LPC1768: TIMERs

- LPC1768 has 8 x 32 bits timers:
  - General use: **Timers 0**, **1**, **2**, **3**, with (at least) 2 Capture Inputs + 4 Match Outputs.
  - 2 for **PWM** generation.
  - **RTC** and **WDT**.
  - **RIT** and **SysTick**
- **Pinned out** with a choice of multiple pins for each
- With a **programmable 32-bit Preescaler**
- Can generate IRQs
- Can generate sampling frequency of ADC.
- GPDMA controller support
  - Allows for timed memory-to-memory transfers.

TIMERs 0/1/2/3

- A minimum of **two Capture inputs** and **four Match outputs** are pinned out for all four timers, with a choice of multiple pins for each.
- A 32-bit Timer/Counter with a programmable 32-bit Prescaler Counter or Timer operation.
- Up to **two 32-bit capture channels** per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- **Four 32-bit match registers that allow:**
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to **four external outputs** corresponding to match registers, with the following capabilities:
  - Set low, high, toggle or do nothing on match.

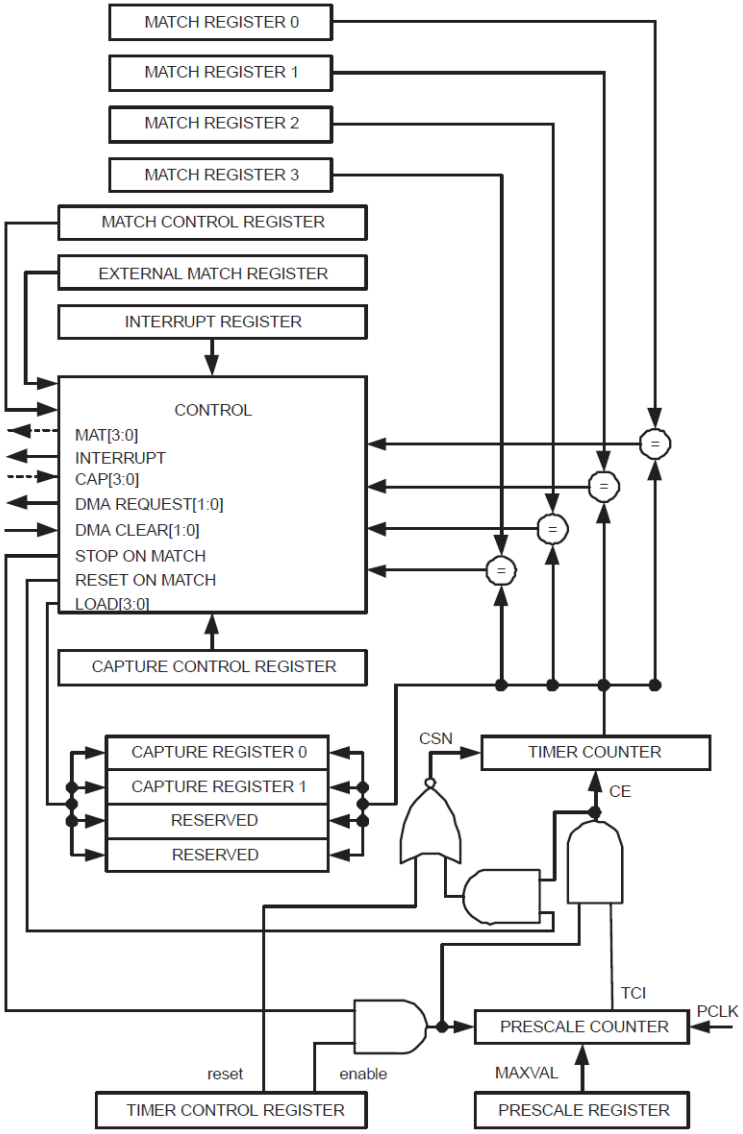
TIMERs 0/1/2/3

Input Capture (CAPn.0, CAP2.1)

- To measure external signals' timings (frequency, pulse width, etc.)
- Up to **2 32-bit capture channels per timer**, can take a snapshot of the timer value with input signal transitions
- **Capture event may also generate an interrupt**

Output Compare (MATn.0 … MATn.3)

- **To generate signals (high precision, different types such as PWM)**
- 4 32-bit match registers that allow:
  - Continuous operation
  - Stop timer on match
  - Reset timer on match
- **Match may also generate an interrupt**
- Up to 4 external outputs corresponding to match registers, with the following capabilities:
  - Set low on match.
  - Set high on match.
  - Toggle on match.
  - Do nothing on match.



2.2. TIMERs: Registers (I)

Table 425. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	T0IR - 0x4000 4000 T1IR - 0x4000 8000 T2IR - 0x4009 0000 T3IR - 0x4009 4000
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	T0TCR - 0x4000 4004 T1TCR - 0x4000 8004 T2TCR - 0x4009 0004 T3TCR - 0x4009 4004
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	T0TC - 0x4000 4008 T1TC - 0x4000 8008 T2TC - 0x4009 0008 T3TC - 0x4009 4008
PR	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	T0PR - 0x4000 400C T1PR - 0x4000 800C T2PR - 0x4009 000C T3PR - 0x4009 400C
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	R/W	0	T0PC - 0x4000 4010 T1PC - 0x4000 8010 T2PC - 0x4009 0010 T3PC - 0x4009 4010
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	T0MCR - 0x4000 4014 T1MCR - 0x4000 8014 T2MCR - 0x4009 0014 T3MCR - 0x4009 4014

2.2. TIMERs: Registers (II)

Table 425. TIMER/COUNTER0-3 register map

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	T0MR0 - 0x4000 4018 T1MR0 - 0x4000 8018 T2MR0 - 0x4009 0018 T3MR0 - 0x4009 4018
MR1	Match Register 1. See MR0 description.	R/W	0	T0MR1 - 0x4000 401C T1MR1 - 0x4000 801C T2MR1 - 0x4009 001C T3MR1 - 0x4009 401C
MR2	Match Register 2. See MR0 description.	R/W	0	T0MR2 - 0x4000 4020 T1MR2 - 0x4000 8020 T2MR2 - 0x4009 0020 T3MR2 - 0x4009 4020
MR3	Match Register 3. See MR0 description.	R/W	0	T0MR3 - 0x4000 4024 T1MR3 - 0x4000 8024 T2MR3 - 0x4009 0024 T3MR3 - 0x4009 4024
CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0	T0CCR - 0x4000 4028 T1CCR - 0x4000 8028 T2CCR - 0x4009 0028 T3CCR - 0x4009 4028

2.2. TIMERs: Registers (III)

Table 425. TIMER/COUNTER0-3 register map *...continued*

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0(CAP0.0 or CAP1.0 respectively) input.	RO	0	T0CR0 - 0x4000 402C T1CR0 - 0x4000 802C T2CR0 - 0x4009 002C T3CR0 - 0x4009 402C
CR1	Capture Register 1. See CR0 description.	RO	0	T0CR1 - 0x4000 4030 T1CR1 - 0x4000 8030 T2CR1 - 0x4009 0030 T3CR1 - 0x4009 4030
EMR	External Match Register. The EMR controls the external match pins MATn.0-3 (MAT0.0-3 and MAT1.0-3 respectively).	R/W	0	T0EMR - 0x4000 403C T1EMR - 0x4000 803C T2EMR - 0x4009 003C T3EMR - 0x4009 403C
CTCR	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	R/W	0	T0CTCR - 0x4000 4070 T1CTCR - 0x4000 8070 T2CTCR - 0x4009 0070 T3CTCR - 0x4009 4070

