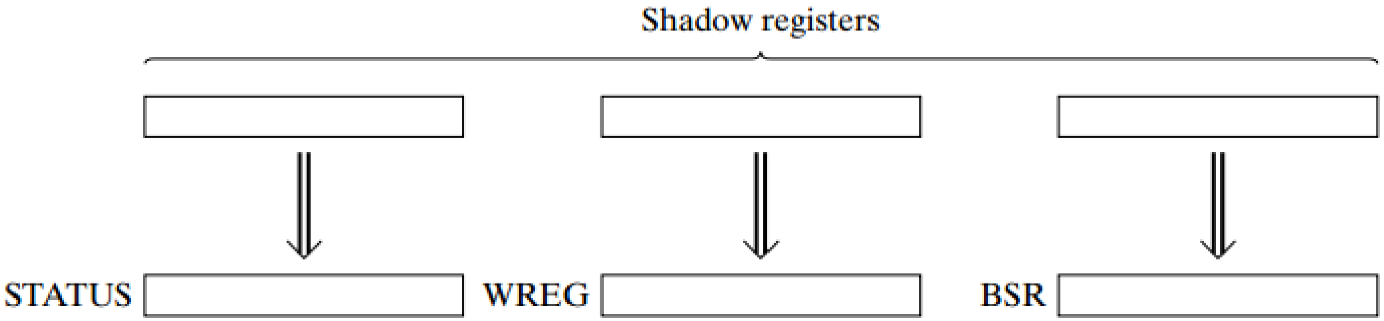
- The **GIEH** bit is cleared, disabling all the interrupts
- The program counter (PC) is stacked
- The program counter is set to 0x0008, the high-priority interrupt vector.
- **STATUS, WREG** and **BSR** are automatically copied into *shadow registers*

High-priority Interrupts

- To return from the low-priority service routine the instruction “**retfie FAST**” needs to be executed, it re-enables the GIEH and restores **STATUS, WREG** and **BSR**



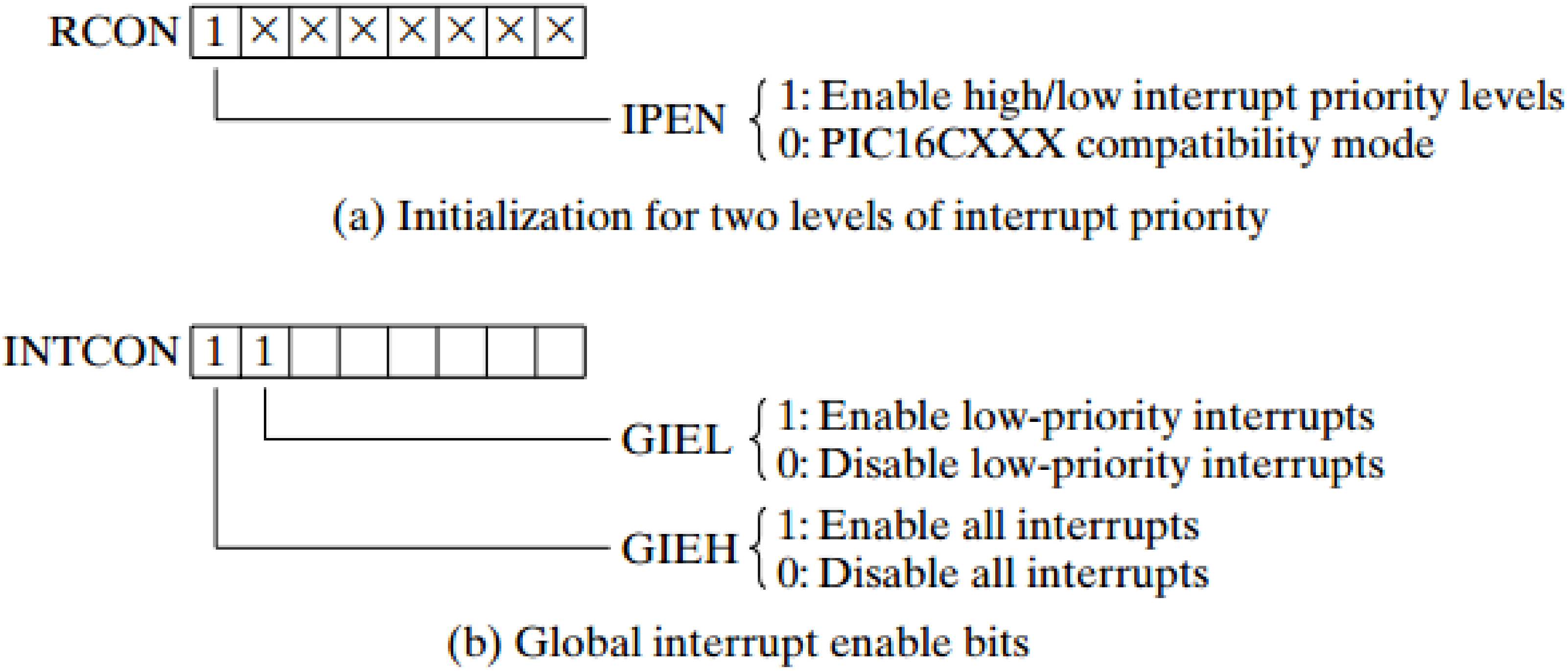
(a) Automatic setting aside of **STATUS**, **WREG**, and **BSR** when a high-priority interrupt occurs



