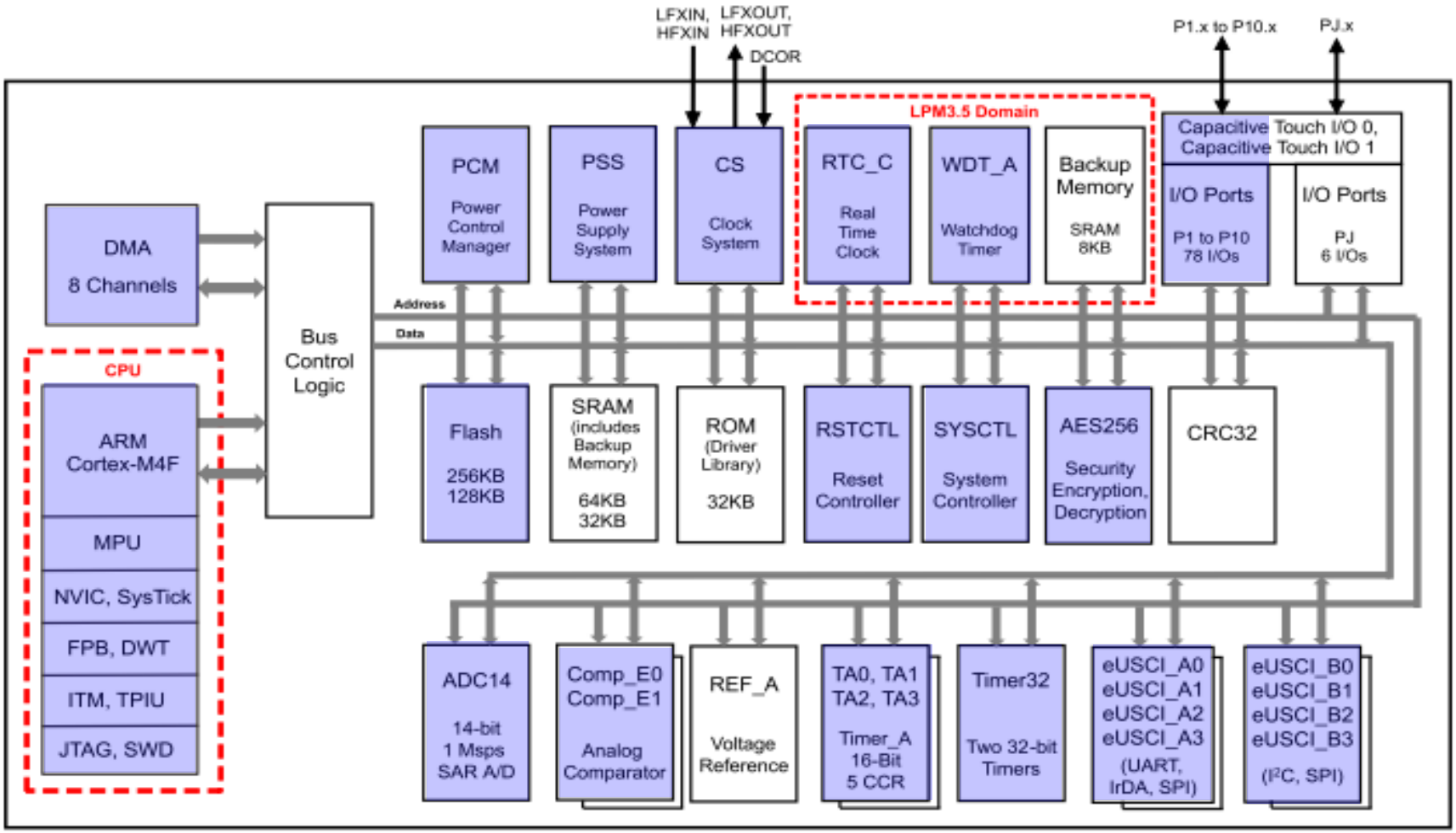
PROGRAMACIÓ D'ARQUITECTURES ENCASTADES

Interrupcions i *Timers*

Classe 3

UNIVERSITAT DE BARCELONA

Dispositius que poden Generar Interrupcions al MSP432P401

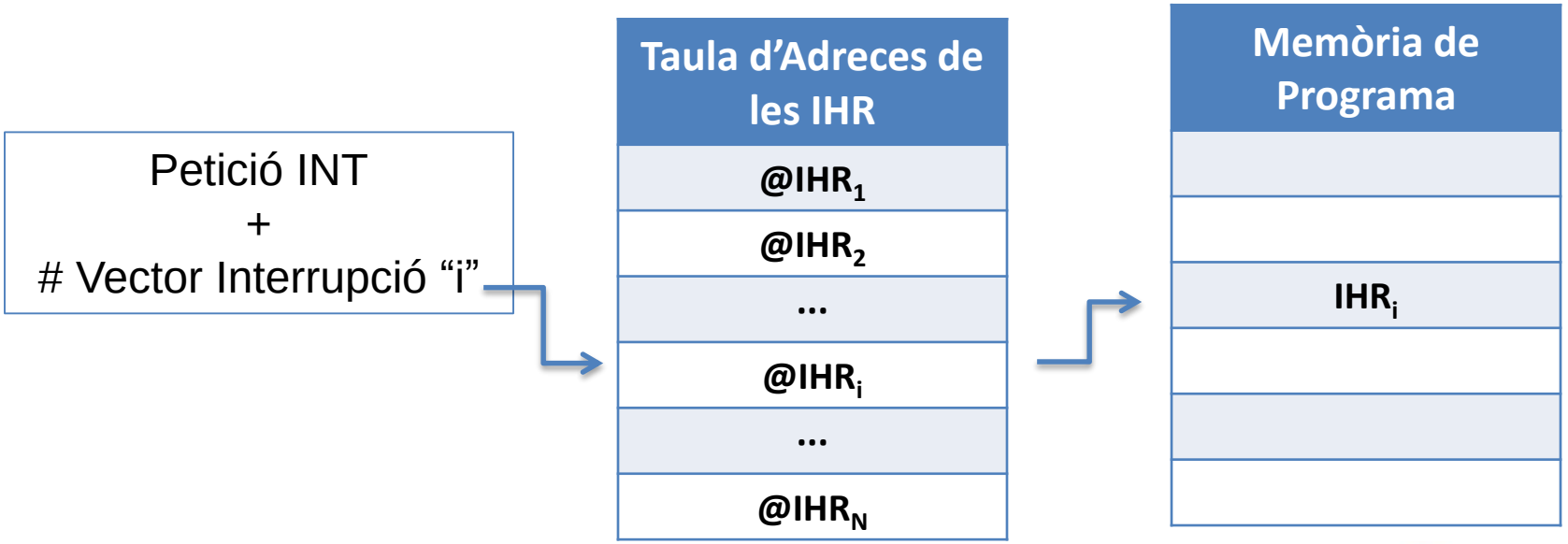


UNIVERSITAT DE BARCELONA

Gestió mitjançant un Controlador d'Interrupcions

Aquests sistemes es basen en que la identificació de la font d'interrupció, les prioritats i l'emmascarament, es fan per hardware.

- Cada dispositiu que pot generar interrupcions té associat un identificador, anomenat **Vector d'Interrupció**, que ha de proporcionar al processador quan demana una interrupció.
- Cada vector d'interrupció té asociada una posició a una taula on es guarden les adreces de les rutines d'atenció a les interrupcions (IHR_i).



UNIVERSITAT DE BARCELONA

Fonts d'Interrupcions del MSP432P401

- ***Reset*:** diverses fonts.
- **CPU *(Cortex-M4):*** varis vectors amb diverses fonts (NMI, *Debug*, Bus...).
- **"*Clock*", "*Power*", "*System*", "FPU"...**: varis vectors amb diverses fonts.
- ***Watchdog Timer_A*** en mode Interval.
- **Comparadors 0 i 1**: un vector per cada font.
- ***4 Timers* A0-A3**: 2 vectors per *timer*, un únic i un altre amb diverses fonts.
- **EUSCIA0 a EUSCIA3:** cada una té un vector amb diverses fonts.
- **EUSCIB0 a EUSCIB3**: cada una té un vector amb diverses fonts.
- **ADC14** (Convertidor Analògic Digital): un vector amb diverses fonts.
- ***2 Timers de 32 bits 1 i 2*** :3 vectors (un combinat), alguns amb diverses fonts.
- **AES256 (Accelerador per encriptació)**.
- ***Real Time Clock, RTC_C***: diverses fonts.
- **DMA** (Accés Directe a Memòria): 4 vectors amb diverses fonts.
- **GPIOs, Port1 a Port6**: cada *port* amb un vector amb diverses fonts.

EUSCI = Enhanced Universal Serial Communication Interface.

UNIVERSITAT DE BARCELONA

Sistema d'Interrupcions del MSP432P401