2.2. TIMERs: Registers (IR, TCR)

Table 426. Interrupt Register (T[0/1/2/3]IR - addresses 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000) bit description

Bit	Symbol	Description	Reset Value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
31:6	-	Reserved	-

Table 427. Timer Control Register (TCR, TIMERn: TnTCR - addresses 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004) bit description

Bit	Symbol	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

2.2. TIMERs: Registers (CCR)

Table 428. Count Control Register (T[0/1/2/3]CTCR - addresses 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Counter/ Timer Mode		This field selects which rising PCLK edges can increment the Timer's Prescale Counter (PC), or clear the PC and increment the Timer Counter (TC).	00
		00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. The Prescale Counter is incremented on every rising PCLK edge.	
		01	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		10	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		11	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.	
3:2	Count Input Select		When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking.	00
		00	CAPn.0 for TIMERn	
		01	CAPn.1 for TIMERn	
		10	Reserved	
		11	Reserved	
			Note: If Counter mode is selected for a particular CAPn input in the TnCTCR, the 3 bits for that input in the Capture Control Register (TnCCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

2.2. TIMERs: Registers (MCR)

MATn.0

MATn.1

MATn.2

MATn.3

Table 429. Match Control Register (T[0/1/2/3]MCR - addresses 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014) bit description

Bit	Symbol	Value	Description	Reset Value
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		0	This interrupt is disabled	
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.	0
		0	Feature disabled.	
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		0	Feature disabled.	
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		0	This interrupt is disabled	
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.	0
		0	Feature disabled.	
5	MR1S	1	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		0	Feature disabled.	
6	MR2I	1	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		0	This interrupt is disabled	
7	MR2R	1	Reset on MR2: the TC will be reset if MR2 matches it.	0
		0	Feature disabled.	
8	MR2S	1	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		0	Feature disabled.	
9	MR3I	1	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		0	This interrupt is disabled	
10	MR3R	1	Reset on MR3: the TC will be reset if MR3 matches it.	0
		0	Feature disabled.	
11	MR3S	1	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		0	Feature disabled.	

2.2. TIMERs: Registers (EMR)

Table 431. External Match Register (T[0/1/2/3]EMR - addresses 0x4000 403C, 0x4000 803C, 0x4009 003C, 0x4009 403C) bit description

Bit	Symbol	Description	Reset Value
0	EM0	External Match 0. When a match occurs between the TC and MR0, this bit can either toggle, go low, go high, or do nothing, depending on bits 5:4 of this register. This bit can be driven onto a MATn.0 pin, in a positive-logic manner (0 = low, 1 = high).	0
1	EM1	External Match 1. When a match occurs between the TC and MR1, this bit can either toggle, go low, go high, or do nothing, depending on bits 7:6 of this register. This bit can be driven onto a MATn.1 pin, in a positive-logic manner (0 = low, 1 = high).	0
2	EM2	External Match 2. When a match occurs between the TC and MR2, this bit can either toggle, go low, go high, or do nothing, depending on bits 9:8 of this register. This bit can be driven onto a MATn.2 pin, in a positive-logic manner (0 = low, 1 = high).	0
3	EM3	External Match 3. When a match occurs between the TC and MR3, this bit can either toggle, go low, go high, or do nothing, depending on bits 11:10 of this register. This bit can be driven onto a MATn.3 pin, in a positive-logic manner (0 = low, 1 = high).	0
5:4	EMC0	External Match Control 0. Determines the functionality of External Match 0. Table 432 shows the encoding of these bits.	00
7:6	EMC1	External Match Control 1. Determines the functionality of External Match 1. Table 432 shows the encoding of these bits.	00
9:8	EMC2	External Match Control 2. Determines the functionality of External Match 2. Table 432 shows the encoding of these bits.	00
11:10	EMC3	External Match Control 3. Determines the functionality of External Match 3. Table 432 shows the encoding of these bits.	00
15:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

2.2. TIMERs: Registers (CCR)

Table 430. Capture Control Register (T[0/1/2/3]CCR - addresses 0x4000 4028, 0x4000 8020, 0x4009 0028, 0x4009 4028) bit description

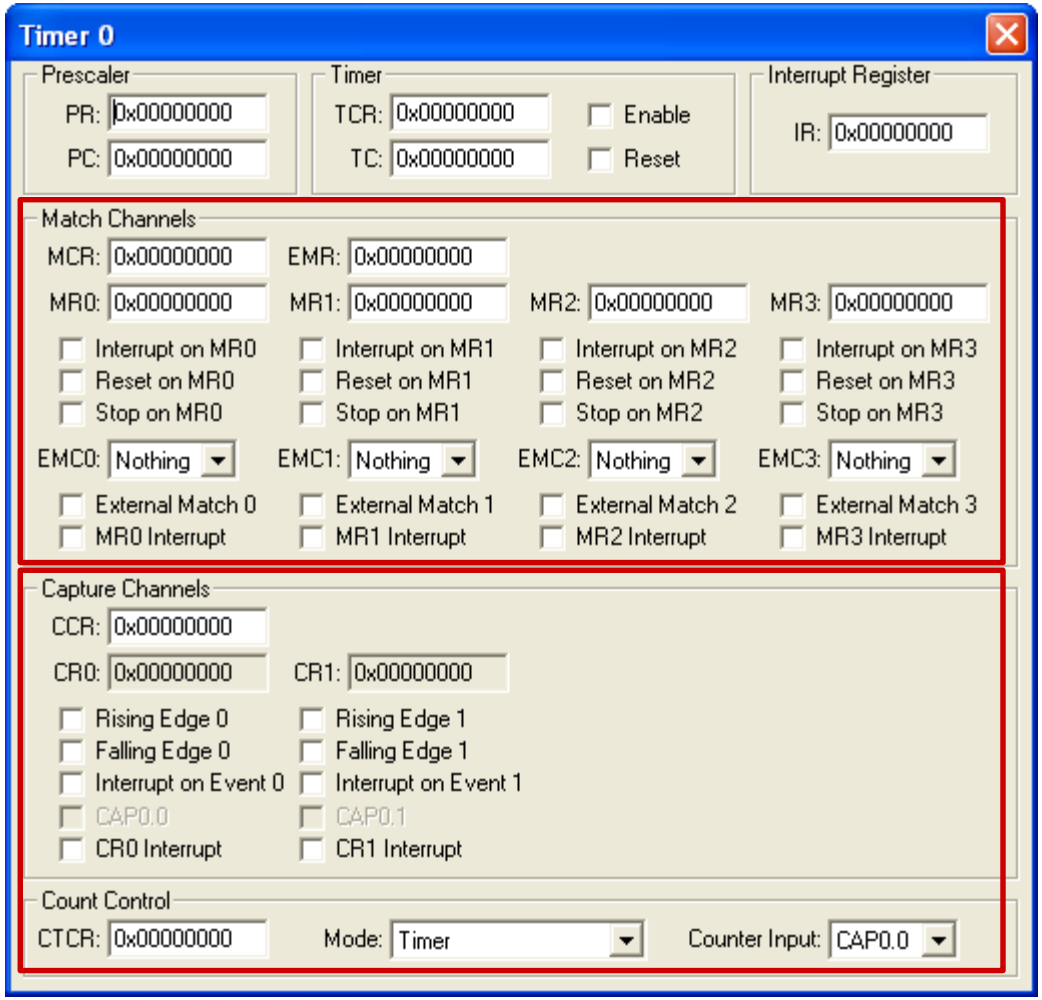
Bit	Symbol	Value	Description	Reset Value
0	CAP0RE	1	Capture on CAPn.0 rising edge: a sequence of 0 then 1 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
1	CAP0FE	1	Capture on CAPn.0 falling edge: a sequence of 1 then 0 on CAPn.0 will cause CR0 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
2	CAP0I	1	Interrupt on CAPn.0 event: a CR0 load due to a CAPn.0 event will generate an interrupt.	0
		0	This feature is disabled.	
3	CAP1RE	1	Capture on CAPn.1 rising edge: a sequence of 0 then 1 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
4	CAP1FE	1	Capture on CAPn.1 falling edge: a sequence of 1 then 0 on CAPn.1 will cause CR1 to be loaded with the contents of TC.	0
		0	This feature is disabled.	
5	CAP1I	1	Interrupt on CAPn.1 event: a CR1 load due to a CAPn.1 event will generate an interrupt.	0
		0	This feature is disabled.	
31:6	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