(b) Automatic restoration of **STATUS**, **WREG**, and **BSR** in response to the “**retfie FAST**” instruction.

Interrupt Initialization

High-priority/low-priority interrupts scheme begins with the settings of the **IPEN** bit and the **GIEL/GIEH** bits



RCON 1 × × × × × × ×

IPEN
- 1: Enable high/low interrupt priority levels
- 0: PIC16CXXX compatibility mode

(a) Initialization for two levels of interrupt priority

INTCON 1 1

GIEL
- 1: Enable low-priority interrupts
- 0: Disable low-priority interrupts

GIEH
- 1: Enable all interrupts
- 0: Disable all interrupts

(b) Global interrupt enable bits

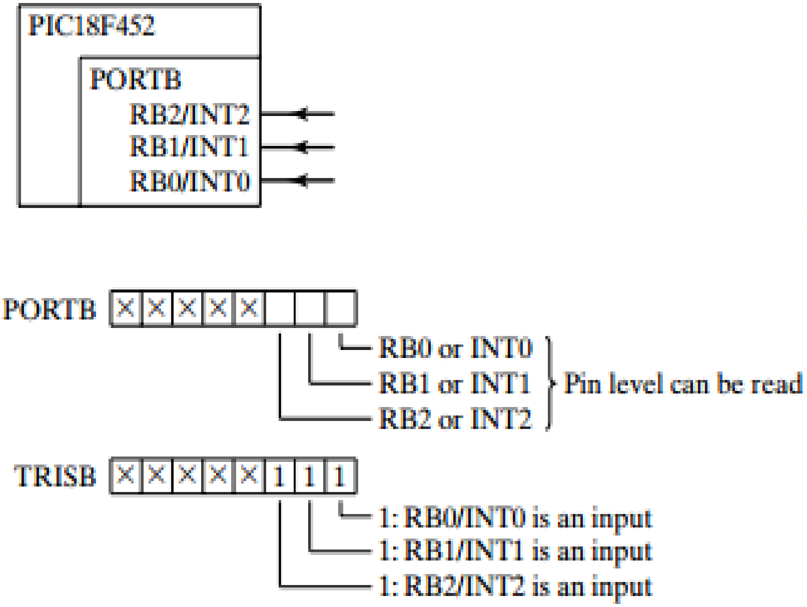
Critical regions

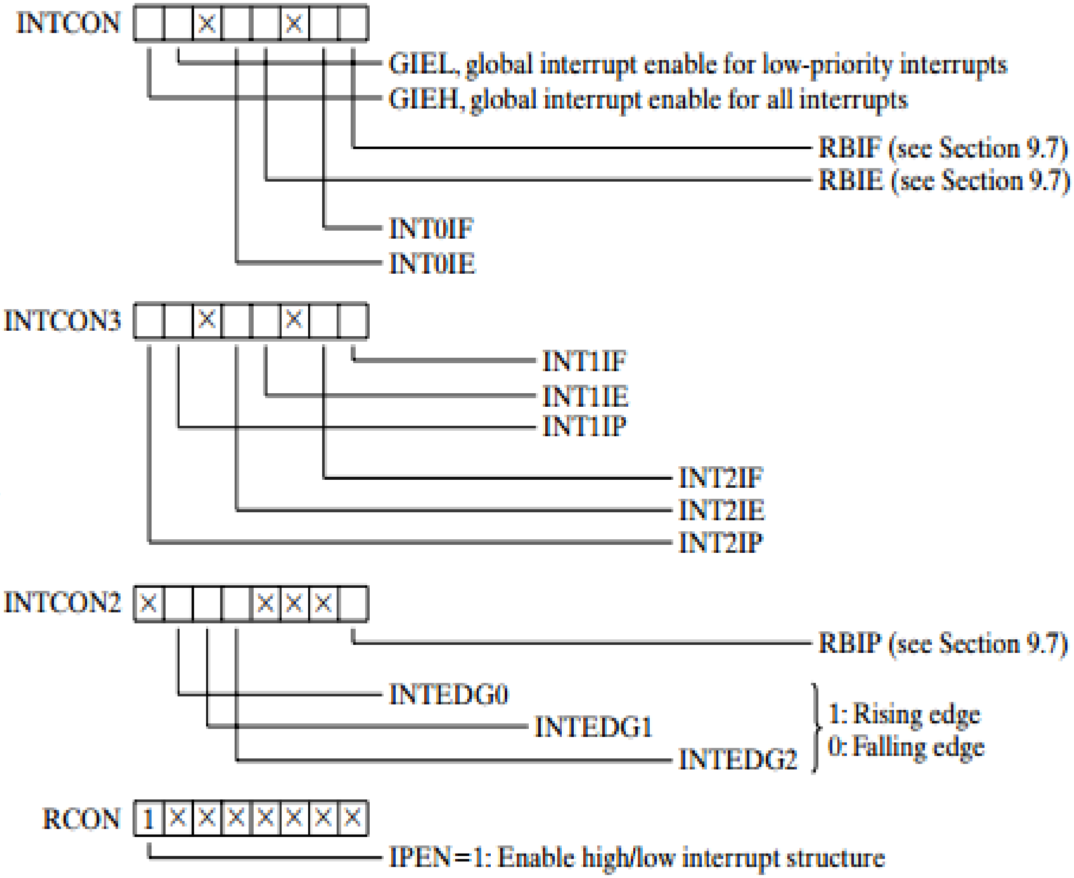
A critical region of code is a sequence of program instruction that must not be interrupted if erroneous operation is to be avoided.

```
LoopTime
    btfss  INTCON,TMR0IF                 ;Wait until ten milliseconds are up
    bra  LoopTime
    movff  INTCON,INTCONCOPY
    bcf  INTCON,GIEH                     ;Disable all interrupts to CPU
    movff  TMR0L, TMR0LCOPY              ;Read 16-bit counter at this moment
    movff  TMR0H, TMR0HCOPY
    movlw  low  Bignum
    addwf  TMR0LCOPY, F
    movlw  high  Bignum
    addwfc  TMR0HCOPY, F
    movff  TMR0HCOPY, TMR0H
    movff  TMR0LCOPY, TMR0L              ;Write 16-bit counter at this moment
    movf  INTCONCOPY, W                  ;Restore GIEH interrupt enable bit
    andlw  B'10000000'
    iorwf  INTCON, F
    bcf  INTCON, TMR0IF                   ;Clear Timer0 flag
    return
```

External Interrupts

The PIC18F452 has three external interrupts inputs **INT0, INT1, INT2** shared with bits 0,1 and 2 of PORTB. To use one of these as interrupt source its control bits must be set up:





See also Ch. 9 in DATASHEET