**1. Configuración del PWM1 (PULSE WIDTH MODULATOR)**

**1. Introducción:**

El PWM es un módulo TIMER que cuenta con funcionalidades añadidas para una fácil generación de señales periódicas TTL de periodo, ciclo de trabajo y desfase deseados.

- Al igual que el TIMER puede ser configurado como TIMER o COUNTER con , se puede resetear y parar con y posee un prescaler configurable con

- El PWM posee 7 **registros de**  () de 32 bits.

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 PWM.

- Reset del TIMER

- Parada del TIMER

La configuración de estas acciones se realiza mediante el registro

Cuando modificamos el valor de estos registros vía software, su valor no cambia instantáneamente sino que es introducido en un Latch, para hacer el cambio efectivo tenemos que escribir un 1 en la posición correspondiente del registro

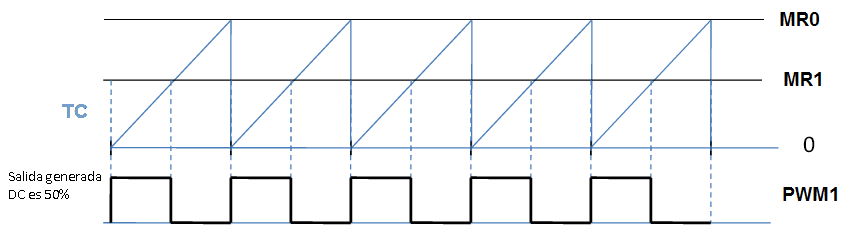
- Los últimos 6 registros de MATCH (), tienen asociado cada uno una señal de salida respectivamente, es decir posee 6 canales .

Es aquí donde reside la principal particularidad del PWM respecto a otro TIMER.

Estas 6 salidas se utilizan para generar la señal TTL deseada de la siguiente manera:

- Cuando el PWM se resetea sus señales PWM están a nivel alto 🡪

- Cuando se produce un match entre el TC y el MR correspondiente 🡪



· Frecuencia controlada con MR0 🡪 Provocando un reset en match.

· Ciclo de trabajo controlado con MR[1-6] respecto a MR0 🡪

🡪 Este es el modo flanco simple de los PWM, en el modo flanco doble usamos 2 MR, y la salida del PWM correspondiente obedece a unas determinadas reglas.

Podemos configurar un GPIO para que dé a su salida dichas señales .

La configuración del modo de funcionamiento y el Enable de las salidas PWM se hace con

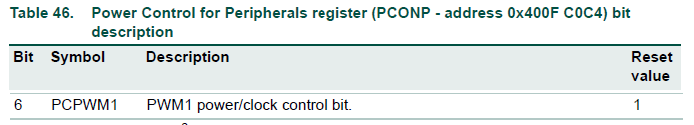
- El PWM posee además 2 capturadores de mismo funcionamiento al de los TIMER.

- Cuando se produce una interrupción, esta puede ser debida a cualquiera de los 7 o de los 2 , para saberlo, tenemos el registro de interrupción que nos indica cual ha sido.

**2. Procedimiento de configuración:**

**1) Encendido del PWM1 y establecimiento como TIMER:**

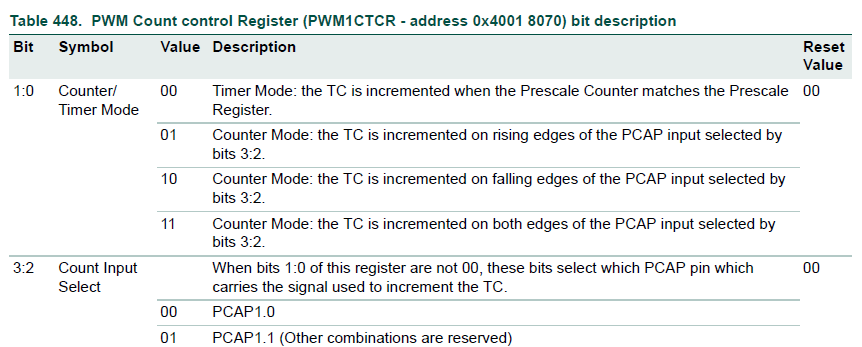
Si el bit de control de un periférico está a 1, este está activado y si está a 0, está desactivado



**LPC\_SC->PCONP |= (1 << 6);**

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

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



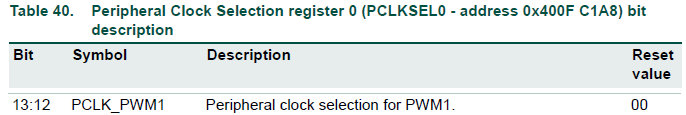
Para ponerlo en modo temporizador normal:

**LPC\_PWM1->CTCR &= ~ (3 << 0);**

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

La frecuencia del PWM dependerá del PCLK y del Prescaler:

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



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

**LPC\_SC->PCLKSEL0 &= ~ (3 << 12);**

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

**LPC\_PWM1->PR = 24;**

**3) Selección de los pines del PWM y su modo de funcionamiento:**

El PWM tiene 6 pines de salida por lo que podemos generar:

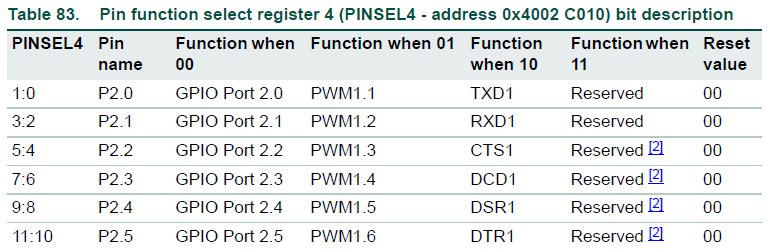
- 6 señales de flanco simple ó

- 3 señales de flanco doble ó

- Una combinación de ambas

Viendo las tablas, los pines se activan mediante el PINSEL4.

(Hay mas pines que pueden usar estas funcionalidades)

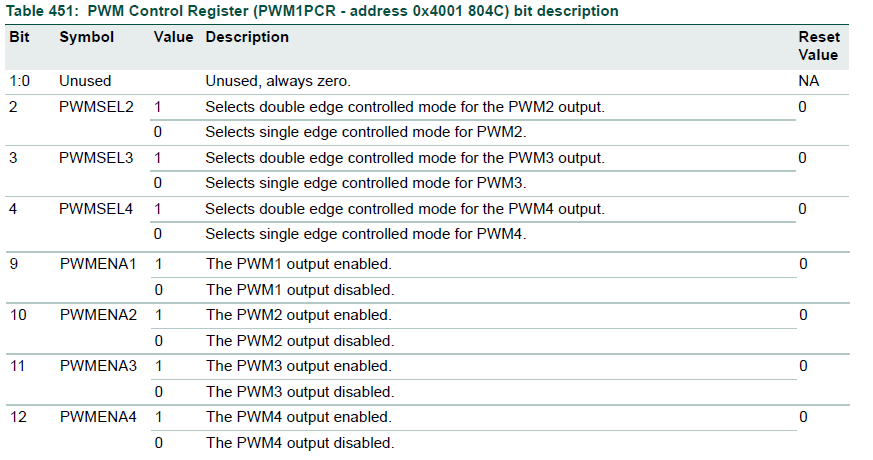


Ejemplo activar PWM1.3 y PWM1.4:

**LPC\_PINCON->PINSEL4 |= ( 1 << 4 ); // PWM1.3 on P2.2**

**LPC\_PINCON->PINSEL4 |= ( 1 << 6 ); // PWM1.4 on P2.3**

- Para elegir entre la salida en modo flanco simple o doble y para activar las salidas usamos el registro **PWM Control Register**

****

Ejemplo habilitación del PWM3 en modo simple:

**LPC\_PWM1->PCR &= ~ (0x1<<3); // Nos aseguramos que sea flanco simple**

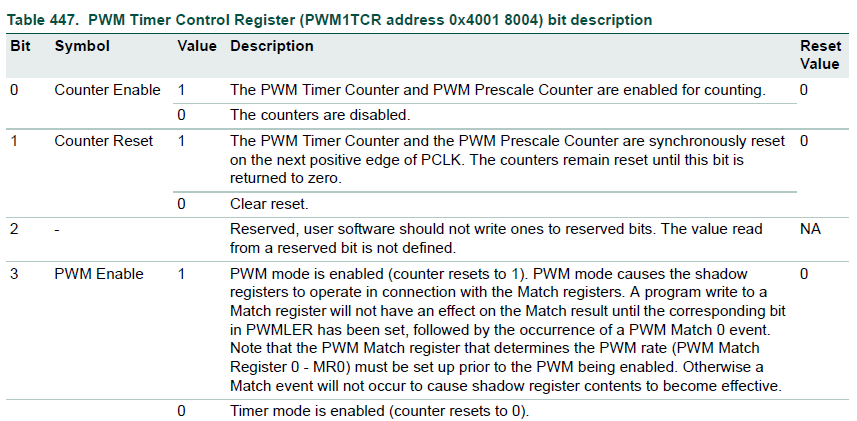
**LPC\_PWM1->PCR |= (0x1<<11); //Habilita la salida del PWM3**

**4) Reset, Enable y activación del PWM**

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

El PWM puede actuar como TIMER o como PWM, para utilizarlo como PWM usamos

el bit TCR[3], PWM Enable, poniéndolo a 1.