CAPn.0

CAPn.1

TIMERs: Keil ARM_MDK Debug Windows

- **Pheriferals -> Timers -> Timer 0-1-2-3**



LPC17xx: Timer0 (code example)

- Go to page 490 in the manual and follow the links provided there
- 1. Powering the Timer0 (Table 46, page 63)

```
LPC_SC->PCONP |= 1 << 1; //Power up Timer 0
```

- 2. After enabling it you need to give the clock for Timer0 (Table 40, page 56. For Timer2/3 refer Table 41, page 57) Select clock as per your requirements.

```
LPC_SC->PCLKSEL0 |= 1 << 3; // Clock for timer = CCLK/2
```

- 3. Configuring the interrupt LPC1768 contains 4 match registers for each timer. These match registers can be used to reset the timer, generate interrupt, stop the timer, to generate timing signal on an external pin. Now we shall use Timer0 Match0 register to generate the interrupt.

```
LPC_TIM0->MR0 = Ftick/Fint; // LED blinking frequency based on the clock frequency timer
```

LPC17xx: Timer0 (code example)

- 4. Configure it as to interrupt when the timer count matches the Match0 (Table 429, page 496)

  LPC_TIM0->MCR |= 1 << 0; // Interrupt on Match0 compare

- 5. Reset the timer (Table 427, page 494)

  LPC_TIM0->TCR |= 1 << 1; // Reset Timer0

- 6. Enable Timer interrupts

  NVIC_EnableIRQ(TIMER0_IRQn); // Enable timer interrupt

- 5. And finally start the timer (Table 427, page 494)

  LPC_TIM0->TCR |= 1 << 0; // Start timer

LPC17xx: Timer0 (code example)

Interrupt function:

```
void TIMER0_IRQHandler (void)
{
    if((LPC_TIM0->IR & 0x01) == 0x01) //  MR0 interrupt ?
      {
       LPC_TIM0->IR |= 1 << 0; // Clear MR0 interrupt flag
       LPC_GPIO1->FIOPIN ^= 1 << 29; // Toggle the LED
      }

     if((LPC_TIM0->IR & 0x02) == 0x02) //  MR1 interrupt ?

    .....

    .....
}
```

Timers: Frequency Measurement (I)

- Caudal meter:

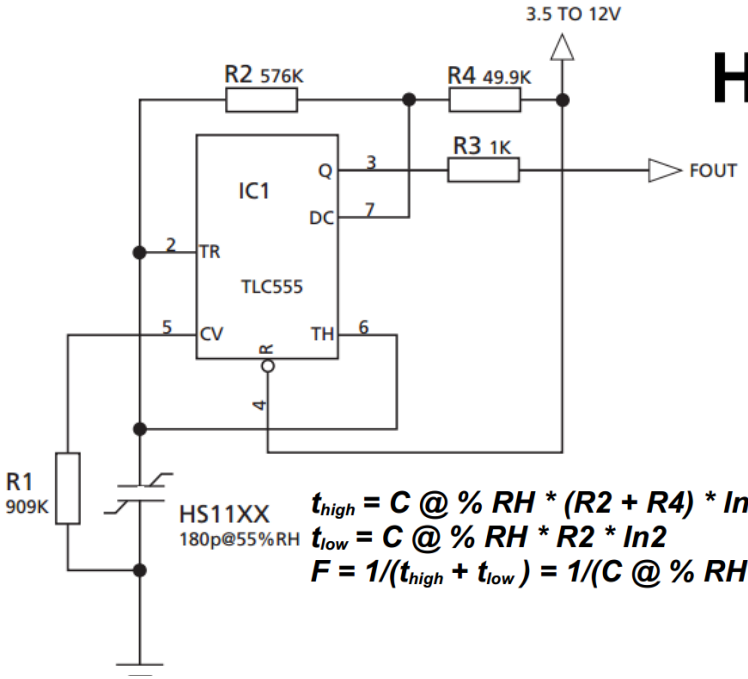
- Working Voltage: 5 to 18VDC
- Max current draw: 15mA @ 5V
- Working Flow Rate: 1 to 30 Liters/Minute
- Working Temperature range: -25 to 80°C
- Working Humidity Range: 35%-80% RH
- Maximum water pressure: 2.0 MPa
- Output duty cycle: 50% +-10%
- Output rise time: 0.04us
- Output fall time: 0.18us
- Flow rate pulse characteristics: Frequency (Hz) = 7.5 * Flow rate (L/min)
- Pulses per Liter: 450
- Durability: minimum 300,000 cycles

- Interrupt function:

```
void TIMER0_IRQHandler (void) // Timer 0 y capture mode (CAP0.0)
{
      static uint32_t  temp, float frequency;
      LPC_TIM0->IR |= 1 << 4;                    // Clear CR0 flag interrupt
      frequency=Fpclk/(LPC_TIM0->CR0-temp);      // frequency in Hz.
      caudal=frecuency/7.5;                      // caudal (L/min).
      temp=LPC_TIM0->CR0;                         // update temp

}
```

Timers: Frequency Measurement (II)



HS1101 Relative Humidity Sensor

$t_{high} = C @ \%\ RH * (R2 + R4) * ln2$

$t_{low} = C @ \%\ RH * R2 * ln2$

$F = 1/(t_{high} + t_{low}) = 1/(C @ \%\ RH * (R4 + 2 * R2) * ln2)$

Typical Characteristics for Frequency Output Circuits

REFERENCE POINT AT 6660Hz FOR 55%RH / 25°C

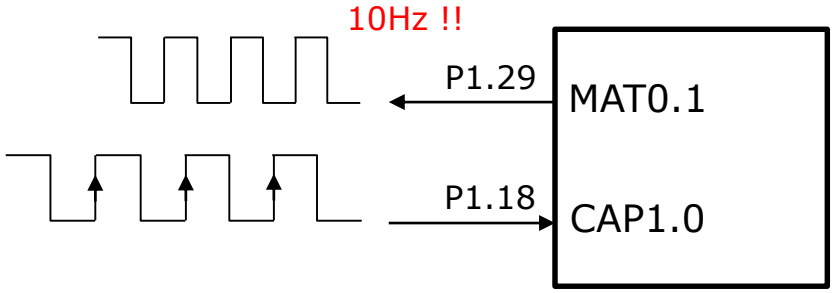
RH	0	10	20	30	40	50	60	70	80	90	100
Frequency	7351	7224	7100	6976	6853	6728	6600	6468	6330	6186	6033

Typical for a 555 Cmos type. TLC555 (RH : Relative Humidity in %, F : Frequency in Hz)

Polynomial response :

$$F_{mes(Hz)} = F55_{(Hz)}(1.1038 - 1.936810^{-3} * RH + 3.011410^{-6} * RH^2 - 3.440310^{-8} * RH^3)$$