Per fer servir les interrupcions de qualsevol recurs, hem de seguir una sèrie de passes:

- Habilitar les interrupcions. Al nostre processador es fa a 3 nivells:
  1. **A nivell de dispositiu** (s'ha de consultar el *Technical reference* per a cada dispositiu).
  2. **A nivell del controlador d'interrupcions del processador**, anomenat **NVIC** (*Nested Vectored Interrupt Controller*).
  3. **A nivell global** al registre de "status" del processador, fent servir la instrucció: **__enable_interrupt();**
- De fet, al primer pas que hem indicat, a nivell de dispositiu, podem emmascarar (inhabilitar) algunes interrupcions i mantenir altres actives.
- El controlador d'interrupcions també gestiona un sistema de prioritats, per aquelles situacions en que més d'una interrupció s'activi simultàniament.

UNIVERSITAT DE BARCELONA

NVIC *(Nested Vectored Interrupt Controller)*

Pot gestionar fins a 64 vectors d'interrupció, amb 8 nivells de prioritat.
Per gestionar el **NVIC** a nivell de programació, hi ha 7 tipus de registres, dels quals nosaltres en farem servir només 2:

1. 2 registres de activació d'interrupcions, **ISER0 i ISER1** (*Interrupt Set Enable Register*). Cada un d'ells de 32 bits. Amb aquests 2 registres farem la habilitació a nivell de NVIC de cada font d'interrupció.
2. 2 registres d'esborrat d'interrupcions pendents, **ICPR0 i ICPR1** (*Interrupt Clear Pending Register*) també de 32 bits.

La resta de registres són per modificar prioritats, marcar interrupcions com a pendents, comprovar si una interrupció està activa, o activar-les per *software*. Però, al menys per ara, no els farem servir.
La informació referent a aquests registres i al NVIC està al capítol 2.4.3 del *technical reference*. (pàg. 105-119)

També necessitarem la informació que hi ha a la taula 6-39 del *Datasheet* (pàg.117 i 118) que ens especifica el significat de cada bit per fer servir els registres ISERx i ICPRx.

