

Temporizadores LPC17xx



Índice

- ❑ Introducción
- ❑ Temporizadores
- ❑ PWM
- ❑ Watchdog
- ❑ Referencias

Ejemplos de señales que requieren temporización

□ Sensor de distancia por ultrasonidos SRF-04 (0,3 a 3m)

- El sensor recibe un pulso de disparo y responde con otro pulso cuya anchura es proporcional a la distancia del objeto.

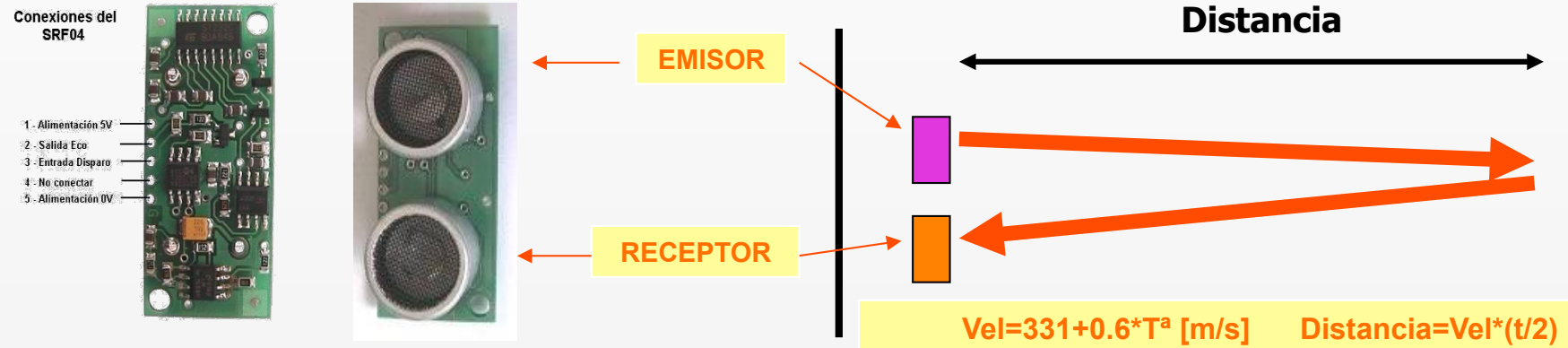
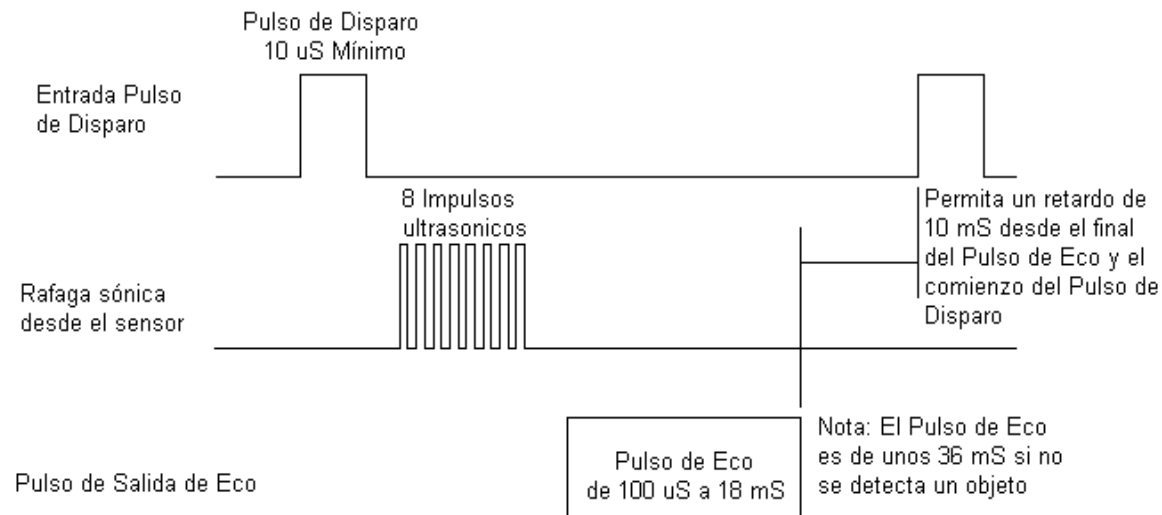


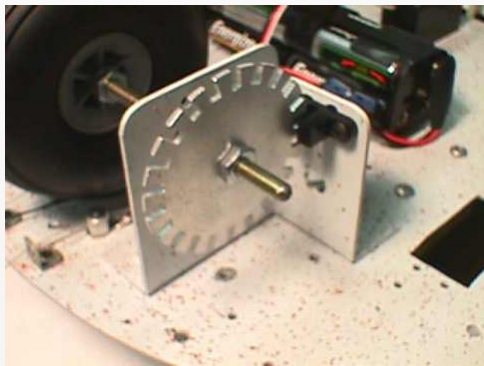
Diagrama de Tiempos del SRF04



Ejemplos de señales que requieren temporización

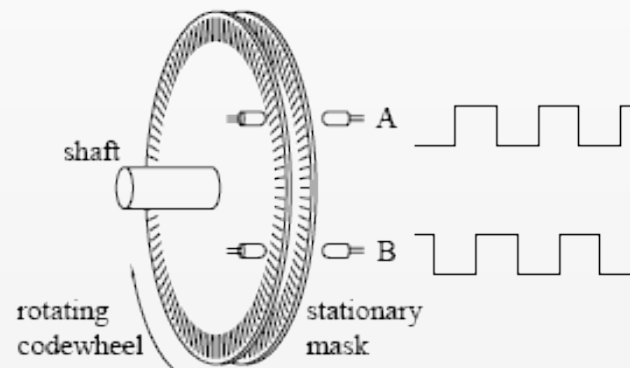
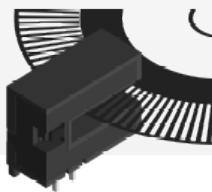
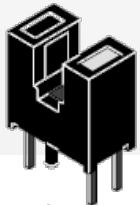
Encoder óptico incremental

- Según va girando el motor va generando pulsos en cuadratura.
- Se puede medir el espacio recorrido o la velocidad

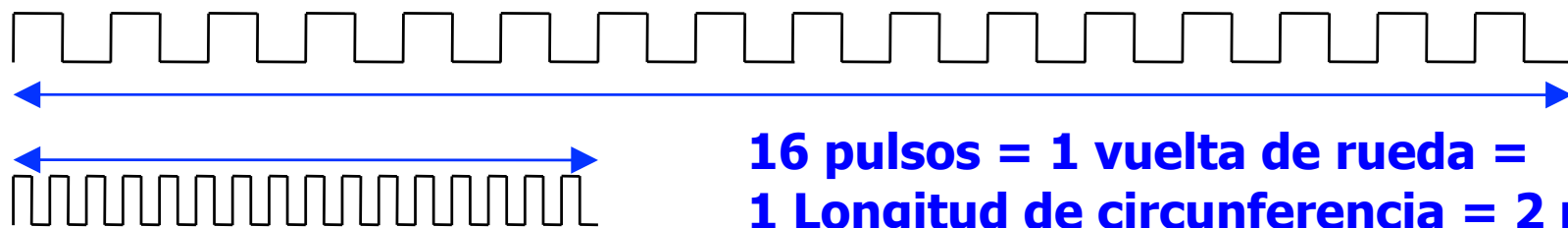
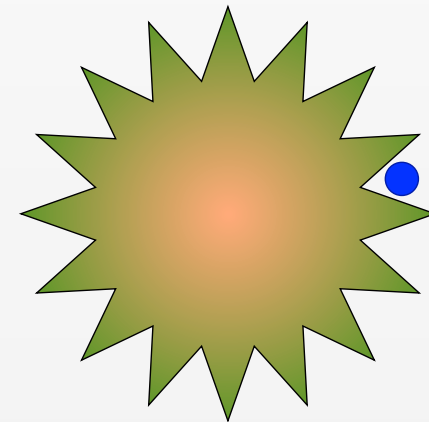


TCST1030

TCST5250



Engranaje de 16 dientes



$$\text{Velocidad} = \text{pulsos} / \text{tiempo} \quad \left\{ \begin{array}{l} N \text{ pulsos} / \text{Tiempo que tardan los } N \text{ pulsos} \\ N \text{ pulsos medidos} / \text{Tiempo fijo} \end{array} \right.$$

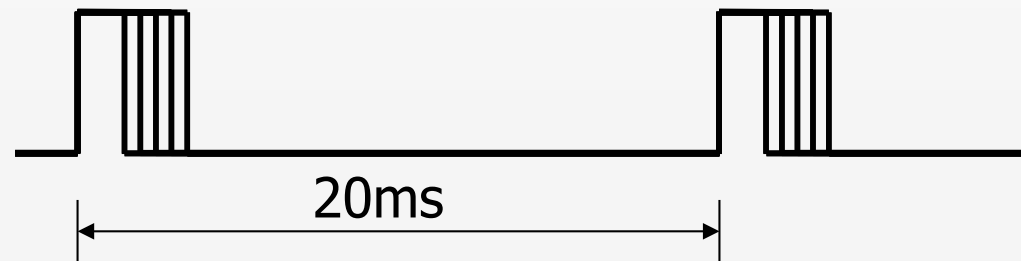
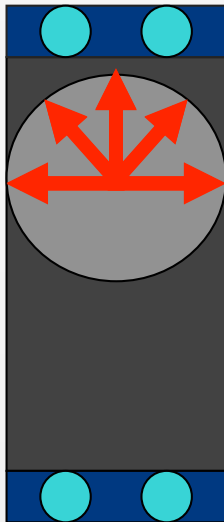
tiempo

