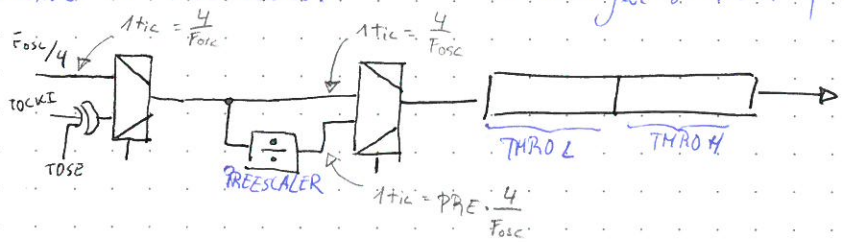
TIMERS

Timer 0

8 bits o 16 bits. Pot ser → Timer → Fa servir Fosc intern (Fosc/4)
→ Counter → Compta periodes externs. (T0CKI)

Hi ha interrupció quan passa de 0xFF..F → 0x00..0

Només té PRE escalers. Pot carregar-se valor previ ament.



Exemple

Càlcul de Valors

1. Primer hem de saber cada quant volem que hi hagi un tick. # Normalment Fosc/4
2. Ara hem de saber si necessitem PRE o no. i com calcular-ho.
   2.1. Volem que hi hagi OVF cada 100 ms. ⇒ Timer 0 compta desde x fins 0xFF..F ⇒ OVF.
   Això sig. que volem saber quants ticks són necessaris per fer 100 ms.
   $100 \times 10^{-3} \div \frac{PRE \cdot 4}{Fosc}$ = ticks necessaris.
   2.2. Donat que estarem fent servir 16b, La quantitat de ticks ha de ser $< 2^{16} - 1$.
   2.3. Finalment queda anar provant possibles valors de PRE # Està formulari.
   ▭ PRE = 2 ⇒ $100 \times 10^{-3} \cdot \frac{1}{\frac{2 \cdot 4}{8 \times 10^6}} = 100.000 > 2^{16} - 1$ ✗
   ▭ PRE = 4 ⇒ $100 \times 10^{-3} \cdot \frac{1}{\frac{4 \cdot 4}{8 \times 10^6}} = 50.000 < 2^{16} - 1$ ✓
3. Això sig. que si PRE = 4, necessitem 50.000 Ticks. ⇒ Com que TMR0 son 16 que és més gran que 50.000 haurem de carregar TMR0 = $(2^{16} - 1) - 50000$ = 15535

Configuran Timer 0

1. Configurar TMR0 # No impo. ordre
2. Afegir el valor del TMR0 ▽ IMPO ORDRE
3. Habilitar interrupcions # No impo. ordre

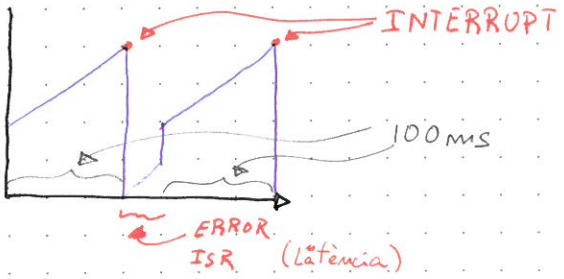
ISR del TMR0

```
void interrupt [low/high] ISR(void){
    if(TMR0IE && TMR0IF){
        TMR0IF=0;
        TMR0 = (2^16-1) - 50000;
    }
}
```

OBS = Si fem servir 16b. necessitem 2 cicles per fer lectura del valor.

Consideracions

- Donat que lectura 16 bits = 2 cicles pot ser que canvi el High i llegim malament. PIC18 fa "foto" del High quan llegim Low.
- A l'hora d'escriure primer s'escriu part alta (En el REG "representant") i en el segón cicle en Low.



H L
0x05 0xFF
0x06 0x00
⇒ Podria passar: L = 0xFF, H = 0x06. Cosa falsa.

Timer 1/3/5

Sempre 16 bits. Pot ser: Timer / Counter } Igual TMR0

Interrupció quan 0xFFFF ⇒ 0x0000 (2B + 2B)

Permeten Complex Gate (Controlen TMR inputs externs) però en CI DESHABILITAT.

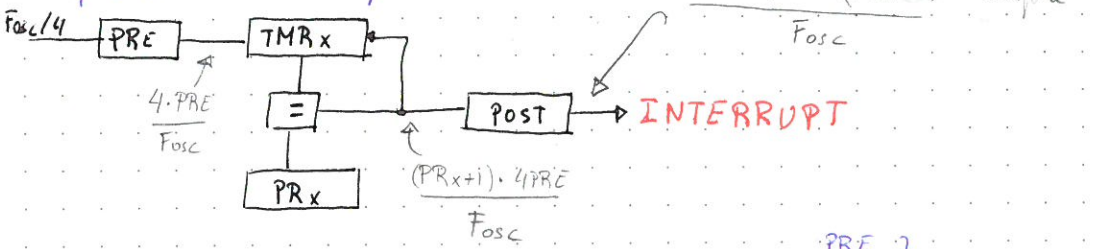
Pot ser usat per Capture / Compare. # Procediment config. igual.

NO OBLIDAR TMRx GE = 0;

Timer 2/4/6

Sempre 8 bits. Sempre Fosc/4. ∇c No té problemes de Latència.

Quan hi ha OVF no hi ha interrupt ⇒ Es compara amb el valor de PRx i quan tenen mateix valor s'envia al POST. Quan hi ha hagut POST vegades si que salta interrupció.