UNIVERSITAT DE BARCELONA

NVIC *(Nested Vectored Interrupt Controller)*

NVIC INTERRUPT INPUT	SOURCE	FLAGS IN SOURCE
INTISR[3]	WDT_A	
INTISR[4]	FPU_INT (2)	Combined interrupt from flags in the FPSCR (part of Cortex-M4 FPU)
INTISR[5]	FLCTL	Flash Controller interrupt flags
INTISR[6]	COMP_E0	Comparator_E0 interrupt flags
INTISR[7]	COMP_E1	Comparator_E1 interrupt flags
INTISR[8]	Timer_A0	TA0CCTL0.CCIFG
INTISR[9]	Timer_A0	TA0CCTLx.CCIFG (x = 1 through 4), TA0CTL.TAIFG
INTISR[10]	Timer_A1	TA1CCTL0.CCIFG
INTISR[11]	Timer_A1	TA1CCTLx.CCIFG (x = 1 through 4), TA1CTL.TAIFG
INTISR[12]	Timer_A2	TA2CCTL0.CCIFG
INTISR[13]	Timer_A2	TA2CCTLx.CCIFG (x = 1 through 4), TA2CTL.TAIFG
INTISR[14]	Timer_A3	TA3CCTL0.CCIFG
INTISR[15]	Timer_A3	TA3CCTLx.CCIFG (x = 1 through 4), TA3CTL.TAIFG
INTISR[16]	eUSCI_A0	UART or SPI mode TX, RX, and Status Flags
INTISR[17]	eUSCI_A1	UART or SPI mode TX, RX, and Status Flags
INTISR[18]	eUSCI_A2	UART or SPI mode TX, RX, and Status Flags
INTISR[19]	eUSCI_A3	UART or SPI mode TX, RX, and Status Flags
INTISR[20]	eUSCI_B0	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[21]	eUSCI_B1	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[22]	eUSCI_B2	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[23]	eUSCI_B3	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[24]	ADC14	IFG[0-31], LO/IN/HI-IFG, RDYIFG, OVIFG, TOVIFG
INTISR[25]	Timer32_INT1	Timer32 Interrupt for Timer1
INTISR[26]	Timer32_INT2	Timer32 Interrupt for Timer2
INTISR[27]	Timer32_INTC	Timer32 Combined Interrupt
INTISR[28]	AES256	AESRDYIFG
INTISR[29]	RTC_C	OFIFG, RDYIFG, TEVIFG, AIFG, RT0PSIFG, RT1PSIFG
INTISR[30]	DMA_ERR	DMA error interrupt
INTISR[31]	DMA_INT3	DMA completion interrupt3
INTISR[32]	DMA_INT2	DMA completion interrupt2
INTISR[33]	DMA_INT1	DMA completion interrupt1
INTISR[34]	DMA_INT0 (3)	DMA completion interrupt0
INTISR[35]	I/O Port P1	P1IFG.x (x = 0 through 7)
INTISR[36]	I/O Port P2	P2IFG.x (x = 0 through 7)
INTISR[37]	I/O Port P3	P3IFG.x (x = 0 through 7)
INTISR[38]	I/O Port P4	P4IFG.x (x = 0 through 7)
INTISR[39]	I/O Port P5	P5IFG.x (x = 0 through 7)
INTISR[40]	I/O Port P6	P6IFG.x (x = 0 through 7)
INTISR[41]	Reserved	
INTISR[42]	Reserved	

Aquesta és una part de la taula 6-39 esmentada. A la primera columna tenim un "registre" de 64 bits (no surt complert) que correspon als nostres ISER[0], ICPR[0] els primers 32 bits i ISER[1], ICPR[1] els últims 32 bits.

El que és important és veure que cada bit d'aquest registre està associat amb un vector d'interrupció. Si el posem a "1" al ISER[x] corresponent, l'habilitem al NVIC. Per inhabilitar-la hem de posar a 1 el corresponent bit als registres ICER[x]

UNIVERSITAT DE BARCELONA

Exemple d'Interrupcions a un *port* GPIO

Suposem que volem fer servir tot el *port* 2 com a GPIOs d'entrada i habilitar tots els seus pins per generar interrupcions:

A nivell de dispositiu (*Technical reference*, cap. 10.4, pàg. 520-521):

```
P2IE |=0xFF; //Habilitem a nivell de dispositiu les interrupcions als 8 pins (P2.0 a P2.7)
P2IES &=0x00; //Volem que les interrupcions saltin al flanc de pujada L->H
```

A nivell de NVIC:

```
NVIC->ICPR[1] |= BIT4; //Ens assegurem que no quedin interrupcions pendents al port 2
NVIC->ISER[1] |= BIT4; //Habilitem les interrupcions al port 2
```

A més haurem de posar en algun moment: **__enable_interrupt();**

Una bona pràctica és agrupar totes les habilitacions d'interrupcions a nivell de NVIC a una funció, per exemple "void init_interrupcions(void)....".
Les habilitacions a nivell de dispositiu és preferible fer-les a la inicialització de cada dispositiu.
I, sobretot, mai executar el "__enable_interrupt();" abans d'haver fet les dues coses anteriors.

UNIVERSITAT DE BARCELONA

Exemple d'Interrupcions a un *port* GPIO

La rutina d'atenció a la interrupció (IHR o ISR)* s'ha d'anomenar obligatòriament:

```
void PORT2_IRQHandler (void)        //Veure nota abaix.
```

Dintre d'aquesta funció, com els 8 *pins* del *port* 2 comparteixen el mateix vector d'interrupció hem d'esbrinar quin dels *pins* ha generat la interrupció. Per exemple, amb un "**switch-case**" (sondeig per programa o *software polling*) dos opcions:

- El Vector d'Interrupcions del Port "x", que es diu **PxIV. (Opció Recomenada)**
- El registre de *Flags* d'Interrupcions del *Port* "x", que es diu **PxIFG**.

NOTA: Els noms de les IHR estan a una "taula" que es troba al fitxer "startup_msp432p401r_ccs.c" que es crea sempre que fem un projecte nou. Haurem de consultar aquesta taula per saber el nom de la IHR associada a cada font (vector) d'interrupció.

***IHR** (*interrupt Handle Register*) és el mateix que **ISR** (*Interrupt Service Register*)

UNIVERSITAT DE BARCELONA

Exemple d'Interrupció: vector amb diverses fonts

Al *Technical reference* (pag. 516 i 520) podem veure l'estructura dels dos registres:

Registre de *Flags* d'Interrupcions del *Port* 1 al *Port* 6, **PxIFG**.

Figure 10-12. PxIFG Register

7	6	5	4	3	2	1	0
PxIFG							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 10-15. PxIFG Register Description

Bit	Field	Type	Reset	Description
7-0	PxIFG	RW	0h	Port X interrupt flag 0b = No interrupt is pending. 1b = Interrupt is pending.