Ejemplos de señales que requieren temporización

□ Servomotor de radiocontrol

- El servomotor mantiene una posición determinada del eje
- Se controla con el tiempo en alto de una señal periódica

Tiempo a nivel alto: de 1 a 2ms (depende de las marcas)

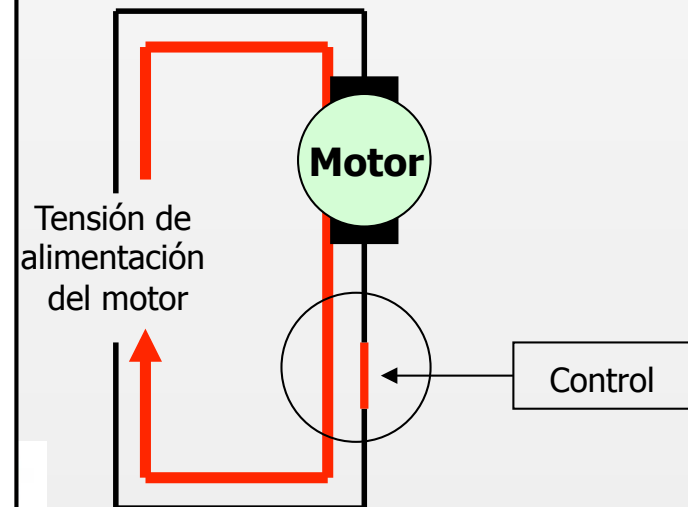
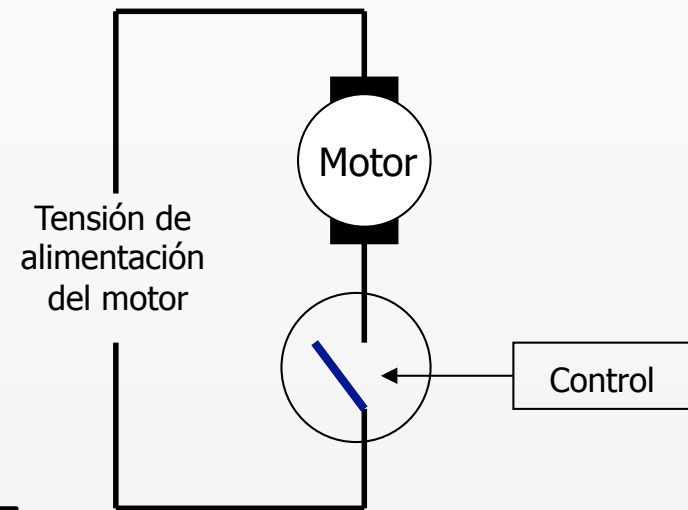
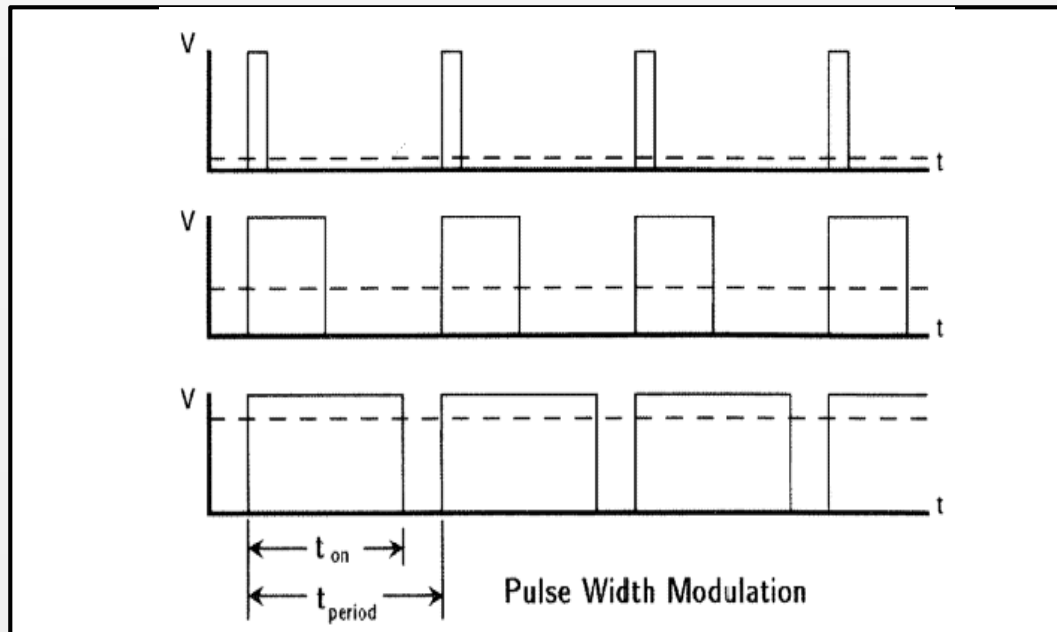


Ejemplos de señales que requieren temporización

❑ Señal PWM de control de un motor de corriente continua

DRIVER DE POTENCIA

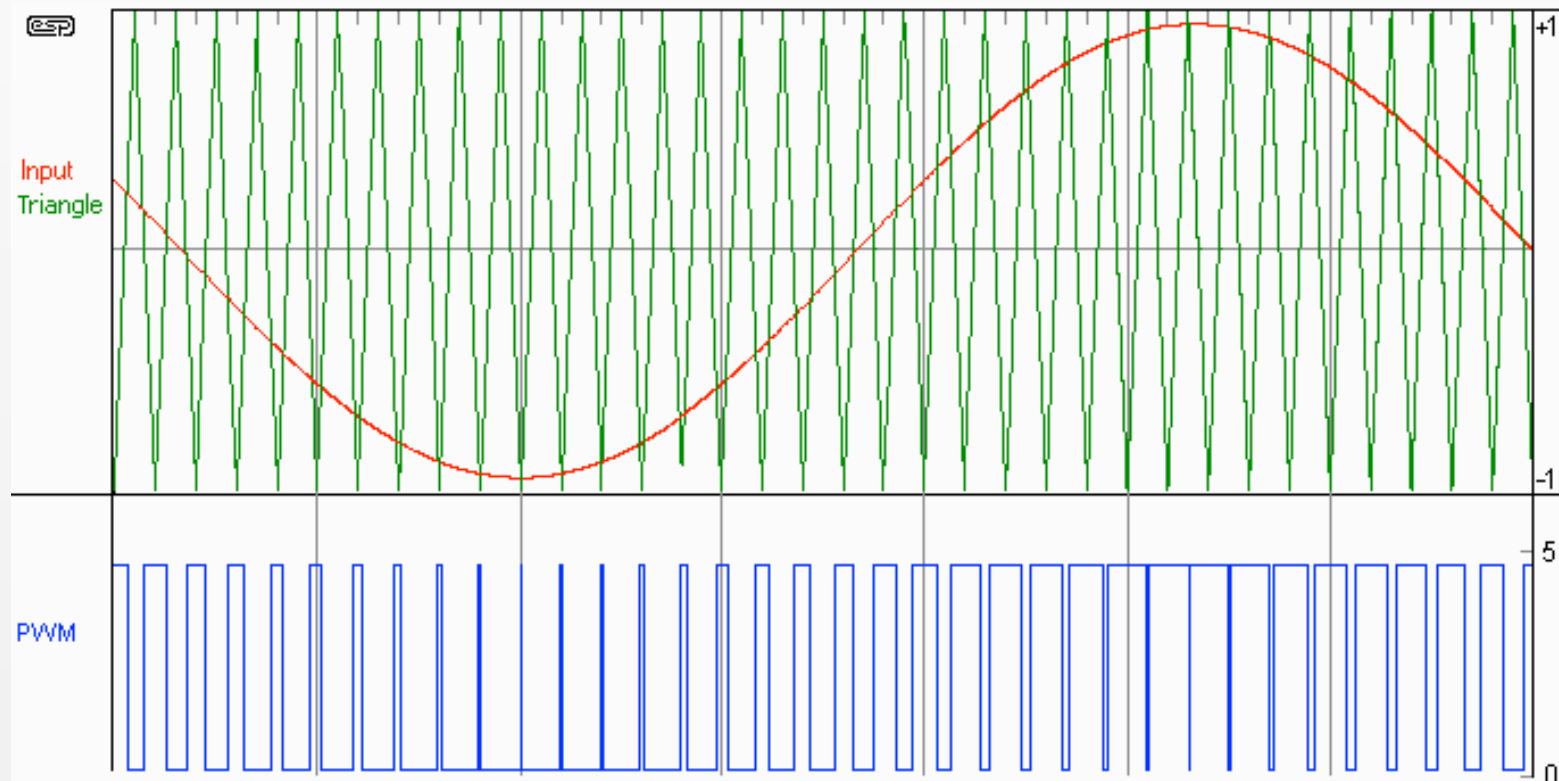
- Los circuitos digitales no pueden generar la corriente que necesita un motor para funcionar
- Se utiliza un circuito de potencia adaptador (driver)
- El driver puede estar formado por componentes discretos o por un circuito integrado
- En control digital los drivers se comportan como interruptores