Timers: Match and Capture: code example




```
int main(void)
{
uint32_t i;
/*
    P1.29:  MAT0.1 ,    P1.18:  CAP1.0
*/

    LPC_SC->PCONP|=(1<<1)|(1<<2);     // Power ON: Timer 0 and Timer 1 and Timer 2
    LPC_PINCON->PINSEL3|= 0x0C000000; //?
    LPC_PINCON->PINSEL3|= 0x00000030; //?

/*****************************************************************************
                            Timer0 (32bit)
*****************************************************************************
  Timer 0 en modo Output Compare (reset T0TC on Match 1)
  Timer clk: 25 MHz
  MAT0.1 : On match Toggle pin/output (P1.29) --> LED D8 blueboard
*/
    LPC_TIM0->PR = 0x0000;   //?
    LPC_TIM0->MCR = 0x0010;  //?
    LPC_TIM0->MR1 = F_pclk/F_parpadeo-1; //?
    LPC_TIM0->EMR = 0x00C2;    //?
    LPC_TIM0->TCR = 0x01;     //?

/*****************************************************************
                            Timer1 (32bit)
*****************************************************************
   Timer 1 en modo Captura (CAP1.0)
   Timer clk: 25 MHz (defecto reset)
   CCR1.0 : Captura en flanco de bajada, y genera interrupción
*/

    LPC_TIM1->PR = 0x0000;   //?
    LPC_TIM1->MCR = 0x0000;  //?
    LPC_TIM1->CCR = 0x0006;  //?
    LPC_TIM1->EMR = 0x0000;  //?
    LPC_TIM1->TCR = 0x01;    //?


   NVIC_SetPriority(TIMER1_IRQn,1);   //?
   NVIC_EnableIRQ(TIMER1_IRQn);       //?
   NVIC_SetPriorityGrouping(2);       //?

   while(1);
```

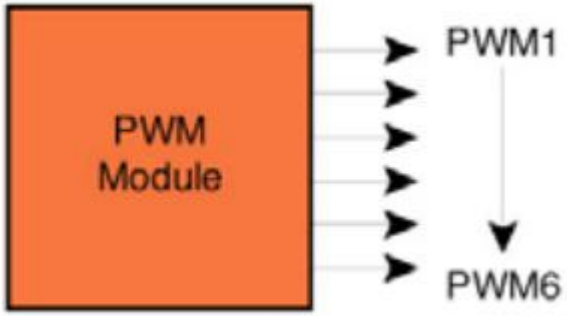
```
#include <LPC17xx.H>
#define F_cpu 100e6      // Defecto Keil (xtal=12Mhz)
#define F_pclk F_cpu/4   // Defecto despues del reset
#define F_parpadeo 20    // Cada 50ms cambia de estado el pin P1.29
float frecuencia;
uint32_t  N;             // variable global para trazar el Nº de cuentas entre flancos

// ISR Timer 1
  void TIMER1_IRQHandler(void) {
  static uint32_t  temp;
  LPC_TIM1->IR|= (1<<4);       // borra el flag
  N=LPC_TIM1->CR0-temp;        // Nº de cuentas de T1TC entre dos flancos consecutivos
  frecuencia=F_pclk/N;         // frecuencia en Hz.
  temp=LPC_TIM1->CR0;          // almacenamos la captura
}
```

Pulse Width Modulator (PWM)

- A 32-bit Timer/Counter with a programmable 32-bit prescaler. (may use the peripheral clock or one of the capture inputs as the clock source).

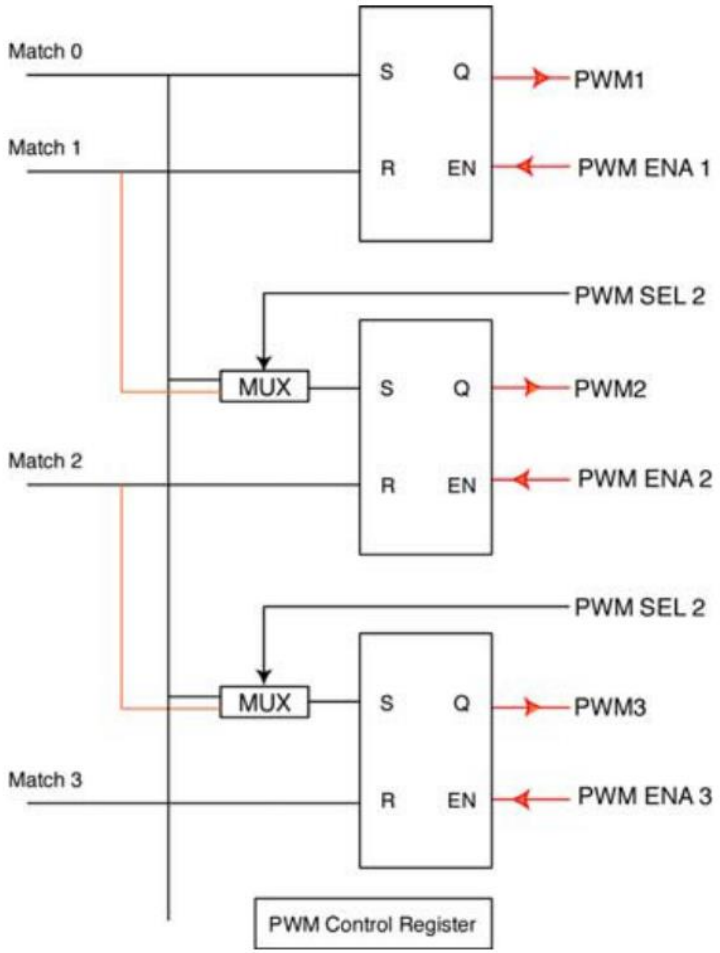
- Seven match registers allow up to **6 single edge** controlled or **3 double edge** controlled PWM outputs, or a mix of both types.



- The match registers also allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.

Pulse Width Modulator (PWM)

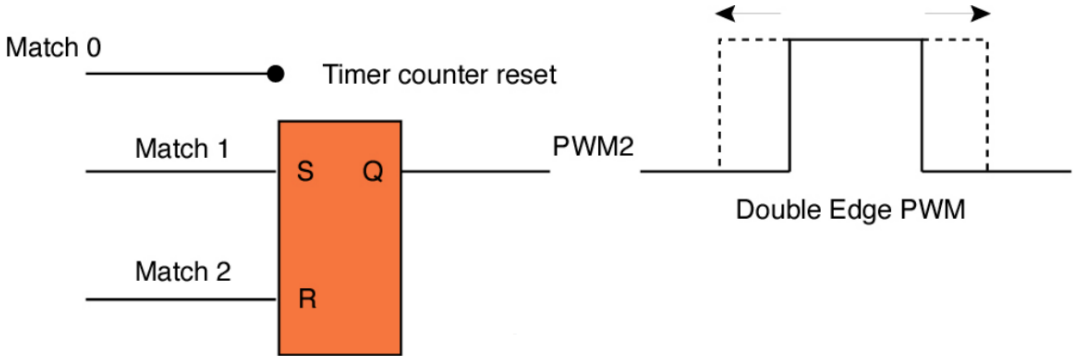
- Supports **single edge** controlled and/or **double edge** controlled PWM outputs.
  - **Single edge** controlled PWM outputs all go high at the beginning of each cycle unless the output is a constant low.
  - **Double edge** controlled PWM outputs can have either edge occur at any position within a cycle. This allows for both positive going and negative going pulses.