Vector d'Interrupcions del *Port* 1 al *Port* 6, **PxIV.**

Figure 10-1. PxIV Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved			PxIV				
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 10-4. PxIV Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Reads return 0h
4-0	PxIV	R	0h	Port x interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Port x.0 interrupt; Interrupt Flag: PxIFG.0; Interrupt Priority: Highest 04h = Interrupt Source: Port x.1 interrupt; Interrupt Flag: PxIFG.1 06h = Interrupt Source: Port x.2 interrupt; Interrupt Flag: PxIFG.2 08h = Interrupt Source: Port x.3 interrupt; Interrupt Flag: PxIFG.3 0Ah = Interrupt Source: Port x.4 interrupt; Interrupt Flag: PxIFG.4 0Ch = Interrupt Source: Port x.5 interrupt; Interrupt Flag: PxIFG.5 0Eh = Interrupt Source: Port x.6 interrupt; Interrupt Flag: PxIFG.6 10b = Interrupt Source: Port x.7 interrupt; Interrupt Flag: PxIFG.7; Interrupt Priority: Lowest

UNIVERSITAT DE BARCELONA

Exemple d'Interrupció: vector amb diverses fonts

Si el port 2 el tinguéssim configurat com GPIOs d'entrada que poden generar interrupcions:

```
void PORT2_IRQHandler (void)  //interrupcions GPIO del port 2.
{
    uint8_t flag = P2IV
    P2IE &= 0x00;    //Inhabilitem temporalment les interrupcions de tot el port 2

    switch (flag) {
        case 0x02:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.0
                    break;
        case 0x04:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.1
                    break;
        case 0x06:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.2
                    break;
        case 0x08:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.3
                    break;
        case 0x0A:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.4
                    break;
        case 0x0C:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.5
                    break;
        case 0x0E:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.6
                    break;
        case 0x10:
    // Aquí posem el que volem fer si s'ha generat una interrupció al P2.7
                    break;
        default:    break;
                    }
    P2IE |= 0xFF;    //Tornem a habilitar les interrupcions del port 2
}
```

Figure 10-1. PxIV Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved			PxIV				
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Table 10-4. PxIV Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Reads return 0h
4-0	PxIV	R	0h	Port x interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Port x.0 interrupt; Interrupt Flag: PxIFG.0; Interrupt Priority: Highest 04h = Interrupt Source: Port x.1 interrupt; Interrupt Flag: PxIFG.1 06h = Interrupt Source: Port x.2 interrupt; Interrupt Flag: PxIFG.2 08h = Interrupt Source: Port x.3 interrupt; Interrupt Flag: PxIFG.3 0Ah = Interrupt Source: Port x.4 interrupt; Interrupt Flag: PxIFG.4 0Ch = Interrupt Source: Port x.5 interrupt; Interrupt Flag: PxIFG.5 0Eh = Interrupt Source: Port x.6 interrupt; Interrupt Flag: PxIFG.6 10h = Interrupt Source: Port x.7 interrupt; Interrupt Flag: PxIFG.7; Interrupt Priority: Lowest

UNIVERSITAT DE BARCELONA

Gestió de les Interrupcions

A més, per fer servir les interrupcions les hem d'activar prèviament a la seva utilització.

Per fer això, cada font d'interrupcions té un bit d'habilitació (Enable). Per exemple, pel Port 2 tenim el registre P2IE (*Technical reference* pag.520) que permet habilitar/inhabilitar cada una de les fonts d'interrupció del port:

Port 2 Interrupt Enable Register (P2IE)

7	6	5	4	3	2	1	0
P2IE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

P2IE	Bits 7-0	Port 2 interrupt enable
	0	Corresponding port interrupt disabled
	1	Corresponding port interrupt enabled

En qualsevol moment podem habilitar/inhabilitar "globalment*" amb:

```
_enable_interrupt();
```