Ejemplos de señales que requieren temporización

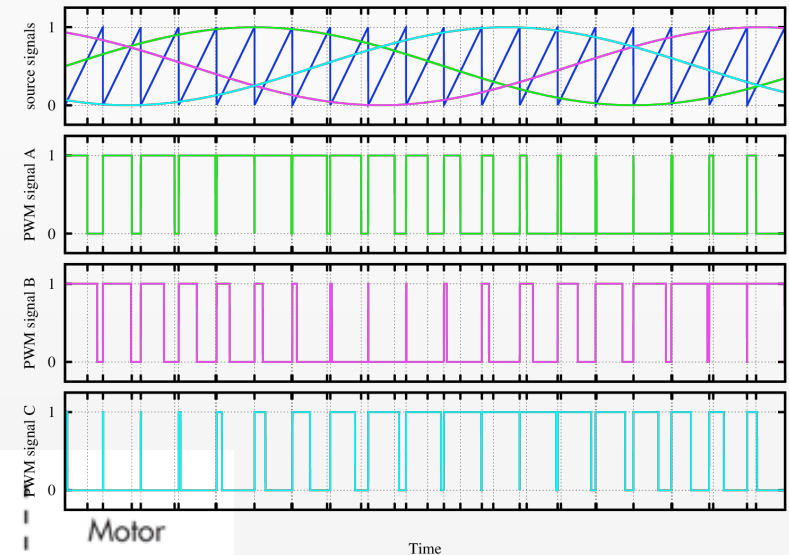
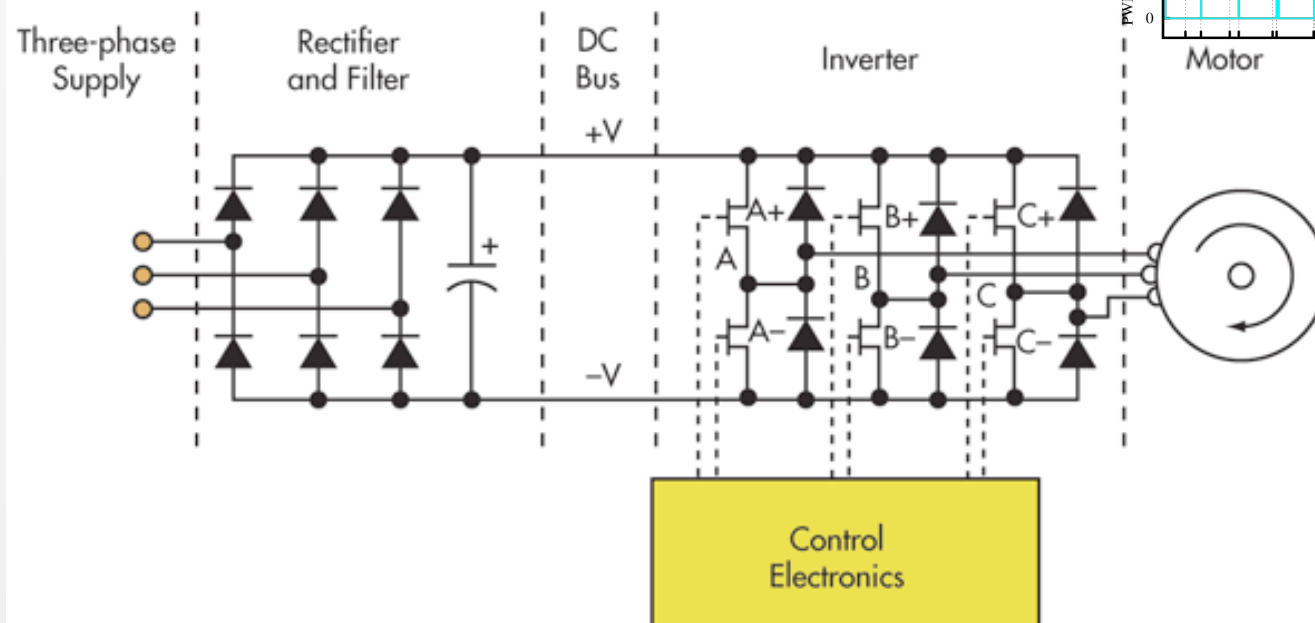
□ Generador de señales

- Se puede obtener una señal analógica con una salida PWM y un filtro paso bajo



Ejemplos de señales que requieren temporización

Control de un variador de frecuencia para control de velocidad de un motor trifásico



<http://powerelectronics.com/motion-systems/power-analysis-pwm-motor-drives>

<http://epccs.org/indexes/Document/nVFD/>

Introducción

□ Pueden emplearse para:

- Medir tiempos (retardos, etc.)
- Medir señales externas.
- Capturar eventos externos (flancos de subida y/o bajada)
- Generar señales
- Interrupciones periódicas
- Modular información
 - PWM (Pulse Wide Modulation)
 - PPM (Pulse Position Modulation)

Introducción

□ Básicamente se componen de:

- Una fuente de reloj (interna o externa)
- Un contador y un pre-escaler.
- Lógica para generar interrupciones, reset del contador, etc.
- Registros de captura y comparación

□ Los timers son módulos especializados en la medida del tiempo con precisión ajustable, desde ns a s.

- Por ejemplo, con 32 bits de contador, alimentado a 100MHz se puede temporizar, desde 10ns hasta 42,92s.
- Para poder medir tiempos mayores, se incluye un divisor antes del contador (preescaler).
 - Con un preescaler de 32 bits, se podrían medir hasta $42,92 \cdot 2^{32}$ s

Introducción

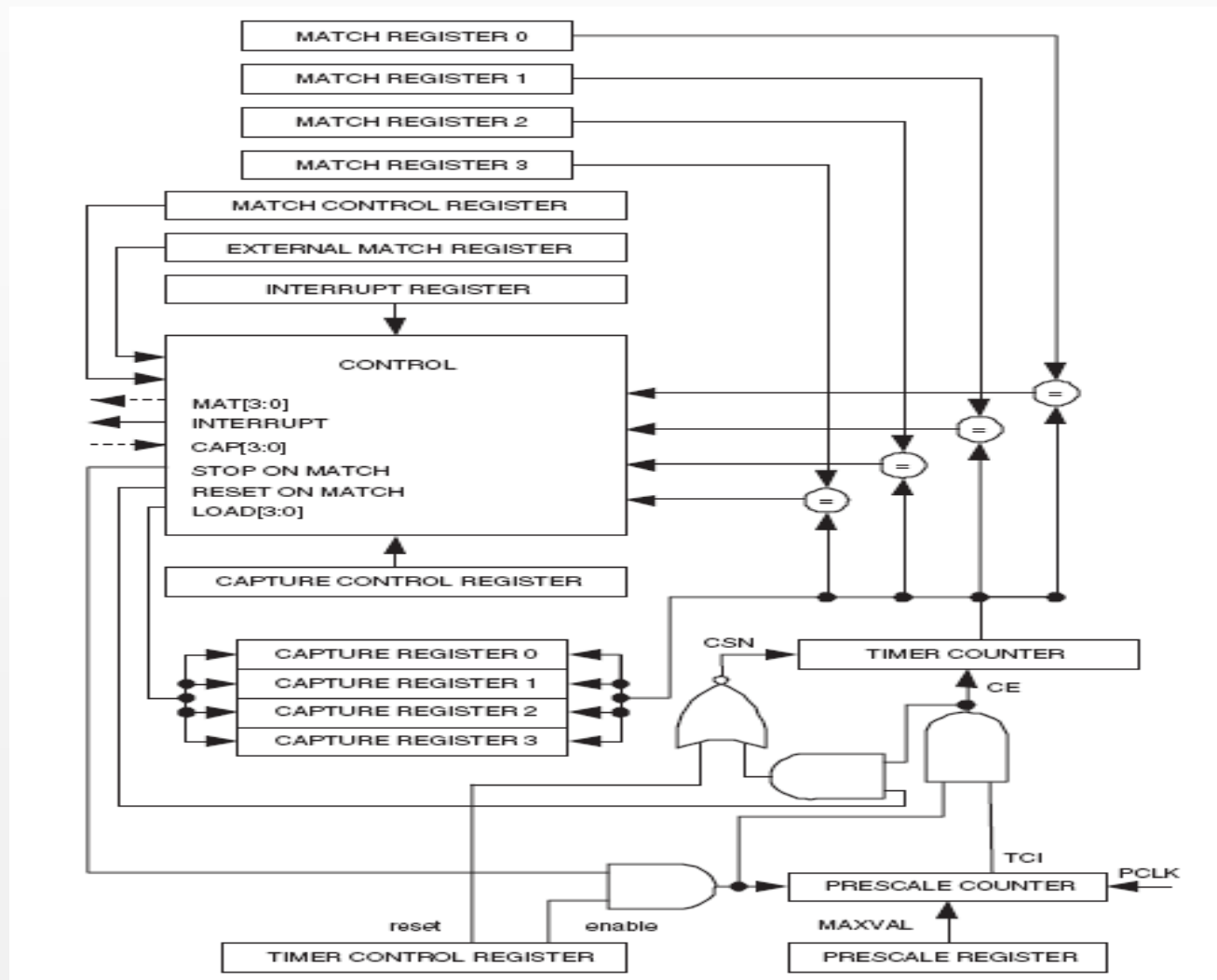
❑ El LPC1768 dispone 8 timers:

- Generales: timer 0, 1, 2 y 3
- 2 módulos de PWM (*timer* con algunas funcionalidades más).
- SysTick y RIT timer

❑ Estructura de los *timers* 0 a 3:

- Contador de 32 bits que se incrementa con:
 - La frecuencia PCLK dividida por el valor de registro preescaler+1 (modo timer)
 - Los flancos configurables de una señal externa (modo contador)
 - Frecuencia máxima externa PCLK/4
- Registros de captura (hasta 2):
 - Para medir señales externas (captura de flancos)
- Registros de comparación y salidas asociadas:
 - Para temporizaciones de amplio rango y alta precisión.
 - Para generar señales, p.ej. Señales PWM.
- Distintas causas de interrupción:
 - Captura de flancos.
 - Cruce de registros de comparación con el contador libre.

