CCS Pic-C: Interrupts

When interrupts are enabled at global level, this will not enable any of the specific interrupts but will allow any of the specific interrupts previously enabled to become active.
When interrupts are disabled at global level, specific level interrupts are still active but are not taken into consideration by the CPU.
To be able to take care of what needs to be done when interrupts occur, programmer must write interrupt handler functions. These are usually called interrupt service routines (ISRs) and should have been defined for required interrupts.

In CCS PIC-C an #int_xxx directive followed by the xxx_isr() function is used to write ISR for each individual interrupt. Here, xxx can be the constants given previously (i.e. RTCC, RB, EXT, EEPROM, TIMER0, etc.)

The main structure of enabling interrupts and associated ISRs for individual interrupts are:

```
#int_RB
void RB_isr (void)
 {
....
....
}
#int_EXT
void EXT_isr (void)
{
....
....
}
#int_EEPROM
void EEPROM_isr (void)
{
....
....
}
.
#int_TIMER0    // Note: RTCC and Timer() are the same
void TIMER0_isr (void)
{
....
....
}


void main (void) {
enable_interrupts (INT_RB);
enable_interrupts (INT_EXT);
enable_interrupts (INT_EEPROM);
enable_interrupts (INT_TIMER0);
enable_interrupts (GLOBAL);
 ....
  ....
```