Pulse Width Modulator (PWM)

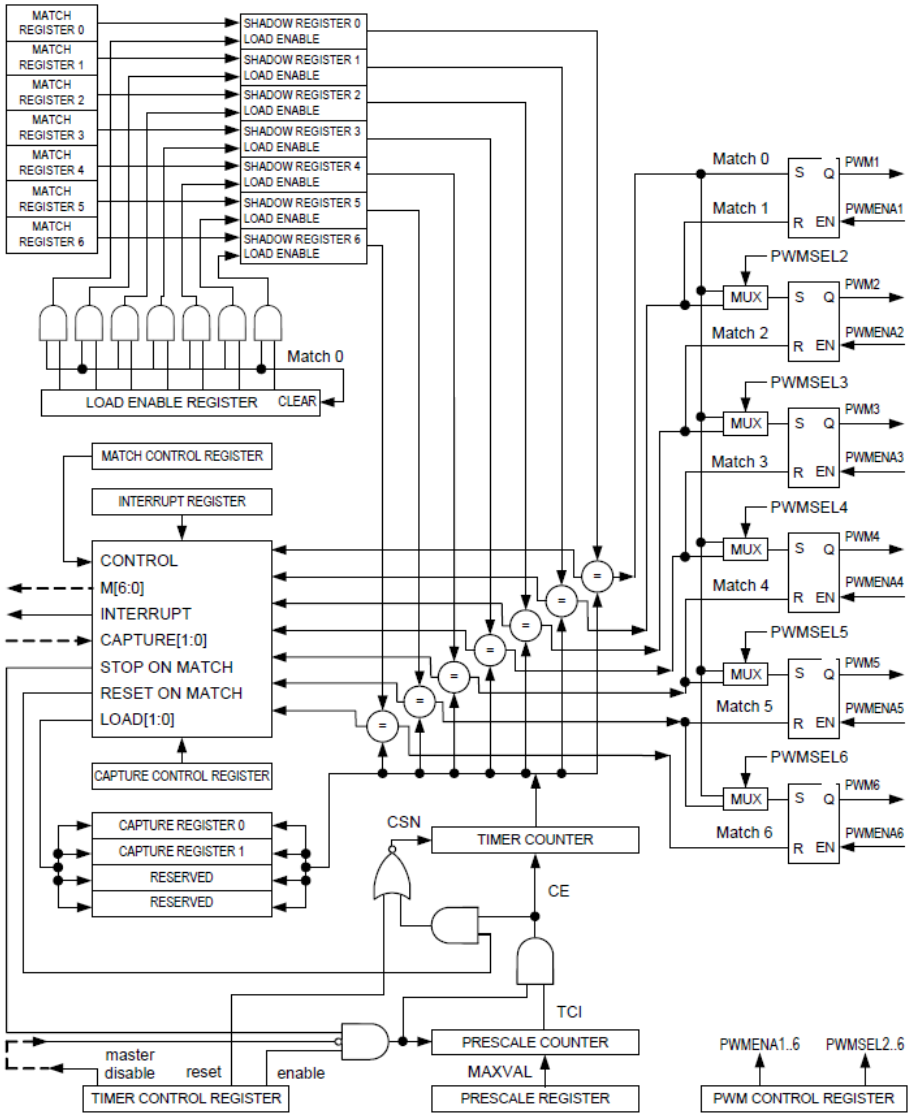
- Pulse period and width can be any number of timer counts.
  - This allows complete flexibility in the trade-off between resolution and repetition rate.
  - All PWM outputs will occur at the same repetition rate.

- **Double edge** controlled PWM outputs can be programmed to be either **positive or negative going pulses**.



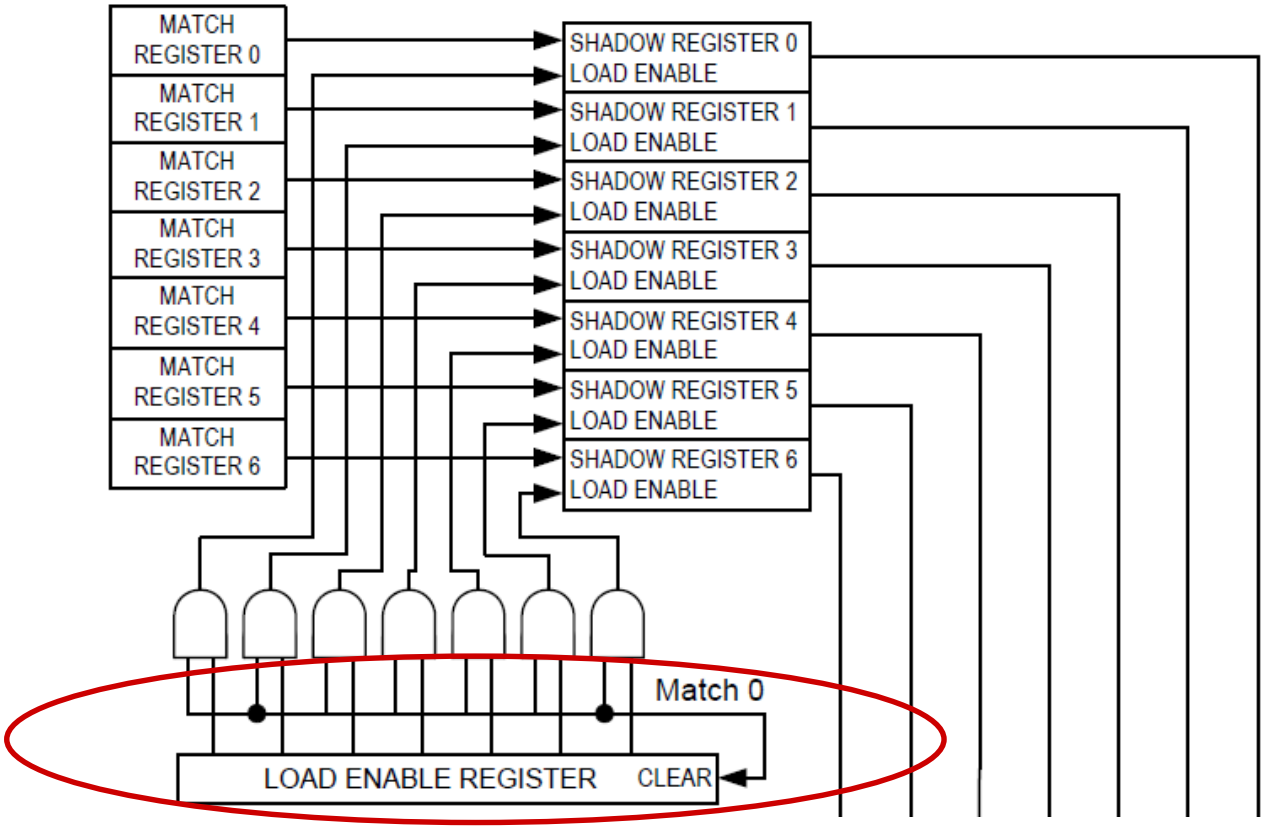
- Shadow latch mechanism.
  - Glitch-less operation.
  - Match register updates are synchronized with pulse outputs to prevent generation of erroneous pulses.
- May be used as a standard timer if the PWM mode is not enabled.