Timers: diagrama de bloques



Timers: pines externos

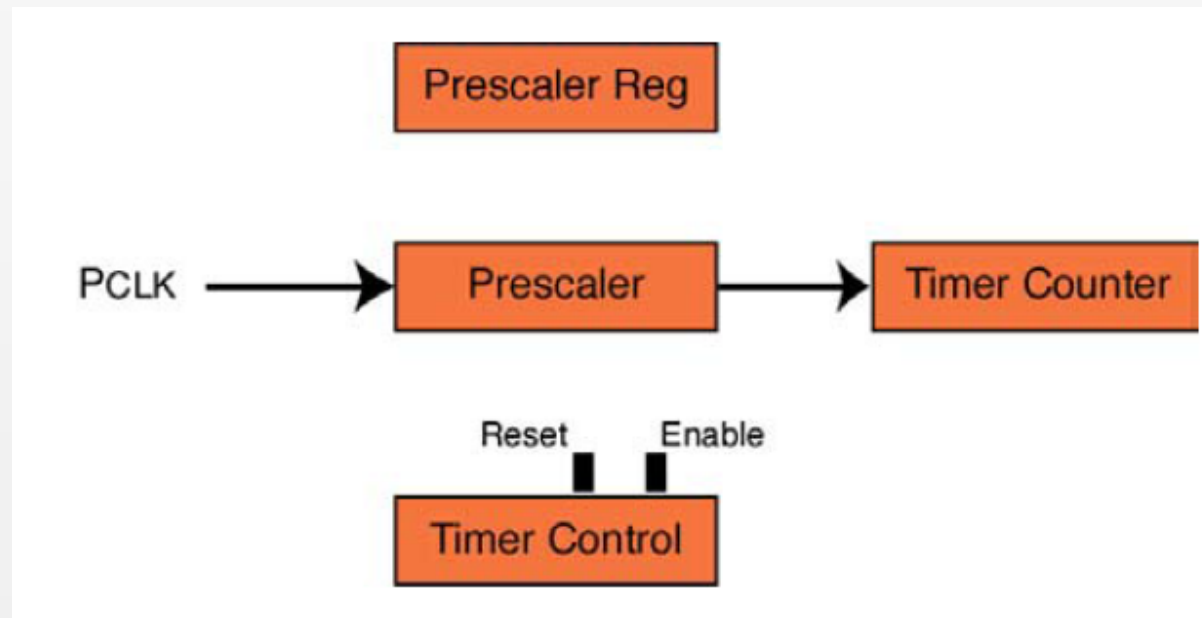
- ❑ Hasta 4 pines de captura y hasta 2 pines de salida.
- ❑ Se configuran con los registros PINSELx

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	00
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	00
9:8	P0.4 ^[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	00
11:10	P0.5 ^[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	00
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	00
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	00
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	00
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	00
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	00
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	00

Timers: el contador

□ Modo temporizador:

- Contador de 32 bits y preescaler de 32 bits.
- $F_{\text{cia timer counter}} = \text{PCLK} / (\text{valor preescaler} + 1)$
- $\text{Tick} = 1 / \text{PCLK} * (\text{preescaler} + 1)$
- $\text{Tick}_{\text{max}} = 1 / \text{PCLK}_{\text{min}} * (\text{preescaler}_{\text{max}} + 1) = 343,597\text{s}$ (para 100MHz CCLK)
- $\text{Tick}_{\text{min}} = 1 / \text{PCLK}_{\text{max}} * (\text{preescaler}_{\text{min}} + 1) = 10\text{ns}$ (para 100 MHz CCLK)



Timers: registros

Registros.

Timer 0

Prescaler
PR: 0x00000000
PC: 0x00000000

Timer
TCR: 0x00000000 ☐ Enable
TC: 0x00000000 ☐ Reset

Interrupt Register
IR: 0x00000000

Match Channels

MCR: 0x00000000 EMR: 0x00000000

MR0: 0x00000000 MR1: 0x00000000 MR2: 0x00000000 MR3: 0x00000000

☐ Interrupt on MR0 ☐ Interrupt on MR1 ☐ Interrupt on MR2 ☐ Interrupt on MR3
☐ Reset on MR0 ☐ Reset on MR1 ☐ Reset on MR2 ☐ Reset on MR3
☐ Stop on MR0 ☐ Stop on MR1 ☐ Stop on MR2 ☐ Stop on MR3

EMC0: Nothing EMC1: Nothing EMC2: Nothing EMC3: Nothing

☐ External Match 0 ☐ External Match 1 ☐ External Match 2 ☐ External Match 3
☐ MR0 Interrupt ☐ MR1 Interrupt ☐ MR2 Interrupt ☐ MR3 Interrupt

Capture Channels

CCR: 0x00000000

CR0: 0x00000000 CR1: 0x00000000

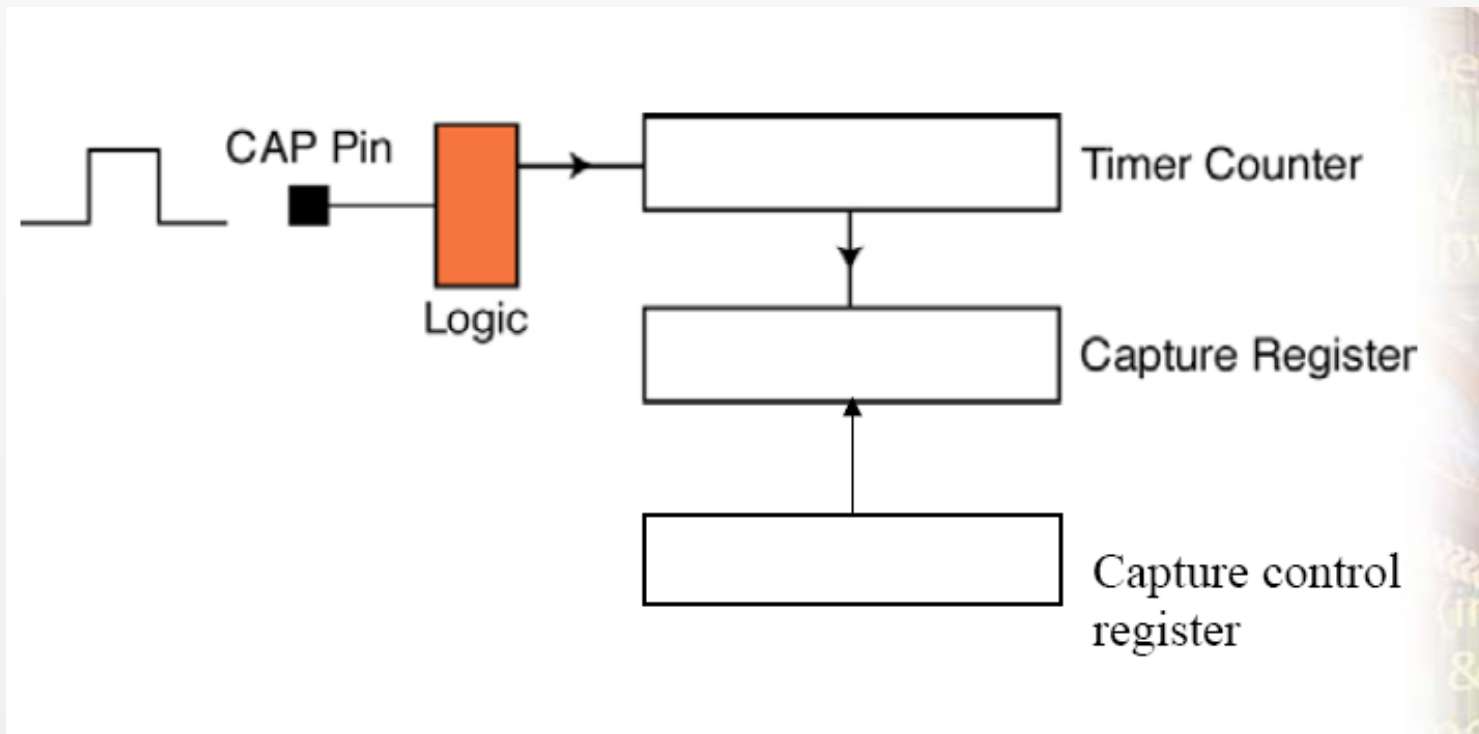
☐ Rising Edge 0 ☐ Rising Edge 1
☐ Falling Edge 0 ☐ Falling Edge 1
☐ Interrupt on Event 0 ☐ Interrupt on Event 1
☐ CAP0.0 ☐ CAP0.1
☐ CR0 Interrupt ☐ CR1 Interrupt

Count Control

CTCR: 0x00000000 Mode: Timer Counter Input: CAP0.0

Timer: Captura de flancos

- Captura configurable de flanco de subida, bajada o toggle.



