**Configuración del TIMER0**

**1. Introducción:**

El Timer/Counter es un dispositivo que cuenta los ciclos de reloj de la señal que se le da como entrada, en función del origen esta señal actúa como:

- TIMER: Si utiliza el reloj que alimenta los periféricos (Peripherial Clock PCLK)

- COUNTER: Si utiliza un reloj externo.

- Sea cual sea su señal de entrada, cuenta los ciclos de la misma en un registro de 32 bits

Para seleccionar la señal de entrada usamos el registro

- El TIMER se puede resetear y parar mediante el registro

- Posee un **Preescaler**: Un registro de 32 bits que sirve para disminuir la frecuencia del reloj de entrada al TIMER. La señal de entrada va primero al Prescaler y cuando este alcance el valor establecido es entonces cuando sube el contador :



- Cada TIMER posee 4 **registros de**  () de 32 bits que controlamos nosotros (R/W).

Podemos configurar el para que cuando el contador sea igual a alguno de estos registros, (evento de match) se produzcan las siguientes acciones:

- Interrupción del del TIMER.

- Reset del TIMER

- Parada del TIMER

La configuración de estas acciones se realiza mediante el registro

- Cada registro MATCH () además posee una señal lógica de salida asociada llamada .

Podemos configurar un GPIO para que dé a su salida dicha señal .

Podemos configurar el correspondiente para que cuando se produzca la igualación (match), se realice vía Hardware alguno de estas operaciones: - MAT = 1

- MAT = 0

- MAT ^= 1 🡪 Se invierte

Estas acciones son llevadas a cabo por Hardware (sin retardos) y se configuran con

- Cada TIMER posee además 2 dispositivos de captura (Capture channels).

Podemos configurar los GPIO como entrada de estos capturadores y que cuando se produzca un flanco de subida o bajada: - Se produzca una interrupción

- Se guarde por Hardware el valor del TC en el registro

La configuración de estas acciones se realiza mediante el registro

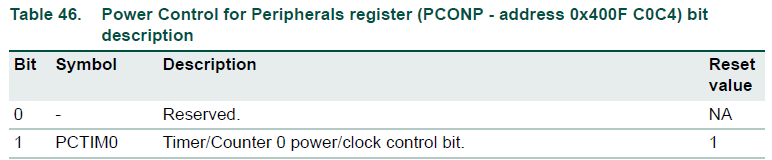
- Cuando se produce una interrupción, esta puede ser debida a cualquiera de los 4 o de los 2 , para saberlo, tenemos el registro de interrupción que nos indica cual ha sido.

A continuación enseñaremos como configurar el TIMER0, el proceso es análogo para los TIMERS 1/2/3 teniendo cuidado con los registros que son independientes de cada TIMER y aquellos que comparten entre todos

**2. Procedimiento de configuración:**

**1) Alimentación del TIMER0 y establecimiento como TIMER:**

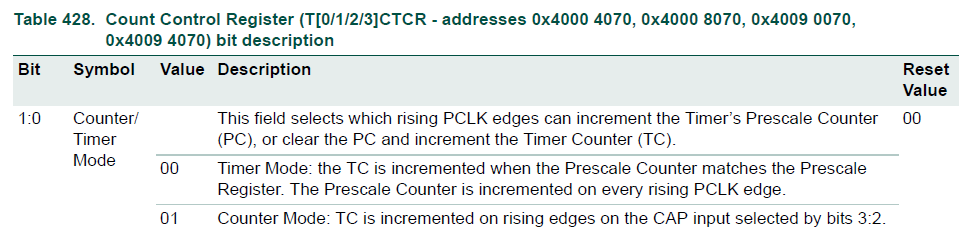
Si el bit de control de un periférico está a 1, este está activado y si está a 0, está desactivado



**LPC\_SC->PCONP |= (0x01<<1);**

Cuando hay un Reset, el TIMER0 está activado, pero por si acaso otra función lo utiliza lo activamos siempre.