Pulse Width Modulator (PWM)



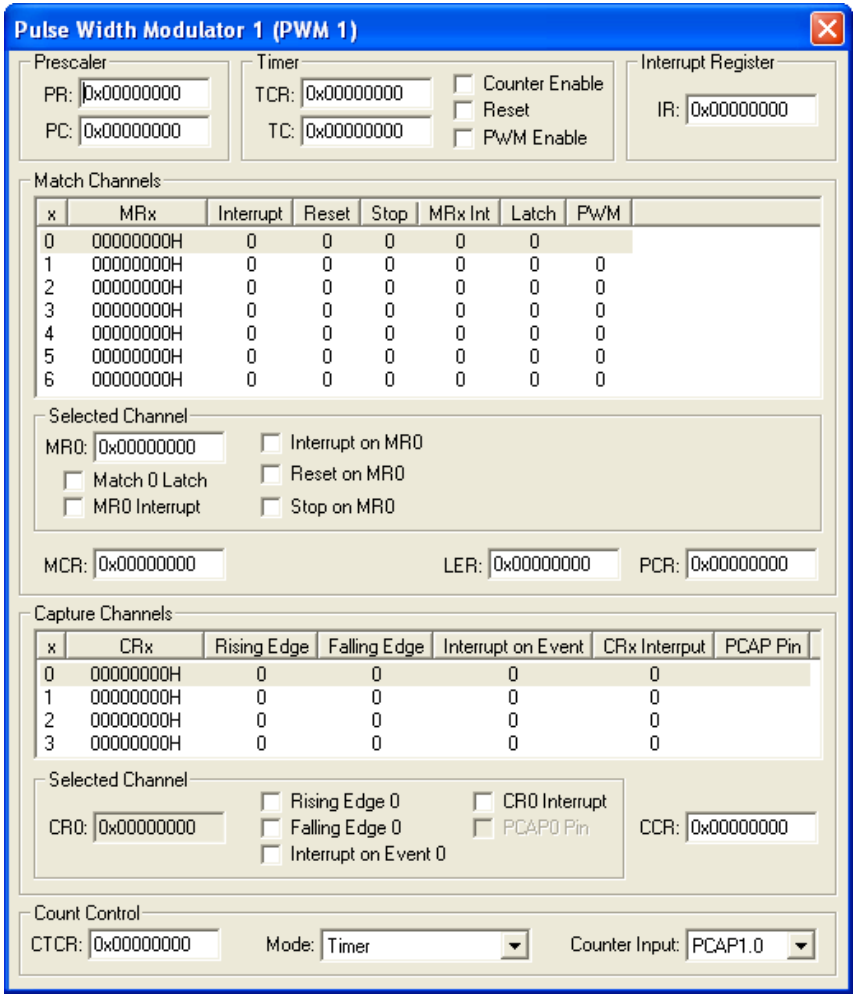
Pulse Width Modulator (PWM)

- Shadow latch mechanism:
  - When modifying a match register (**PWM1MR**n) is necessary set to "1" in PMW1LER bit corresponding.



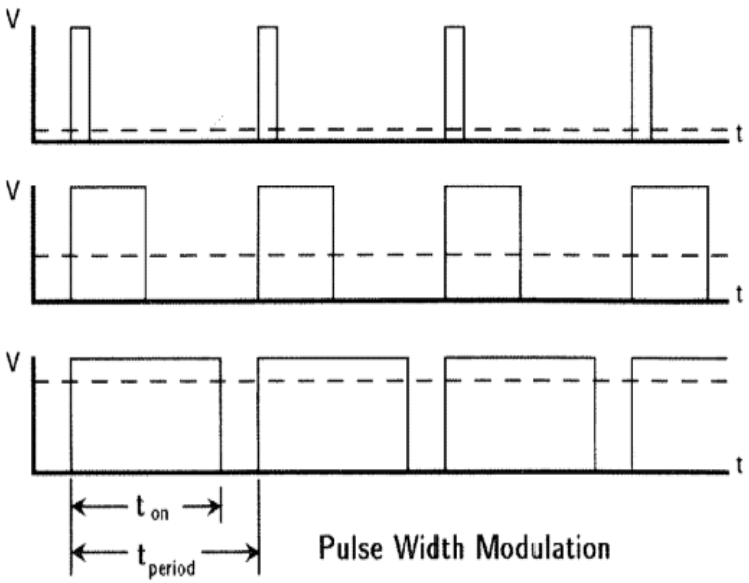
PWM: Keil ARM-MDK Debug Windows

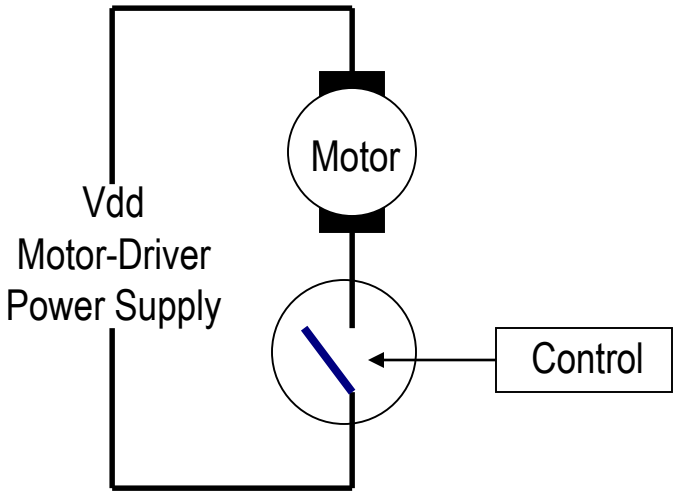
- **Peripherals-> Pulse Width Modulator 1 (PWM 1)**

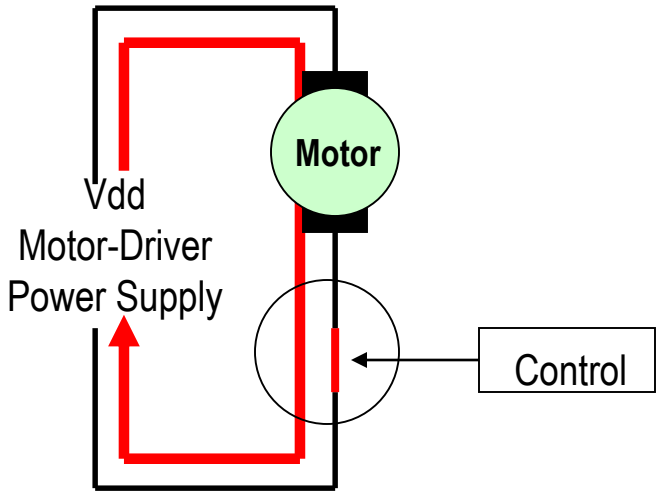


PWM: Drive DC-motors

PWM to drive DC-motors







PWM: Motor speed control

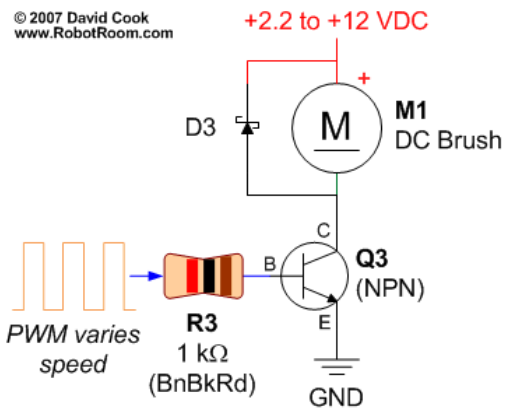
Peristaltic Liquid Pump with Silicone Tubing

- Uses approx 3/16" (4.8mm) outer diameter silicone tubing, the pump tube size has changed on us, so please measure the tubing that comes with your pump to verify!
- Working Temperature: 0°C - 40 °C
- Motor voltage: 12VDC
- Motor current: 200-300mA
- Flow rate: up to 100 mL/min
- Weight: 200 grams
- Dimensions: 27mm diameter motor, 72mm total length
- Mounting holes: 2.7mm diameter, 50mm center-to-center distance