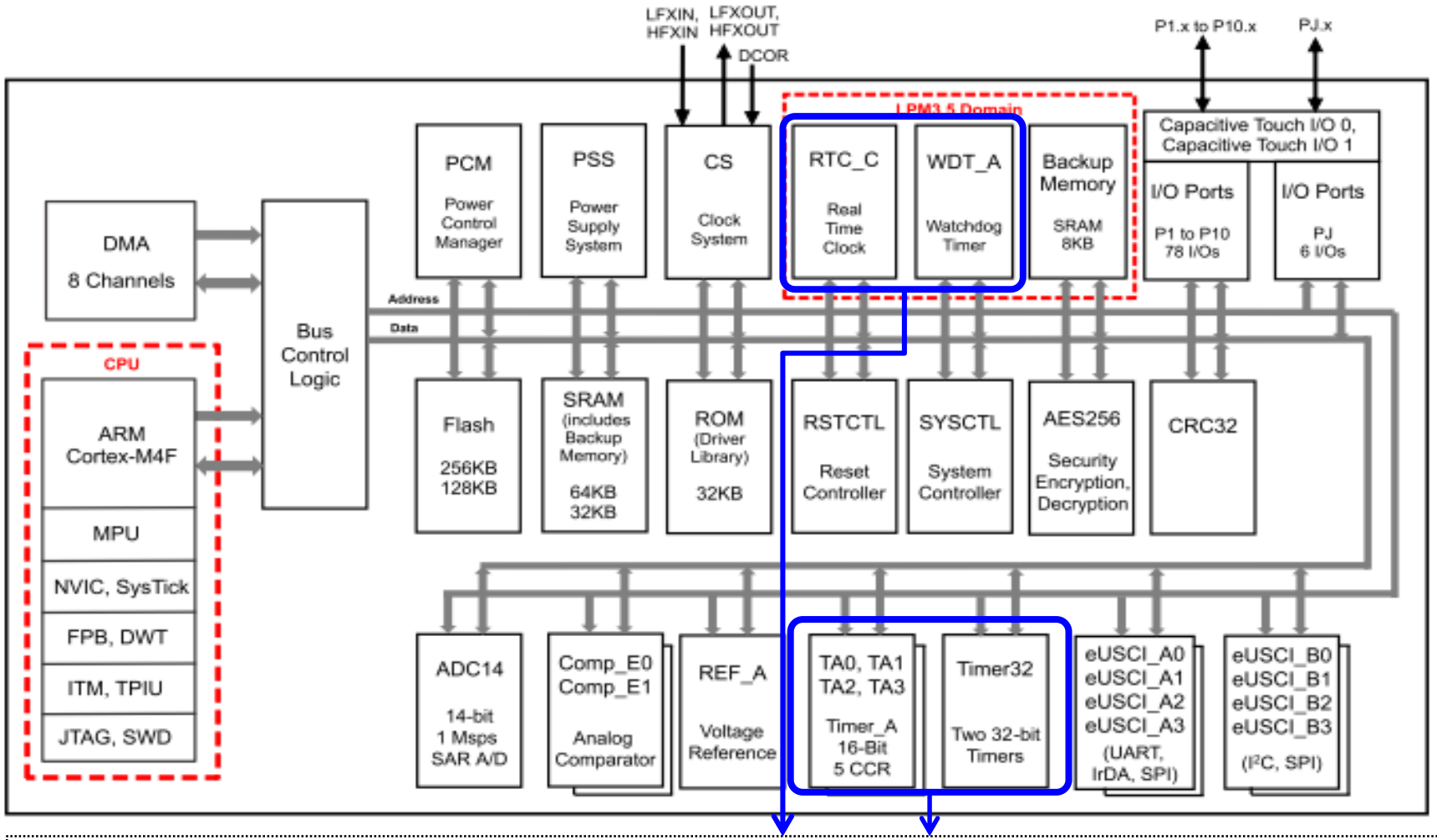
i

```
_disable_interrupt();
```

*** Només s'habilitaran les interrupcions que prèviament s'han activat individualment al seu registre.**

UNIVERSITAT DE BARCELONA

Diagrama de blocs Funcional del MSP432P401



Sistema de *Timers/Counters* (Temporitzadors/Comptadors) del nostre microcontrolador.

UNIVERSITAT DE BARCELONA

Sistema de *Timers* del MSP432P401

Disposem dels següents recursos:

- *Timer* TA0
- *Timer* TA1
- *Timer* TA2
- *Timer* TA3

Fins 16 bits, poden treballar com *PWM*, Repetitiu, Sortir a pins, *Capture/Compare*, Generar Interrupcions…

- 2 Timers de 32 bits. (poden generar interrupcions).
- RTC_C (*Real Time Clock*) que pot treballar com:
  - Rellotge en temps real amb funcions de calendari (compensació de mesos de diferents número de dies) i alarmes.
  - Pot generar interrupcions.

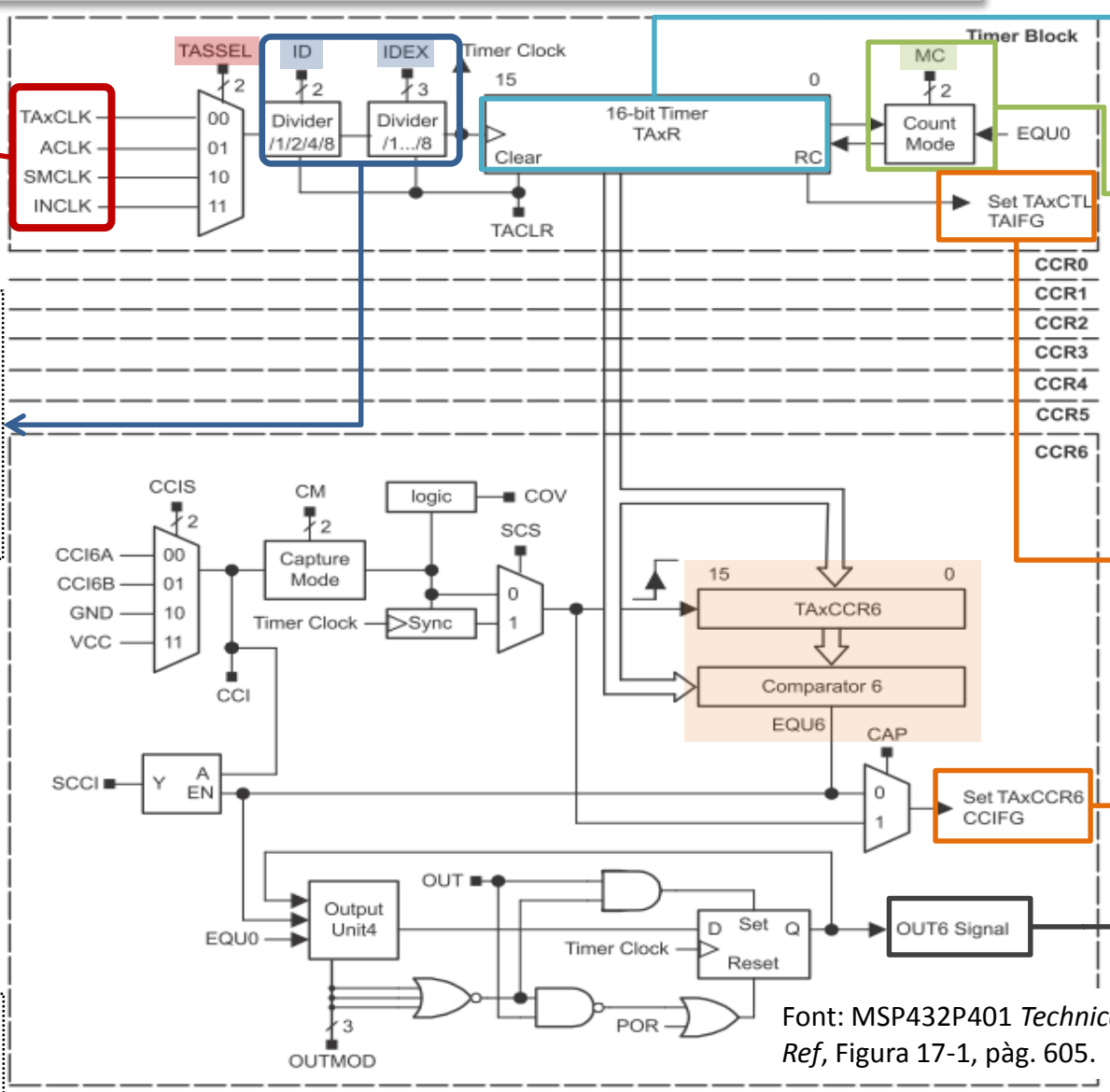
Font: MSP432P401 *Datasheet*, capítol 6.9 *Peripherals*