Timer: Captura de flancos

Registros

Timer 0

Prescaler
PR: 0x00000000
PC: 0x00000000

Timer
TCR: 0x00000000 ☐ Enable
TC: 0x00000000 ☐ Reset

Interrupt Register
IR: 0x00000000

Match Channels

MCR: 0x00000000 EMR: 0x00000000
MR0: 0x00000000 MR1: 0x00000000 MR2: 0x00000000 MR3: 0x00000000

☐ Interrupt on MR0 ☐ Interrupt on MR1 ☐ Interrupt on MR2 ☐ Interrupt on MR3
☐ Reset on MR0 ☐ Reset on MR1 ☐ Reset on MR2 ☐ Reset on MR3
☐ Stop on MR0 ☐ Stop on MR1 ☐ Stop on MR2 ☐ Stop on MR3

EMC0: Nothing EMC1: Nothing EMC2: Nothing EMC3: Nothing

☐ External Match 0 ☐ External Match 1 ☐ External Match 2 ☐ External Match 3
☐ MR0 Interrupt ☐ MR1 Interrupt ☐ MR2 Interrupt ☐ MR3 Interrupt

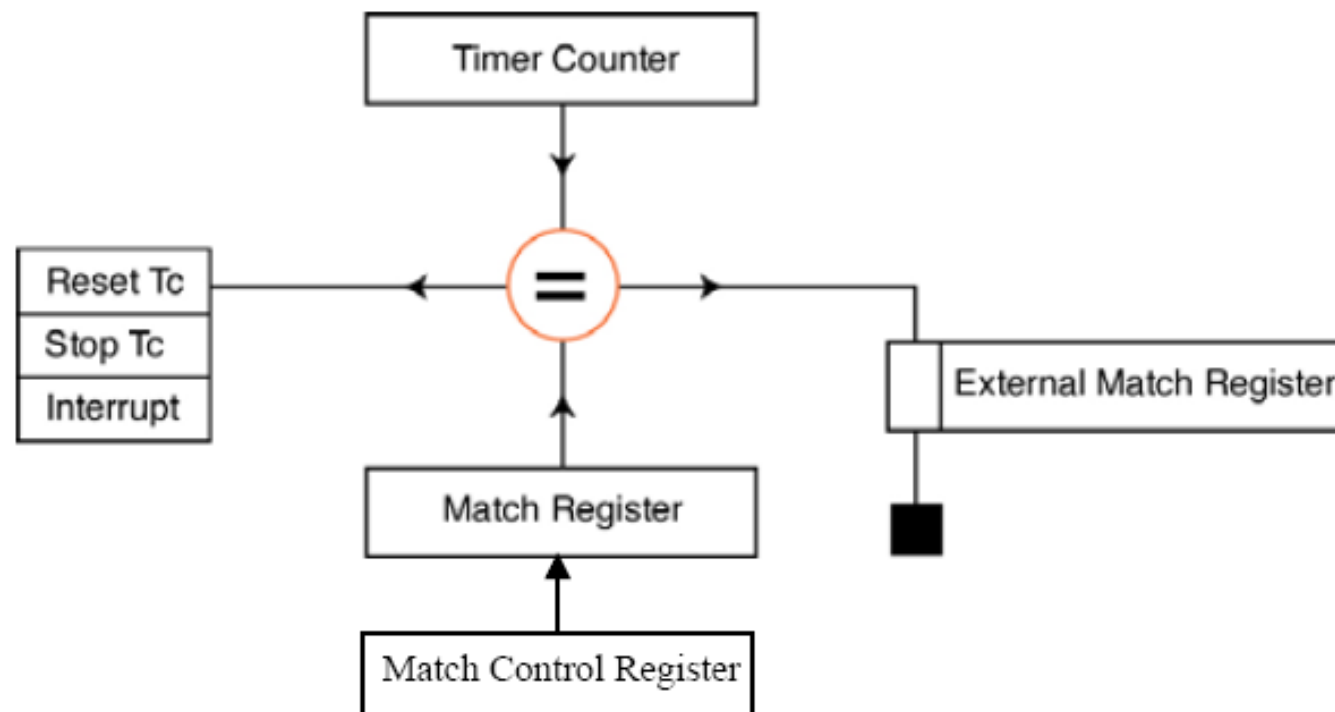
Capture Channels

CCR: 0x00000000
CR0: 0x00000000 CR1: 0x00000000

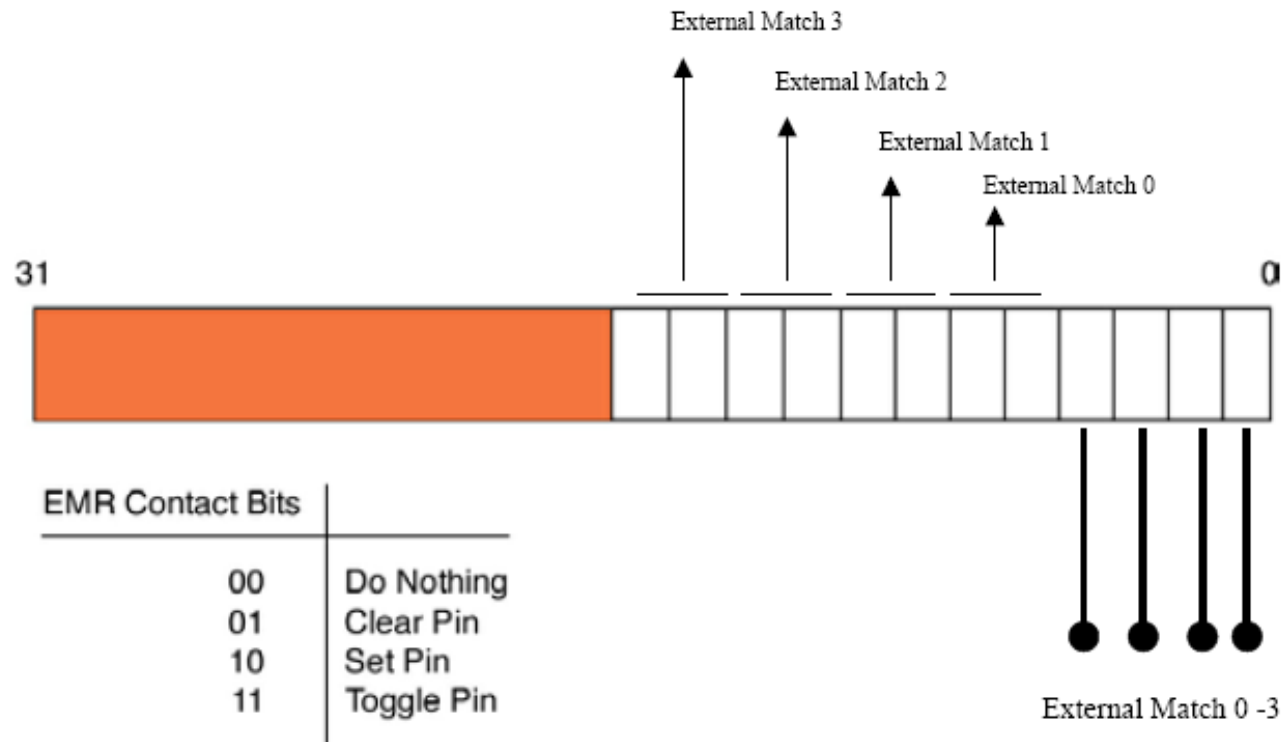
☐ Rising Edge 0 ☐ Rising Edge 1
☐ Falling Edge 0 ☐ Falling Edge 1
☐ Interrupt on Event 0 ☐ Interrupt on Event 1
☐ CAP0.0 ☐ CAP0.1
☐ CR0 Interrupt ☐ CR1 Interrupt

Count Control
CTCR: 0x00000000 Mode: Timer Counter Input: CAP0.0

Timer: Comparación



Comparación: generación de señales



Comparación: registros

Timer 0

Prescaler
PR: 0x00000000
PC: 0x00000000

Timer
TCR: 0x00000000 ☐ Enable
TC: 0x00000000 ☐ Reset

Interrupt Register
IR: 0x00000000

Match Channels

MCR: 0x00000000 EMR: 0x00000000

MR0: 0x00000000 MR1: 0x00000000 MR2: 0x00000000 MR3: 0x00000000

☐ Interrupt on MR0 ☐ Interrupt on MR1 ☐ Interrupt on MR2 ☐ Interrupt on MR3
☐ Reset on MR0 ☐ Reset on MR1 ☐ Reset on MR2 ☐ Reset on MR3
☐ Stop on MR0 ☐ Stop on MR1 ☐ Stop on MR2 ☐ Stop on MR3

EMC0: Nothing EMC1: Nothing EMC2: Nothing EMC3: Nothing

☐ External Match 0 ☐ External Match 1 ☐ External Match 2 ☐ External Match 3
☐ MR0 Interrupt ☐ MR1 Interrupt ☐ MR2 Interrupt ☐ MR3 Interrupt

Capture Channels

CCR: 0x00000000

CR0: 0x00000000 CR1: 0x00000000

☐ Rising Edge 0 ☐ Rising Edge 1
☐ Falling Edge 0 ☐ Falling Edge 1
☐ Interrupt on Event 0 ☐ Interrupt on Event 1
☐ CAP0.0 ☐ CAP0.1
☐ CR0 Interrupt ☐ CR1 Interrupt