Per calcular-ho hem d'anar provant valors de PRE, POST, PRx } fins interrupt cada 1 ms.

Ara en la ISR no fa falta tocar TMRx pq es RESET automàtic.

Es fa servir per PWM.

Capture, Compare i PWM (CCP)

Capture TMR1/3/5

Registra valor del TMRx en el moment exacte que succeeix un event extern.
Amb el valor del TMRx_PREV fa una resta i pot saber quant temps hi ha hagut entre events. ⇒ Freq senyal entrada.
"TMRx INTERRUPT permeta saber el temps entre CCPx INTERRUPTS":

Exemple

```
void main(void){

  ANSELRC2 = 0; Digital
  TRISRC2 = 1; Input

  CCP1M = 0b0101; Captura ↑↑
  C1TSEL = 0; TMR1
  CCP1IF = 0; Seg
  CCP1IE = 1;

  TMR1GE = 0; ▽ IMPO
  TMR1CS = 0; Fosc/4
  T1CKPS = 0b00; PRE=1
  TMR1IF = 0; Seg
  TMR1IE = 1;

  PEIE = 1; Perifèric
  GIE = 1; Interrupt

{
```

▽ RECORDA

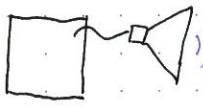
volatile uint16_t valor_captura, valor_captura_prev, num_ticks;
↑ Pq ho tractem en ISR

```
void interrupt high biciISR(void){
  if(TMR1IE && TMR1IF){

    TMR1IF = 0;
    num_ovf++; ← Tractar les vegades que no estem tenint en compte

  {

  if(CCP1IE && CCP1IF){

    valor_captura = CCPR1H<<8 | CCPR1L;

    if(valor_captura_prev >= valor_captura){

      num_ticks = (65535-valor_captura_prev) +
                  valor_captura;

    } else {

      num_ticks = valor_captura - valor_captura_prev;

    {
    num_ticks += (65535)* num_ovf; ← Sumar el que no hem tingut en compte.

    num_ovf = CCP1IF = 0;

  {
}
```

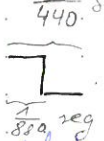
Compare TMR 1/3/5

Compara el valor del TMRx amb CCPx i quan coincideixen CCPx genera INTERRUPT.
Pot fer-se servir amb una lògica post que canvi l'estat d'un PIN.
Quan hi ha una INTERRUPT (ISR) hem d'augmentar el valor del CCPx pq següent sigui al mateix temps. ▽ No hi ha problema ovf pq TMRx tmb. ← NO el capturem

Exemple



Això sig. que volem un període de $\frac{1}{440}$ seg



Però jo només vull sortida a 1 la meitat del temps $\frac{1}{2} \cdot \frac{1}{440}$

Sabem $F_{osc} = 8 \times 10^6$ Hz i $1\ tic = \frac{4 \cdot PRE}{F_{osc}}$ així que calc nº ticks per fer $\frac{1}{880}$ seg.

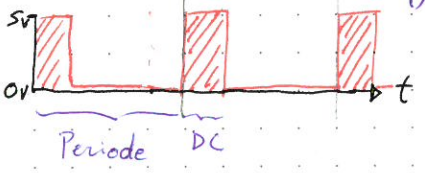
$\frac{1}{880} \div \frac{4\ PRE}{F_{osc}} = \overset{\text{Tanteig}}{[PRE = 4]} \cong 2273$ ticks.

Per configurar-ho és tot igual amb 2 mod.:

- NO fa falta iniciar TMRxIE
- En la ISR, dins CCPxIE, s'ha de fer `CCPx += num_ticks;` (No passa res. pq OVF pq tots dos TMRx, CCPx. fan ovf.)
- Iniciar TMRx a 0 per ser rigurosos (Sino simplement fallem la primera).

PWM TMR2/4/6

Permet generar un senyal digital a una freq. constant però variant el període.



- Periode: Relacionat amb Freq o temps → PRx
- DC: Percentatge temps que està "ON". CCPRx $\frac{t_{ON}}{T} \cdot 100$

Exemple Ens diuen: $F_{osc} = 8$ MHz, $F_{PWM} = 200$ kHz, DC % = modular

- Càlculs: 1. Primer hem de saber el Periode del PWM (Formula al Formulari)

$$\frac{1}{200 \times 10^3} = (PR2+1) * 4 * \frac{1}{8 \times 10^6} * TMR2\_PRE \Rightarrow PR2 = 9$$

(1 tanteig)

⚠ RECORDEM: PRx de 8b i PRE sempre més petit.

2. Ara que sabem el periode, necessitem saber quanta estona ON. (Si 100%)

Pulse Width = DCV * Tosc * TMR2_PRE = $1/F_{PWM}$ # Està en form

$DCV = \frac{1}{200 \times 10^3} \cdot \frac{8 \times 10^3}{1 * PRE} = 40$ Però aquest és el de 100%, i el nostre modular.

3. Calculem el DCV amb regla de 3

$DCV = (40 * modular)/100$ on modular és un valor de %.

40 — 100%
DCV — modular %

⚠ IMPO: CCPxM = 1100, no 11xx.

OBS: Donat que no podrem dividir entre 2, afegim 2b (Multi per 4) i per això tenim CCPRxL : CCPxCON<5:4>.

```
CCPRxL = (DCV >> 2) & 0xFF;
CCPxCON bits. DCxB = DCV & 0x03;   } IMPO
```