- Podemos establecer el temporizador como TIMER o COUNTER mediante el registro CTCR Count Control Register. Nosotros lo estableceremos como TIMER

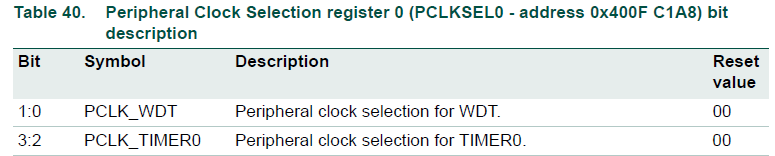


**LPC\_TIM0->CTCR &= ~ (3 << 0);**

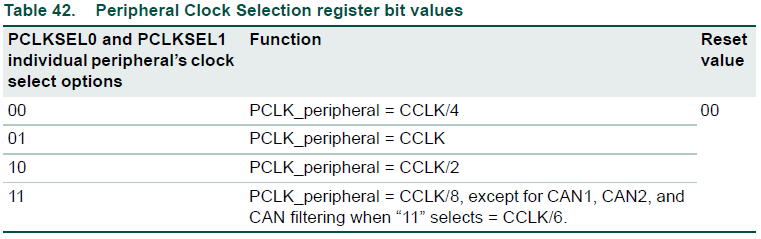
**2) Establecimiento de la frecuencia del Reloj del TIMER0:**

La frecuencia del temporizador dependerá del PCLK y del Prescaler:

- El registro PCLKSEL0 posee los bits de configuración del PCLK del TIMER0:



- Los significados de 2 bits son:



- Para coger la frecuencia de 25 MHz 🡪 CCLK / 4

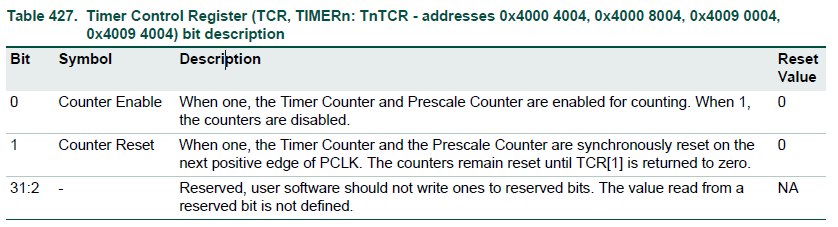
**LPC\_SC->PCLKSEL0 &= ~ (3 << 2);**

- Para que el reloj trabaje a 1 MHz dividimos la frecuencia de 25 MHz mediante el Prescaler Register del TIMER0.

**LPC\_TIM0->PR = 24;**

**3) Reset y Enable del Temporizador**

Para el control del Reset y Enable del temporizador usamos el registro TCR[1:0]:



Al hacer el RESET, se resetea tanto el PC como el TC.

Para resetear el TC y pararlo hacemos:

**LPC\_TIM0->TCR = 0x10;**

Cuando queramos que se inicie la cuenta:

**LPC\_TIM0->TCR = 0x01;**

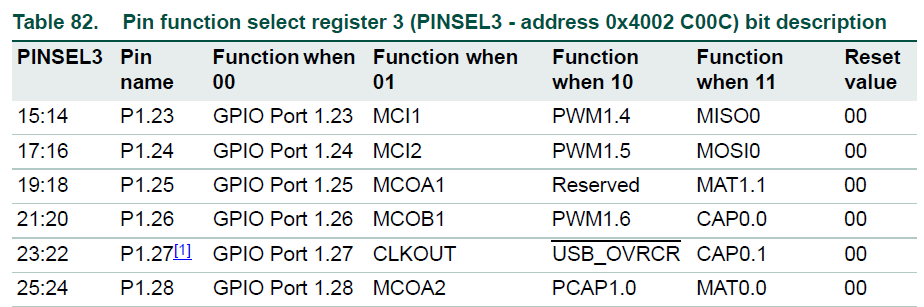
- A la hora de configurar nuestro TIMER, lo normal es pararlo y resetearlo al principio del todo, configurarlo a nuestro gusto y después activarlo.

**4) Selección de los pines del TIMER0:**

Cada pin GPIO puede tener 4 funciones diferentes, pero sólo 1 a la vez, dichas funciones se seleccionan mediante el registro PINSEL.