Count Control

CTCR: 0x00000000 Mode: Timer Counter Input: CAP0.0

Timer: Configuración inicial

- ❑ **Activar timer en el registro PCON**
 - Ej para el timer0: `LPC_SC->PCONP |= (0x01<<1);`
- ❑ **Configurar pines externos para captura o generación de señales (si se van utilizar).**
- ❑ **Ajustar la frecuencia PCLK para el timer en el registro PCLKSEL0 (por defecto CCLK/4)**
 - Ej: para elegir CCLK en timer 0: `LPC_SC->PCLKSEL0 |= (0x01<<2);`
- ❑ **Configurar modo de funcionamiento timer**
- ❑ **Configurar interrupciones (opcional)**
- ❑ **Escribir rutina ISR (opcional)**
 - `void TIMER0_IRQHandler(void) {`
 - `....`
 - `LPC_TIM0->IR=0x1<<X; borra el flag}`
- ❑ **Habilitar interrupción timer y asignar prioridad (opcional)**
 - `NVIC_EnableIRQ(TIMER0_IRQn);`
- ❑ **Iniciar timer**
 - Ej para timer0: `LPC_TIM0->TCR = 0x01;`

Ejemplo captura flancos

```
/* Captura de flanco subida en CAP0.0 */

void Cap_Init( void )
{
    LPC_SC->PCONP |= 1<<1;          /* enable TIM0 Power
    LPC_PINCON->PINSEL3 = (3<<20); /* set GPIOs for CAP0.0 */
    /* PCLK por defecto*/
    LPC_TIM0->TCR = 0x00000002;      /* Counter Reset */
    LPC_TIM0->PR = 0x00;             /* count frequency: pclk */
    LPC_TIM0->CCR = (0x1<<0)|(0x1<<2); //captura flanco de subida y genera
                                     interrupción

    LPC_TIM0->IR = 0xFF;             /* borra flags interrupción
    NVIC_EnableIRQ(TIMER0_IRQn);
    LPC_TIM0->TCR = 1;               /* Inicia el timer*/
}

/*función de atención interrupción
void TIMER0_IRQHandler (void)
{
    if ( LPC_TIM0->IR & (0x1<<4) )
    {
        LPC_TIM0->IR = 0x1<<4;      /* clear interrupt flag */
        /* En el registro LPC_TIM0->CR0 se copió el valor del contador cuando llegó el flanco
        de subida*/
        .....
    }
}
```

Comparación: ejemplo

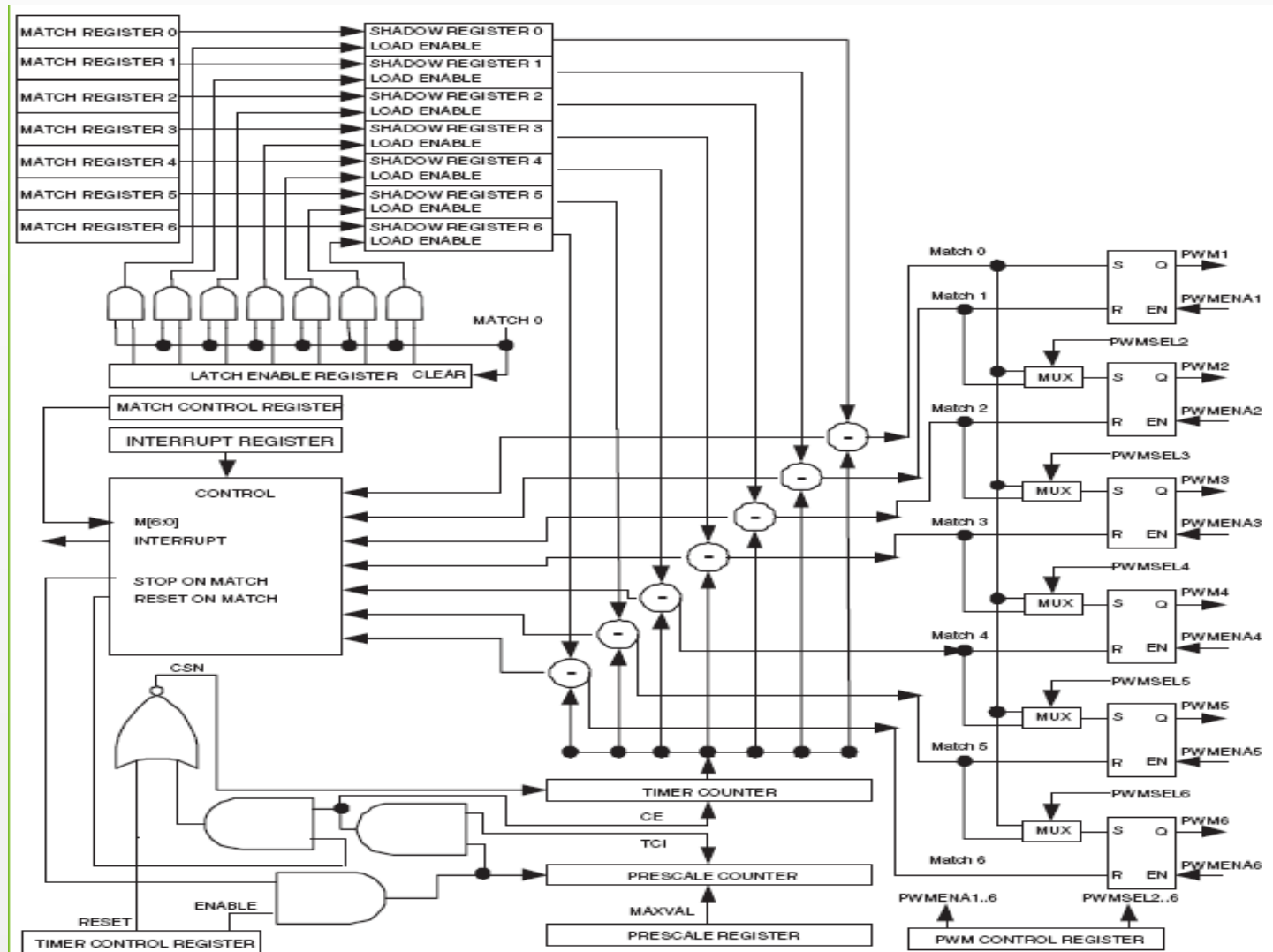
❑ Retardo de varios milisegundos:

- El preescaler a 0 (valor de reset)
- PCLK=9MHz.

```
void delayMs(uint32_t delayInMs)
{
    LPC_TIM0->TCR = 0x02;           /* reset timer */
    LPC_TIM0->PR  = 0x00;           /* set prescaler to zero */
    LPC_TIM0->MR0 = delayInMs * (9000000 / 1000 - 1);
    LPC_TIM0->IR  = 0xff;          /* reset all interrupts */
    LPC_TIM0->MCR = 0x04;          /* stop timer on match */
    LPC_TIM0->TCR = 0x01;          /* start timer */

    /* wait until delay time has elapsed */
    while (LPC_TIM0->TCR & 0x01);
}
```

PWM: esquema general



PWM: registros

Pulse Width Modulator 1 (PWM 1)

Prescaler
PR: 0x00000000
PC: 0x00000000

Timer
TCR: 0x00000000
TC: 0x00000000

☐ Counter Enable
☐ Reset
☐ PWM Enable

Interrupt Register
IR: 0x00000000

Match Channels

x	MRx	Interrupt	Reset	Stop	MRx Int	Latch	PWM
0	00000000H	0	0	0	0	0	
1	00000000H	0	0	0	0	0	0
2	00000000H	0	0	0	0	0	0
3	00000000H	0	0	0	0	0	0
4	00000000H	0	0	0	0	0	0
5	00000000H	0	0	0	0	0	0
6	00000000H	0	0	0	0	0	0

Selected Channel
MR0: 0x00000000
☐ Match 0 Latch
☐ MR0 Interrupt
☐ Interrupt on MR0
☐ Reset on MR0
☐ Stop on MR0

MCR: 0x00000000 LER: 0x00000000 PCR: 0x00000000

Capture Channels

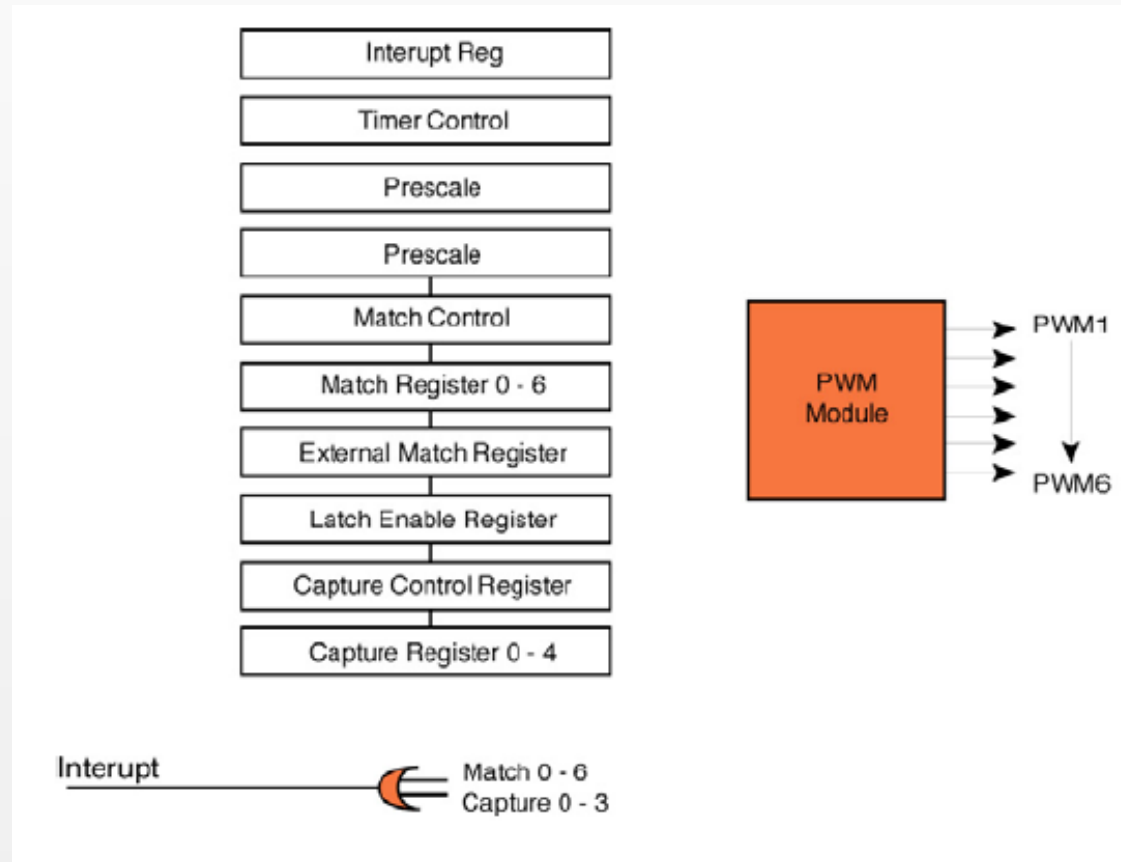
x	CRx	Rising Edge	Falling Edge	Interrupt on Event	CRx Interrupt	PCAP Pin
0	00000000H	0	0	0	0	
1	00000000H	0	0	0	0	
2	00000000H	0	0	0	0	
3	00000000H	0	0	0	0	