PWM1 configuration:

```
LPC_PINCON->PINSEL4 |= 1<<0;          // P2.0 works as PWM1.1 output (table 83)
LPC_SC->PCONP |= 1<<6;                // power on
LPC_PWM1->MR0= Fpclk/Fpwm–1;          // set frequency of PWM1
LPC_PWM1->PCR|=1<<9;                  // PWMENA1=1
LPC_PWM1->MCR|=1<<1;                  // reset timer on Match0
LPC_PWM1->TCR |= (1<<0) | (1<<3);     // start timer, timer enable
```

```
void set_duty_cycle_pwm1 (float cycle)
{
      LPC_PWM1->MR1= LPC_PWM1->MR0*cycle/100;     // duty cycle
      LPC_PWM1->LER|= (1 << 1)|(1<<0);            // PWMLER[0-1]=1
}
```



PWM: Motor speed/direction control

H-Bridge circuit:

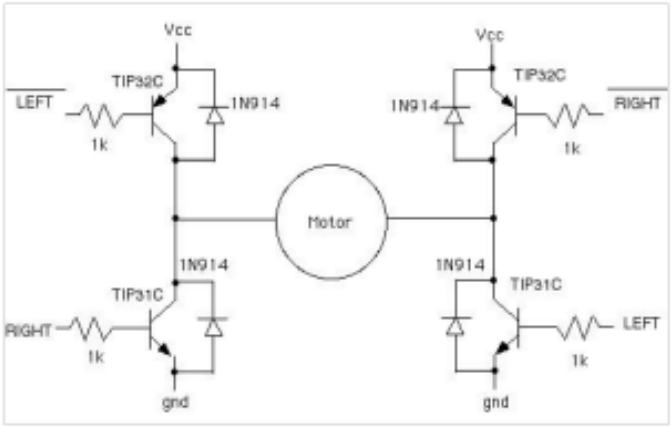
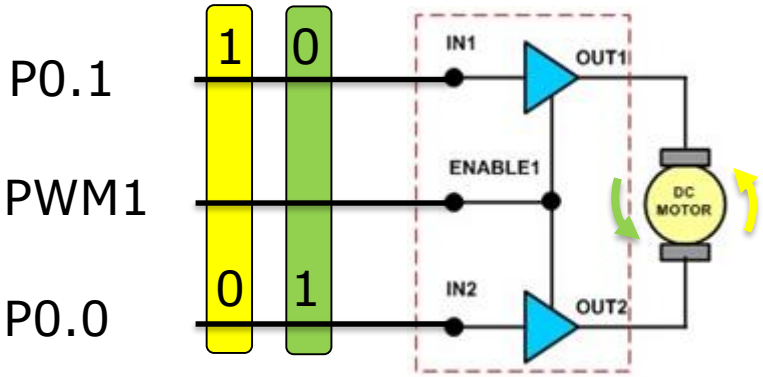
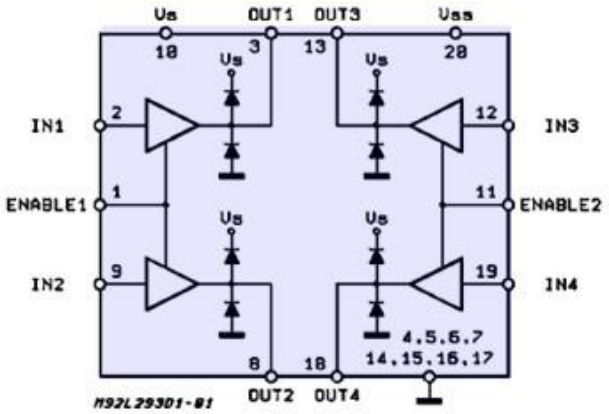
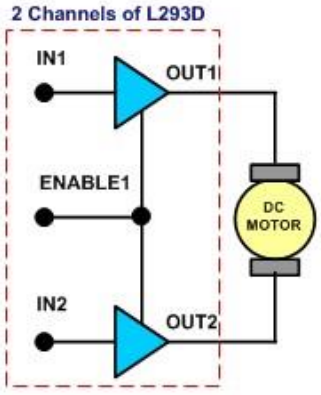


Figure 1: Typical H-bridge circuit



http://www.ermicro.com/blog





TRUTH TABLE

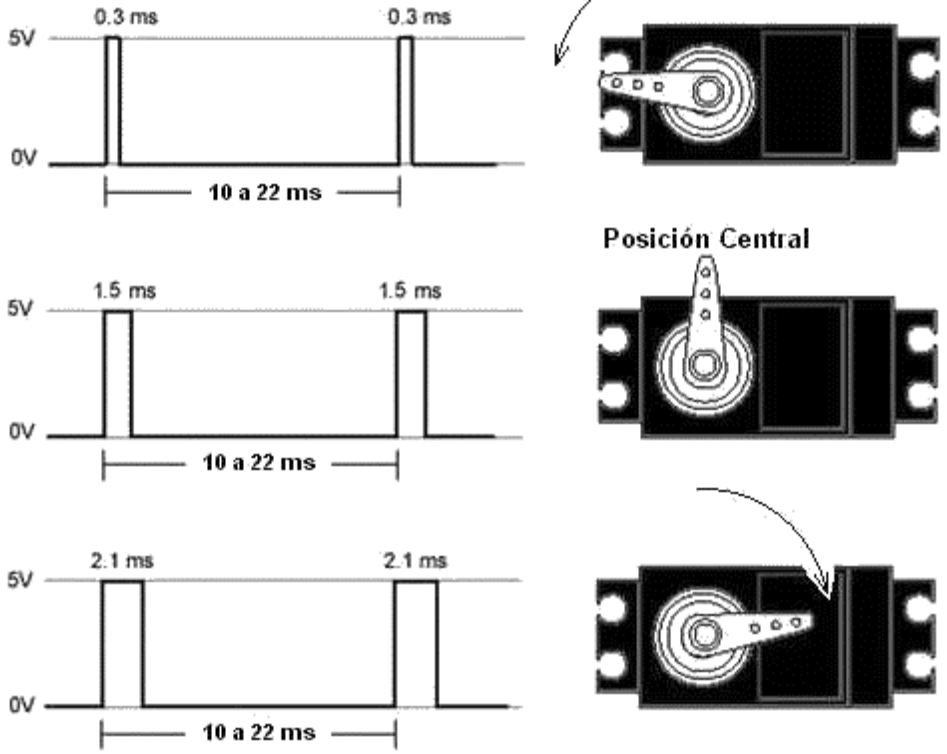
ENABLE1	IN1	IN2	OUT1	OUT2	MOTOR
H	L	L	L	L	Stop
H	H	L	H	L	Forward
H	L	H	L	H	Reverse
H	H	H	H	H	Break
L	X	X	Z	Z	Stop

H - High L - Low X - High/Low Z - High Impedance

The L293D Four Channels Motor Driver Truth Table

PWM: Servomotor control

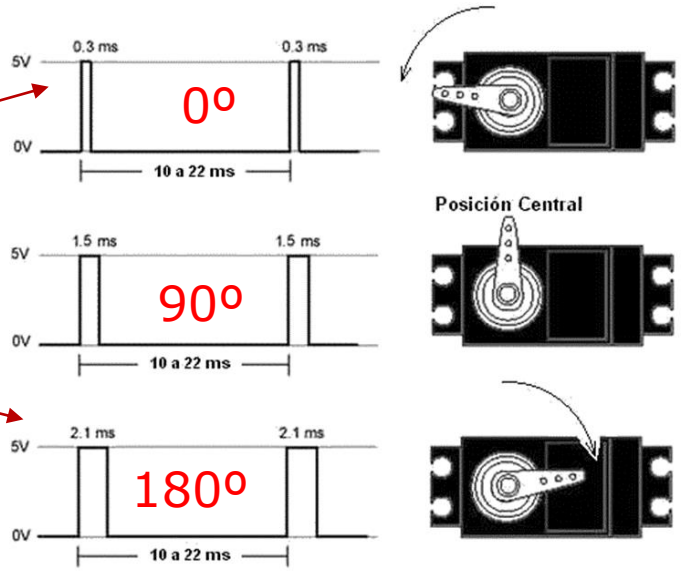
- Servomotor
  - Low power motor position-controlled.
  - Setpoint fixed with T_{on} in a fixed T signal ("PWM" type)