Debemos encontrar los GPIO que pueden actuar como MAT0 (salidas) y CAP0 (entradas) del TIMER0 y activar esta funcionalidad.

Viendo las tablas, algunos GPIO con esta funcionalidad se activan mediante el PINSEL3:

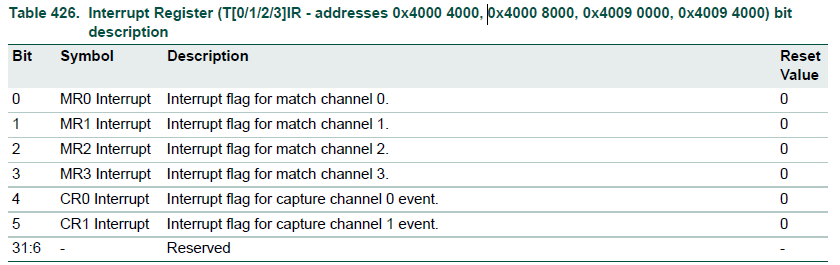


**LPC\_PINCON->PINSEL3 |= (3 << 20);** // Select CAP0.0 (P1.26) pin for input.

**LPC\_PINCON->PINSEL3 |= (3 << 24);** // Select MAT0.0 (P1.28) pin for output match.

**5) Activamos las interrupciones del TIMER0**

El TIMER0 tiene muchas fuentes de interrupción que son atendidas mediante una misma función, para saber qué ocasionó la interrupción miramos el registro de interrupción IR



- Las primeras 4 interrupciones corresponden a la situación "Match" en la que

Los son registros especiales en los cuales podemos escribir un valor y cuando se dé el Match se puede activar un reset o disenable del TC y/ó una interrupción del TIMER0.

- Las últimas 2 son las interrupciones que se dan cuando capturamos un evento de entrada con los capturadores CAP0 y CAP1.

- Para activar y configurar estas interrupciones habrá que activar ciertos bits de otros registros.

· MCR: Para los eventos de TC = MRn

· CCR: Para los eventos de captura de CAP0 y CAP1

- Cuando se produce una de estas interrupciones, el bit correspondiente del IR se pone a 1

Para resetear la interrupción escribimos un '1' en dicho bit.

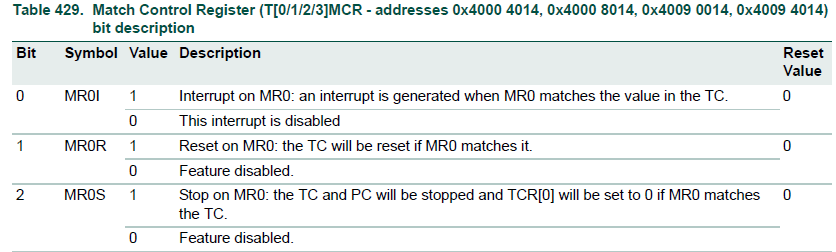
- Primero damos prioridad y activamos la función de atención a la interrupción del TIMER0

**NVIC\_SetPriority(TIMER0\_IRQn,0x01);**

**NVIC\_EnableIRQ(TIMER0\_IRQn);**

**6) Configuramos los registros de MATCH, MCR (Match Control Register):**

Este registro se utiliza para controlar las operaciones que se realizan cuando el TC se iguala a uno de los .



- Para activar la interrupción cuando

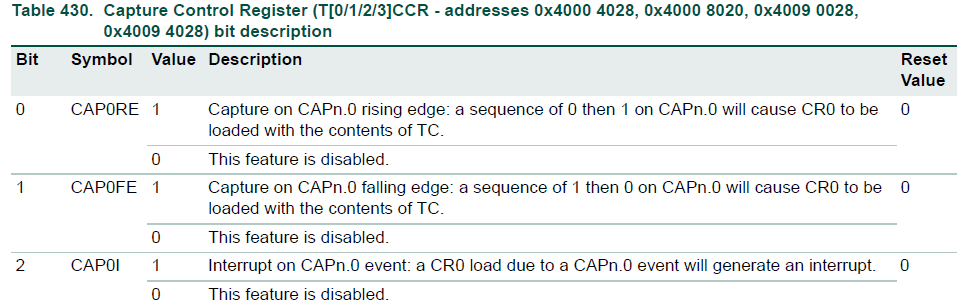
**LPC\_TIM0->MCR |= (1 << 0);**

- Si queremos que el TC se resetee cuando escribiríamos:

**LPC\_TIM0->MCR |= (1 << 1);**

**7) Configuramos los capturadores CCR (Capture Control Register):**

Este registro se utiliza para controlar las operaciones que se realizan cuando se produce un evento de captura (flanco de subida y/o bajada según esté configurado).



- Para que se detecte flanco de subida en CAP0:

**LPC\_TIM0->CCR |= (1 << 0); // Also makes: LPC\_TIM0->CR0 = TC**

- Para que se detecte flanco de bajada en CAP0:

**LPC\_TIM0->CCR |= (1 << 1); // Also makes: LPC\_TIM0->CR0 = TC**

- Para activar la interrupción en un evento de captura en CAP0:

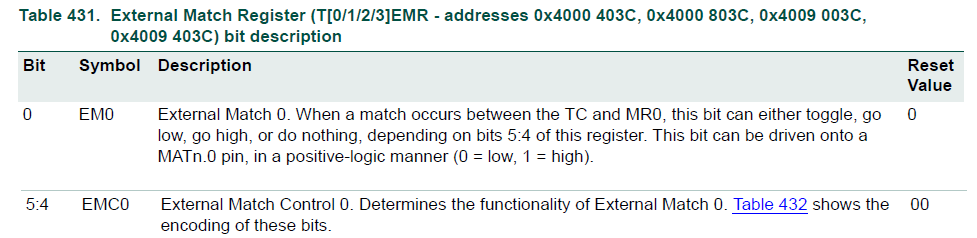
**LPC\_TIM0->CCR |= (1 << 2);**

En la función de atención a la interrupción tendremos que mirar este registro para saber si ha sido un flanco de subida o de bajada y cambiarlo si queremos.

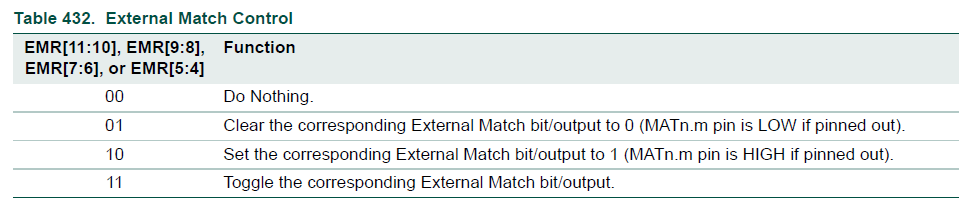
**8) Configuramos el pin de salida MAT0.0 mediante el registro EMC External Match Register.**

Cada uno de los 4 comparadores que posee cada TIMER posee un pin de salida MATn.m cuyo valor puede cambiar o no cuando se produce un Match con el contador .

Para configurar las operaciones que se realizan sobre el MATn.m cuando se da el Match, se usa este registro.



- Si ponemos EM0 a 1, cuando se produzca el match TC = MR0 se podrá producir un evento sobre la salida MAT0.0. Este evento se podrá controlar con los bits de EMC0



También podemos poner el valor que deseemos a la salida de TIM0.0 con el EM = 0

- Ejemplo para poner MAT0.0 a 1 hacemos.

**LPC\_TIM0->EMR &= ~ (1 << 0); LPC\_TIM0->EMR |= (1 << 5); LPC\_TIM0->EMR &= ~ (1 << 4);**

Esto hace EM0 = 0 y EMR0 = 10

- Ejemplo para poner un 0 en MAT0.0 cuando TC = MRC:

**LPC\_TIM0->EMR |= (1 << 0); LPC\_TIM0->EMR |= (1 << 4); LPC\_TIM0->EMR &= ~ (1 << 5);**

Esto hace EM0 = 1 y EMR0 = 01.

**3. Ejemplo configuración del TIMER0:**