INTERRUPCIONS

El PIC18 és molt més ràpid que I/O. No pot perdre temps esperant si ha rebut dada.

- Blind Cycle = Sabem que es repeteix cada X temps. Comprovem quan sigui moment.
- Bussy Waiting = Només mirem l'estat.
- Interrupció = Es fa ús de HW per causar una execució especial. # Temps resposta essencial
- DMA = Tranf. de dades en temps real Dispo. ↔ Mem. sense passar CPU. # Latència importa.

Interrupció

Mentre executa codi principal, si hi ha interrupció que nosaltres haguem configurat, saltem a una rutina (ISR) especial per tractar event. Posteriorment retorna on estavem.

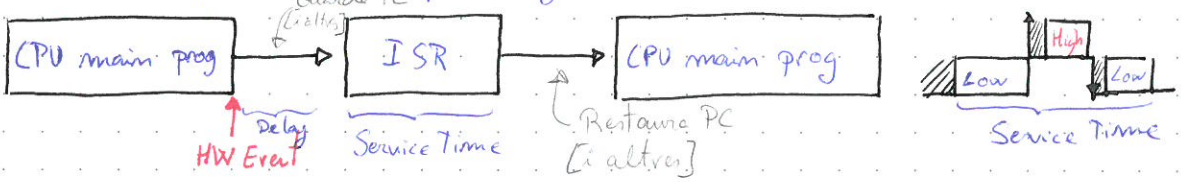
⚠ Són asíncrones respecte les instruccions. Quan arriben, primer acaba l'int. si meitat. Les excepcions / traps sí que síncrones i estan lligades a instruccions.

Reset Service Routine: Situada 0x0000. Reseteja registres, biestables, ...
Reset Registra indica motiu de reset.
No és una ISR.

Per defecte PIC18 guarda PC per retornar post-ISR "retfie" ← El "return" específic d'una ISR i restaura PC

És la nostra obligació guardar registres si es de baixa prioritat. Abans del codi ISR → Guardar. Després del codi ISR → Restaurar

Si es d'alta Prioritat, per defecte es guarden "STATUS, WREG, BSR" i "retfie FAST" restaura.



- GIE = Global Interrupt Enable. Prenem (1) o no (0) atenció a les interrupcions.
  - ↳ GIEH = Només High ; GIEL = Només Low
- xxxIF = Interrupt Flag. Diu si hi ha flag pendent.
- xxxIP = Interrupt Priority ↳ 1 = High, 0 = Low
- xxxIE = Interrupt Enable. (1) Sí (0) No

⚠ Revisar Form per saber INTCONx

- PEIE = Peripheral int. Enable
- IPEN = Interrupt Priority Enable ↳ Low + High (1), High (0)
- INTEDGx = Selecionan flanc ascendent o no ↳ High (1), Low (0)

Resum

Prioritats		Core?	Que Activo?
NO		SI	xxx IE = 1 GIE = 1
		NO	xxx IE = 1 GIE = 1 PEIE = 1
SI	HIGH		xxx IE = 1 IPEN = 1 xxx IP = 1 GIEH = 1
	LOW		xxx IE = 1 IPEN = 1 xxx IP = 0 GEIH = 1 GEIL = 1

High ISR (0x00008)

```
void interrupt FUNCTION(void){
    if(xxxIE && xxxIF){
        ...
    }
}
```

Low ISR (0x00018)

IMPO això

```
void interrupt low_priority FUNCTION(void){
    if(xxxIE && xxxIF){
        ...
    }
}
```

Fixem-se que pot tardar 3-4 cicles d'instrucció de latència.