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

- Para resetear el TC y pararlo hacemos:

**LPC\_PWM1->TCR = 0x10;**

- Cuando queramos que se inicie la cuenta:

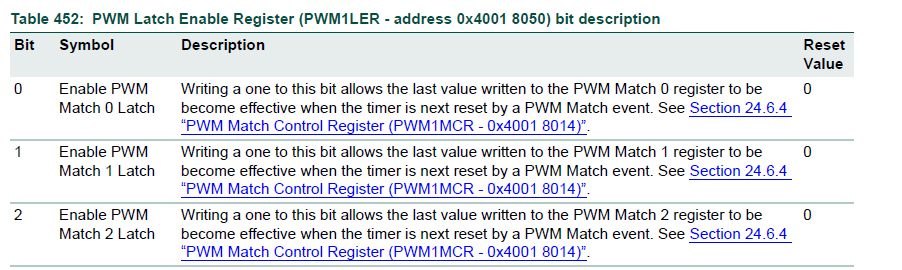
**LPC\_PWM1->TCR = 0x01;**

- Para activar el modo PWM:

**LPC\_PWM1->TCR |=( 0x01 << 3);**

**5) Configuramos el valor de los MR:**

Cuando escribimos un valor en alguno de los registro dicho registro no cambia inmediatamente si no que tenemos que activar un bit (latch) mediante el cual, en al proxímo reset se realizaran los cambios pertinentes. Este es el **PWM Latch Enable Register**



Con el MR0 establecemos la frecuencia y con el MR1 el ciclo de trabajo.

Por ejemplos si establecemos valores para MR0 y MR1 y los queremos hacer efectivos:

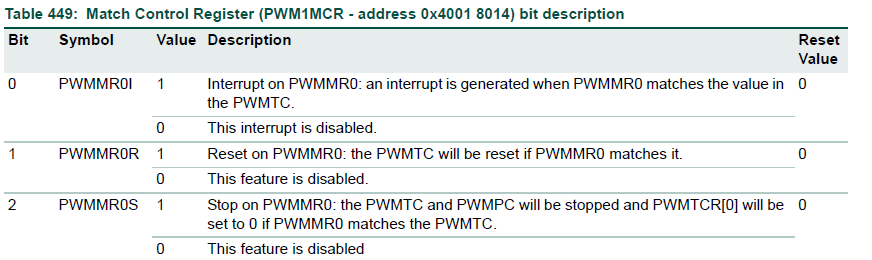
**LPC\_PWM1->MR0 = PERIOD\_PWM;**

**LPC\_PWM1->MR1 = PERIOD\_PWM/2;**

**LPC\_PWM1->LER |= (0x1<<0) | (1<< 1); /\* PWM 0 y 1 latch enabled \*/**

**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 .



Hay un trío de estos bits por cada uno de los 7 MR.

- Para activar la interrupción cuando

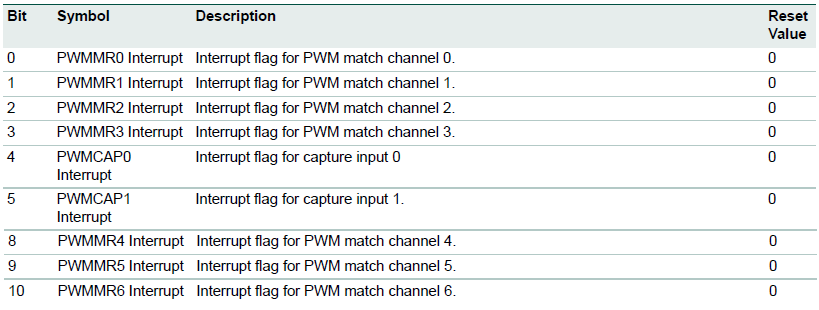
**LPC\_PWM1->MCR |= (1 << 3) ;**

- Si queremos que el TC se resetee cuando escribiríamos:

**LPC\_PWM1->MCR |= (1 << 1);**

**7) Activamos las interrupciones del PWM**

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



- Los bits corresponden a la situación "Match" de los registros MR 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 disable del TC y/ó una interrupción del TIMER0.

- Los bits corresponden a los eventos de captura de 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(PWM1\_IRQn, 0x01);**

**NVIC\_EnableIRQ(PWM1\_IRQn);**

**3. Ejemplo configuración del PWM1:**