UNIVERSITAT DE BARCELONA

Diagrama de Blocs del *Timers* de 16 bits

Fonts de rellotge, es seleccionen mitjançant **TAxCTL**

Abans d'entrar al *Timer* es pot dividir la font:
ID a **TAxCTL**
IDEX a **TAxEX0**



Font: MSP432P401 *Technical Ref*, Figura 17-1, pàg. 605.

Copyright © 2016, Texas Instruments Incorporated

Valors per omissió:
ACLK : 32.768Hz
SMCLK : 2MHz

Aquest és el comptador que es va incrementant amb la font seleccionada **TAxR**

Selecció del Mode de funcionament **TAxCTL**

Podem fer que es generi una interrupció quan el comptador torna a 0 després d'arribar al màxim comptatge **TAxR** (*flag* **TAIFG** de **TAxCTL**)
O carregar un valor diferent al registre de captura comparació que volem fer servir **TAxCCRx** i generar interrupcions al arribar a aquest valor (*flag* **CCIFG** de **TAxCCTLx**)

També podem triar que surti un senyal per un pin del microcontrolador.

UNIVERSITAT DE BARCELONA

Modes de Funcionament dels *Timers* de 16 bits

Els ***Timers* Ax** tenem 4 modes de funcionament que es controlen mitjançant els bits MC del registre TA0CTL, TA1CTL, TA2CTL, TA3CTL, segons el *timer*:

- ***Stop:*** MC=00 (**MC_0**) El *timer* està parat.
- ***Up:*** MC=01 (**MC_1**) El *timer* compta de forma cíclica des de 0 a **TAxCCR0***.
- ***Continous:*** MC=10 (**MC_2**) El *timer* compta de forma cíclica des de 0 al seu valor màxim.
- ***Up/Down:*** MC=11 (**MC_3**) El *timer* compta de forma cíclica des de 0 a **TAxCCR0*** i torna a 0.

A qualsevol dels modes podem fer que generi una interrupció cada cicle.

*** TA0CCR0, TA1CCR0, TA2CCR0, TA3CCR0 segons treballem amb TA0 , TA1, TA2 o TA3 respectivament.**

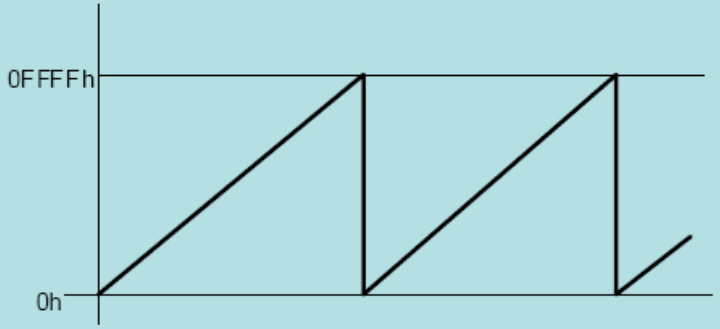
Font: MSP432P4xx *Technical reference*, 17.2.3 pàg. 607-610.

UNIVERSITAT DE BARCELONA

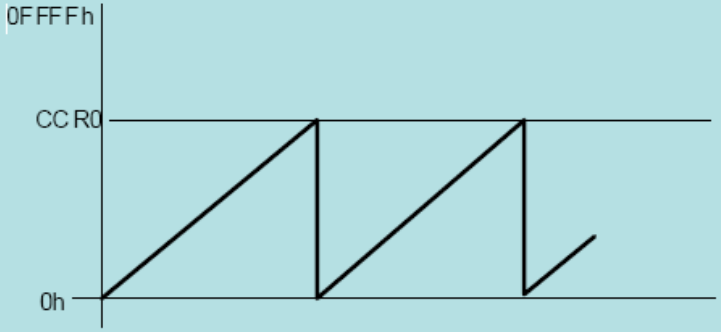
Modes de Funcionament dels *Timers* de 16 bits