Selected Channel
CR0: 0x00000000
☐ Rising Edge 0
☐ Falling Edge 0
☐ Interrupt on Event 0
☐ CR0 Interrupt
☐ PCAP0 Pin
CCR: 0x00000000

Count Control
CTCR: 0x00000000 Mode: Timer Counter Input: PCAP1.0

PWM

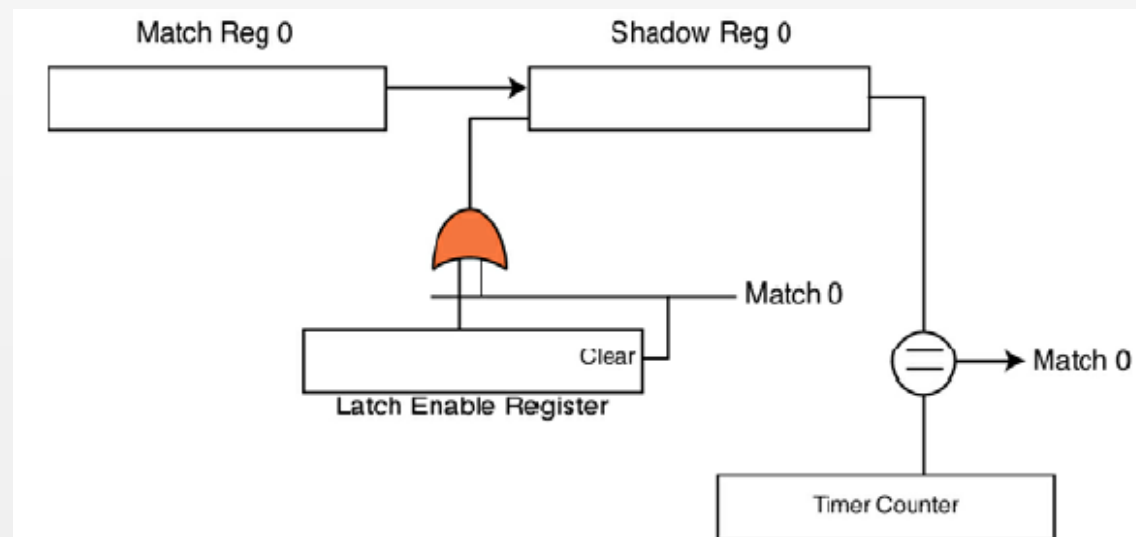
- ❑ Es un módulo timer con algunas funcionalidades añadidas.
 - 6 canales PWM.



PWM

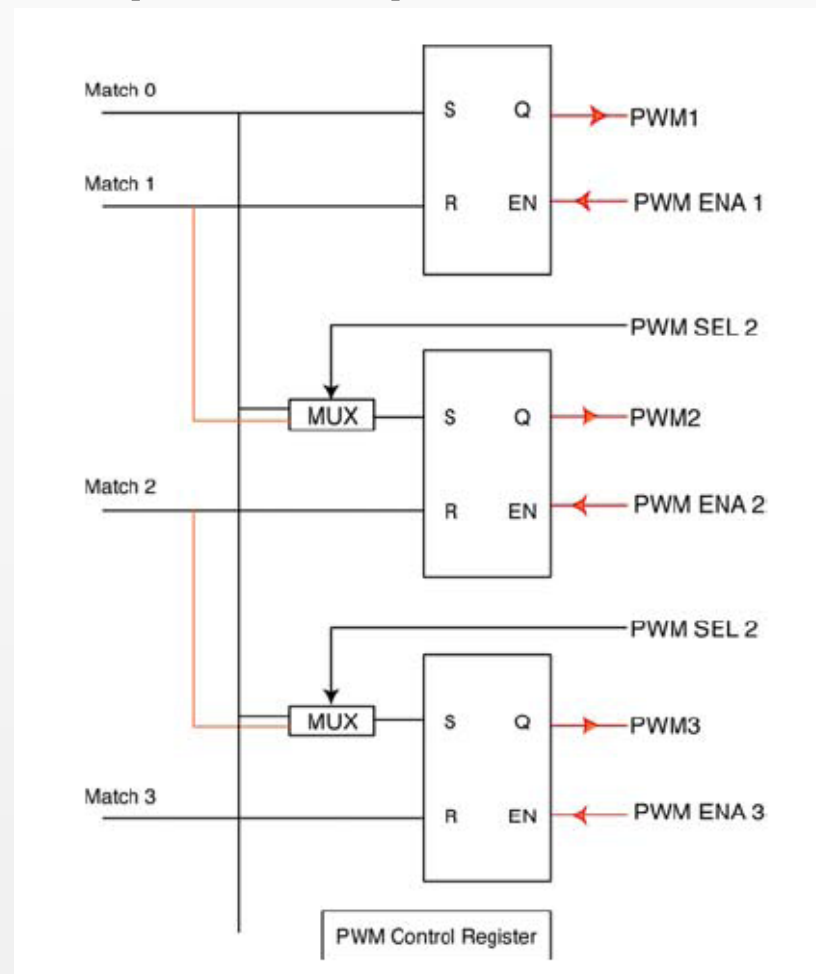
□ El cambio del valor de los registros de comparación se aplica, no cuando se modifican, sino con el comienzo de un nuevo ciclo.

- Así se evita la pérdida de algunos flancos de subida de la señal PWM.
- Ej fallo: contador varía de 0 a 100, el registro de comparación tenía un valor 30 y se desea cambiar a 10. Si el cambio se hace cuando el contador tiene un valor superior a 10, se pierde un flanco de subida.