Stop/Halt
Timer is halted

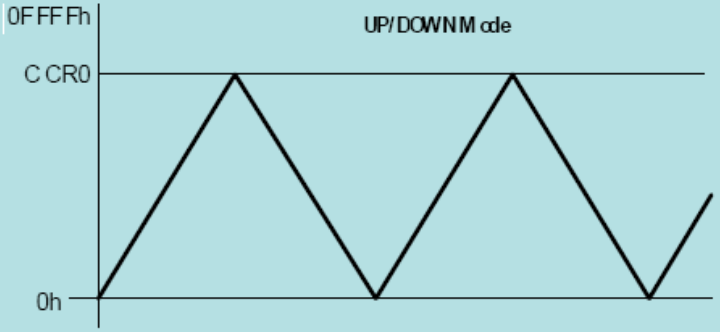
Continuous
Timer continuously counts up



Up
Timer counts between 0 and CCR0

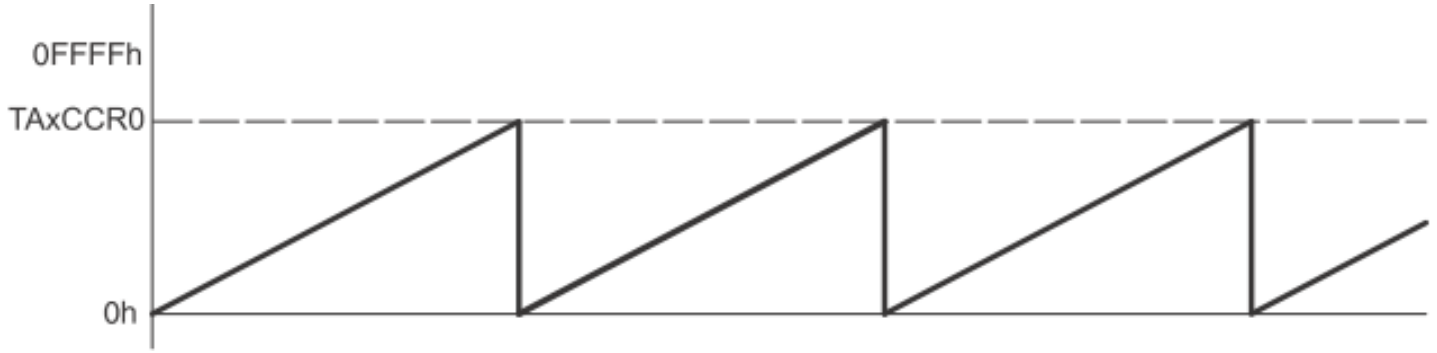


Up/Down
Timer counts between 0 and CCR0 and 0

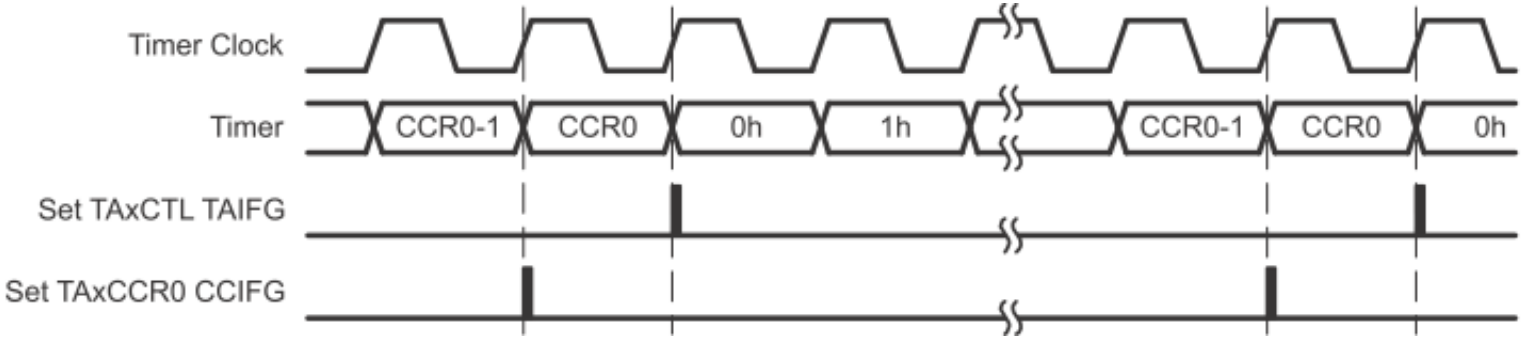


UNIVERSITAT DE BARCELONA

Exemple Mode *Up* al *timers* TA



Evolució d'un *Timer* A en mode Up



Si hem habilitat les interrupcions es poden generar interrupcions amb 2 *flags* diferents el del tornada a 0 del *timer* (TAxIFG) i/o el de arribar a TAxCCR0. De fet, es poden generar més si fem servir els altres registres *capture/compare* del *timer* (veure *Technical reference*).

Font: MSP432P4xx *Technical reference*, 17.2.3 pàg. 607-610.

UNIVERSITAT DE BARCELONA

Més característiques dels *timers* de 16 bits

- Els *timers,* tal com els hem descrit fins ara, treballen en el mode denominat "compare". També es podem programar per treballar en mode "*capture*" (**CAP=1** als registres **TAxCCTLn**). En aquest mode es fan servir junt amb pins d'entrada per capturar esdeveniments externs i poder mesurar quan succeeixen, permetent calcular intervals de temps o velocitats.

- Per altre banda, si fem servir els pins de sortida dels *timers*, podem aconseguir una gran varietat de senyals. Sent de gran interès els **PWM** (*Pulse With Modulation*) ja que permeten controlar molts tipus de dispositius (motors, calefactors, il·luminació, so...).

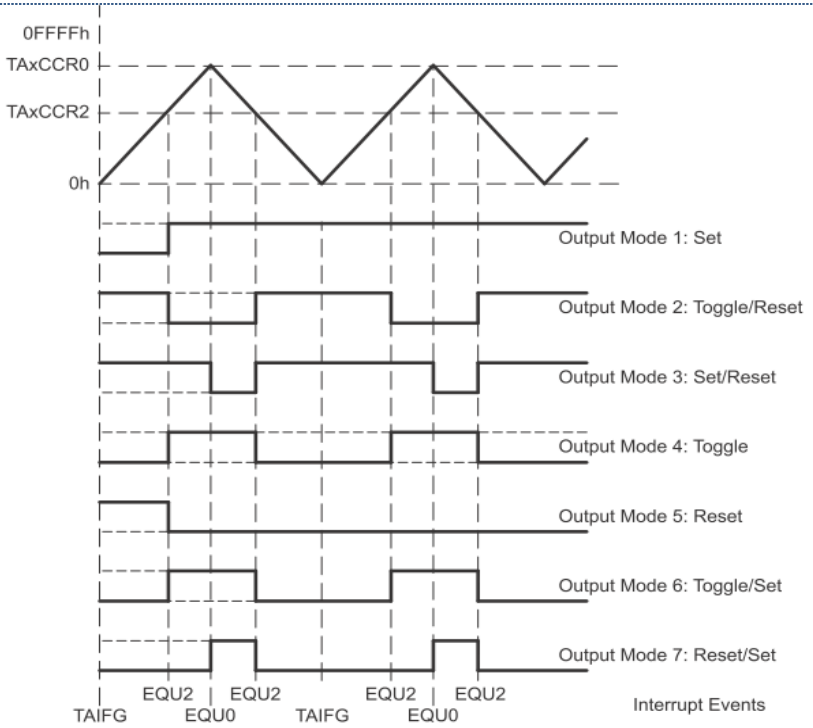


Figure 12-14. Output Example – Timer in Up/Down Mode

Font: MSP432P4xx *Technical reference*, 17.2.3 pàg. 607-610.