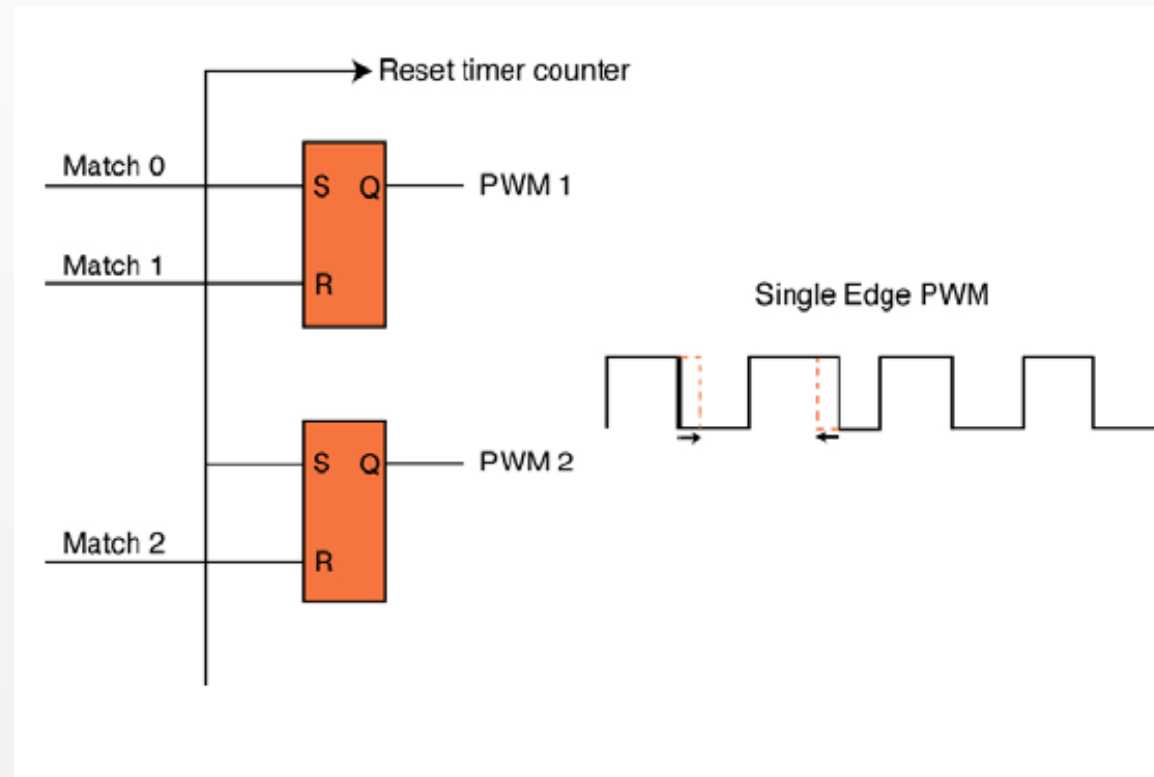
PWM

Salidas de comparación pasadas por biestables



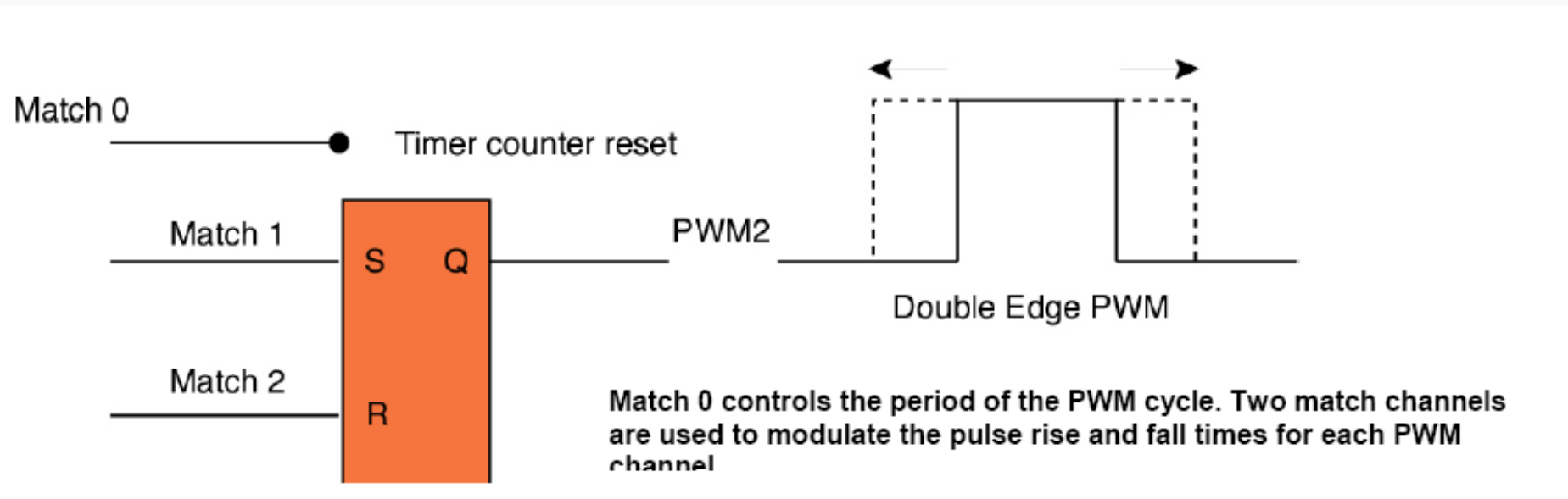
PWM: modos

□ PWM con control del flanco de bajada



PWM: modos

□ PWM con control de ambos flancos



PWM ejemplo

```
/* PWM simple flanco en el canal 1 */

void PWM_Init(uint32_t periodo )
{
    LPC_SC->PCONP |= 1<<6;          /* enable PWM1 Power
    LPC_PINCON->PINSEL4 = (1<<0); /* set GPIOs for PWM1 pin on PWM */

    LPC_PWM1->TCR = 0x00000002;      /* Counter Reset */
    LPC_PWM1->PR = 0x00;              /* count frequency: pclk */
    LPC_PWM1->MCR = (1<<1);          /* reset on PWMMR0*/

    LPC_PWM1->MR0 = periodo;          /* set PWM cycle */
    LPC_PWM1->MR0 = periodo/2;        /* valor para un ciclo de trabajo del 50%*/
    LPC_PWM1->LER = (0x1<<0) | (0x1<<1); /* PWM 0 y 1 latch enabled */

    LPC_PWM1->PCR = (0x1<<9);        /* Habilita PWM1 modo simple*/
    LPC_PWM1->TCR = (0x1<<0) | (0x1<<3); /* counter enable, PWM enable */
}

/*función para cambiar el ciclo de trabajo
void PWM_Set( uint32_t duty )
{
    LPC_PWM1->MR1 = duty;
    LPC_PWM1->LER = (1<<1);
}
```

WATCHDOG

- Periodos configurables de hasta 1145 s (15 MHz de PCLK)

$$(T_{PCLK} \times 256 \times 4) \text{ to } (T_{PCLK} \times 2^{32} \times 4)$$

- Modos de funcionamiento:

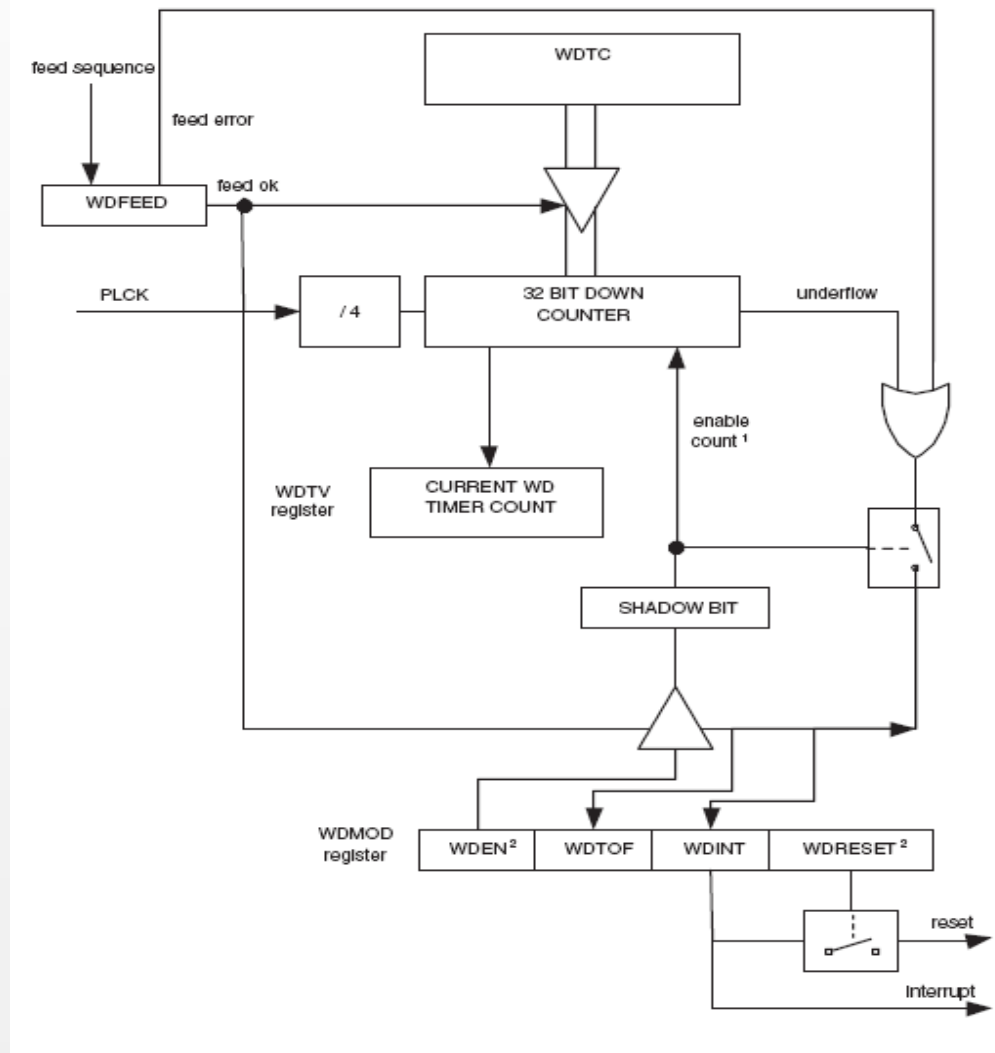
- Reset
- Interrupción.

- Se habilita por software, pero sólo se puede deshabilitar tras un reset o una interrupción por desbordamiento del watchdog.

- La duración del temporizador se fija según el valor del registro WDTC y del valor de PCLK

- Antes de que venza la temporización, el watchdog debe reiniciarse realizando una secuencia de escritura determinada en el registro WDFEED. Si no, se produce un reset o una interrupción

WATCHDOG: diagrama de bloques



Watchdog: Ejemplo

```
#define WDEN                0x00000001
#define WDRESET             0x00000002
#define WDTOF              0x00000004
#define WDINT              0x00000008

#define WDT_FEED_VALUE     0x003FFFFFF

void WDTInit( void )
{
    /* once WDEN is set, the WDT will start after feeding */
    LPC_WDT->TC = WDT_FEED_VALUE;
    LPC_WDT->MOD = WDEN;

    LPC_WDT->FEED = 0xAA;          /* Feeding sequence */
    LPC_WDT->FEED = 0x55;
}

void WDTFeed( void ) /* esta función debe llamarse antes de que venza la
temporización */
{
    LPC_WDT->FEED = 0xAA;          /* Feeding sequence */
    LPC_WDT->FEED = 0x55;
}
```

Referencias

- ❑ Manual LPC17xx
- ❑ Ejemplos de programación de Keil.