UNIVERSITAT DE BARCELONA

Interrupcions als *Timers* de 16 bits

Cada *timer* (**TA0, TA1, TA2 i TA3**) té associat 2 vectors d'interrupció, per tant al nostre microcontrolador tenim 8 vectors d'interrupció:

- **Vector d'Interrupció associat a *TAxCCR0***, *flag* **CCIFG** de ***TAxCCTL0***. Per fer servir aquesta font d'interrupció s'ha d'activar al registre ***TAxCCTL0*** (bit **CCIE**), i al NVIC.
- **Vector d'Interrupció *TAxIV*** per la resta de *flags* **CCIFGn** i el **TAxIFG** (*overflow*). Aquestes les activem al registre ***TAxCTL*** (bit **TAIE**), i al NIVC.

Per tant, quan es genera una interrupció d'aquestes, les rutines d'atenció a les interrupcions presenten dues situacions:

- **TA0_0_IRQHandler, TA1_0_IRQHandler, TA2_0_IRQHandler, TA3_0_IRQHandler:** Directes.
- **TA0_N_IRQHandler, TA1_N_IRQHandler, TA2_N_IRQHandler, TA3_N_IRQHandler:** S'han de comprovar els diferents *flags* per saber qui ha generat la interrupció.

UNIVERSITAT DE BARCELONA

Interrupcions als *Timers* de 16 bits

*** Cas del *timer* A0, si és el A1, A2 o A3 això canvia pel nom corresponent**

Com les fem servir:

- Cas *Timer A0, Timer A1, Timer A2 i Timer A3* (associats a CCR0): **Directes**.

```
void TA0_0_IRQHandler (void)*        /Cas del TA0. Aquest és el nom important
{
   TA0CCTL0 &= ~CCIE; //Convé inhabilitar la interrupció al començament
      /* El que volem fer a la rutina d'atenció d'Interrupció */
      /* Aquí no hem de fer cap comprovació addicional ja que */
      /* només pot haver una causa per generar la interrupció */
      /* que el timer corresponent ha arribat al valor de CCR0 programat */
   TA0CCTL0 &= ~CCIFG; //Hem de netejar el flag de la interrupció
   TA0CCTL0 |= CCIE; //S'ha d'habilitar la interrupció abans de sortir
}
```

- Cas *Timer A0, Timer A1, Timer A2 i Timer A3 (CCR1-CCR6 i TAxIFG):* **Indirectes.**

```
void TA0_N_IRQHandler (void)*        /Cas del TA0. Aquest és el nom important
{
      /* Analitzar els flags CCIFGx i TxIFG per saber qui ha demanat la interrupció,
         llavors fer que el corresponent a cadascuna.
         Com al cas dels Ports, si fem servir TAxIV, l'hem de guardar a una variable
         ja que quan accedim al registre es neteja automàticament */
}
```


UNIVERSITAT DE BARCELONA

Cas Timer A0 Vector Múltiple

```
void TA0_N_IRQHandler(void);
{
  uint16_t flag = TA0IV;
  TA0CTL &= ~(TAIE);
  switch (flag)
  {
    case  0:          // No interrupt
                break;
    case  2:          // TA0CCR1
                break;
    case  4:          // TA0CCR2
                break;
    case  6:          // TA0CCR3
                break;
    case  8:          // TA0CCR4
                break;
    case 10:          // TA0CCR5
                break;
    case 12:          // TA0CCR6
                break;
    case 14:          // TA0IFG overflow handler
                 break;
  }
  TA0CTL |= TAIE;
}
```

VECTOR D'INTERRUPCIÓ: TAxIV

Figure 17-19. TAxIV Register

15	14	13	12	11	10	9	8
TAIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
TAIV							
r0	r0	r0	r0	r-0	r-0	r-0	r0

Table 17-8. TAxIV Register Description

Bit	Field	Type	Reset	Description
15-0	TAIV	R	0h	Timer_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Capture/compare 1; Interrupt Flag: TAxCCR1 CCIFG; Interrupt Priority: Highest 04h = Interrupt Source: Capture/compare 2; Interrupt Flag: TAxCCR2 CCIFG 06h = Interrupt Source: Capture/compare 3; Interrupt Flag: TAxCCR3 CCIFG 08h = Interrupt Source: Capture/compare 4; Interrupt Flag: TAxCCR4 CCIFG 0Ah = Interrupt Source: Capture/compare 5; Interrupt Flag: TAxCCR5 CCIFG 0Ch = Interrupt Source: Capture/compare 6; Interrupt Flag: TAxCCR6 CCIFG 0Eh = Interrupt Source: Timer overflow; Interrupt Flag: TAxCTL TAIFG; Interrupt Priority: Lowest

MSP432P401 *Technical reference*, 17.3 pàg. 622

UNIVERSITAT DE BARCELONA

Bibliografia i Documentació

- MSP432P4xx *Technical Reference Manual*.
- MSP432P401 *Datasheet*.
- www.ti.com/msp432
- MSP-EXP432P401R *LaunchPad User's Guide*.
- *Educational BoosterPack* EDUMKII *User's Guide*.
- http://www.bioloid.info/tiki/tiki-index.php

UNIVERSITAT DE BARCELONA