PWM: Servomotor code example

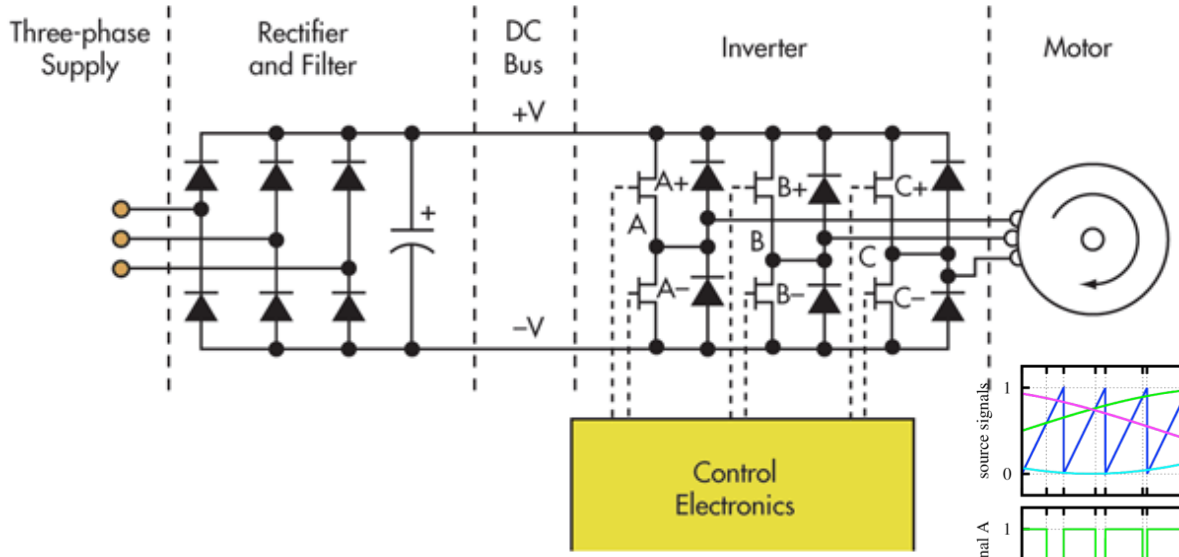
- Control de un servo para implementar un barrido (0-180º)

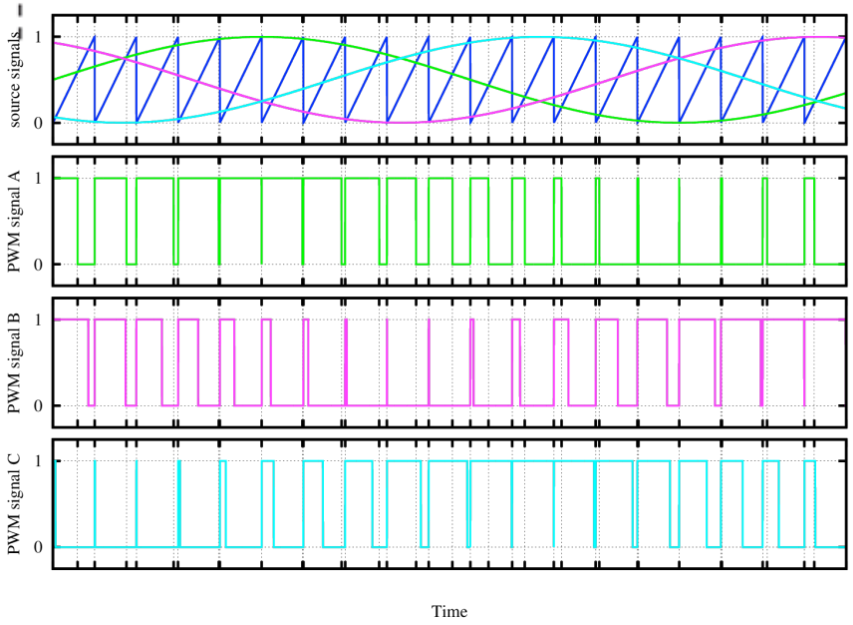
```
#include <LPC17xx.H>
#define Fpclk 25e6  // Fcpu/4 (defecto después del reset)
#define Tpwm 15e-3  // Perido de la señal PWM (15ms)
void config_pwm2(void)
  {
    LPC_PINCON->PINSEL3|=(2<<8); // P1.20 salida PWM (PWM1.2)
    LPC_SC->PCONP|=(1<<6);
    LPC_PWM1->MR0=Fpclk*Tpwm-1;
    LPC_PWM1->PCR|=(1<<10); //configurado el ENA2 (1.2)
    LPC_PWM1->MCR|=(1<<1);
    LPC_PWM1->TCR|=(1<<0)|(1<<3);
  }
void set_servo(float grados)
  {
    LPC_PWM1->MR2=(Fpclk*0.3e-3 + Fpclk*1.8e-3*grados/180);
    LPC_PWM1->LER|=(1<<2)|(1<<0);
}
int main(void)
{
  unsigned int t;
  char i;float grados1;
  config_pwm2();
  for(i=0;i<18;i++)  {
    for(t=0;t<10000000;t++); // variar el límite en simulación para ver más rápido la variación
    set_servo(grados1+=10);  // incrementamos la posición del servo de 10 en 10°
  }
  while(1);
}
```



PWM: Three-phase motors control

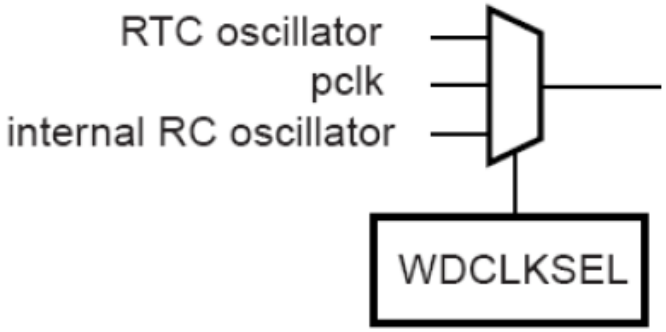
- PWM to drive three-phase motors (frequency controlled inverters)





WDT: Watchdog Timer (I)

- Watchdog is a OS tool: **CPU grabbing management.**
- The Watchdog consists of fixed prescaler ($\mathbf{T}_{WDCLK} \times 4$) and a 32-bit time-out counter.
- Clocked from the RTC, IRC oscillator, or the peripheral clock.



- The timer decrements when clocked, and when arrives to 0, it may:
  - Reset the uC.
  - Generate an IRQ.
- Cannot be disabled by software, just in the Reset or IRQ ISRs.
- The maximum + minimum Watchdog intervals are:
  - The minimum Watchdog interval is ($\mathbf{T}_{WDCLK} \times 256 \times 4$)
  - The maximum Watchdog interval is ($\mathbf{T}_{WDCLK} \times 2^{32} \times 4$) in multiples of ($\mathbf{T}_{WDCLK} \times 4$)

WDT: Watchdog Timer (II)

- 32-bit with programmable time-out.
- Clocked from the IRC, RTC oscillator, or the peripheral clock.
- Cannot be disabled by software.

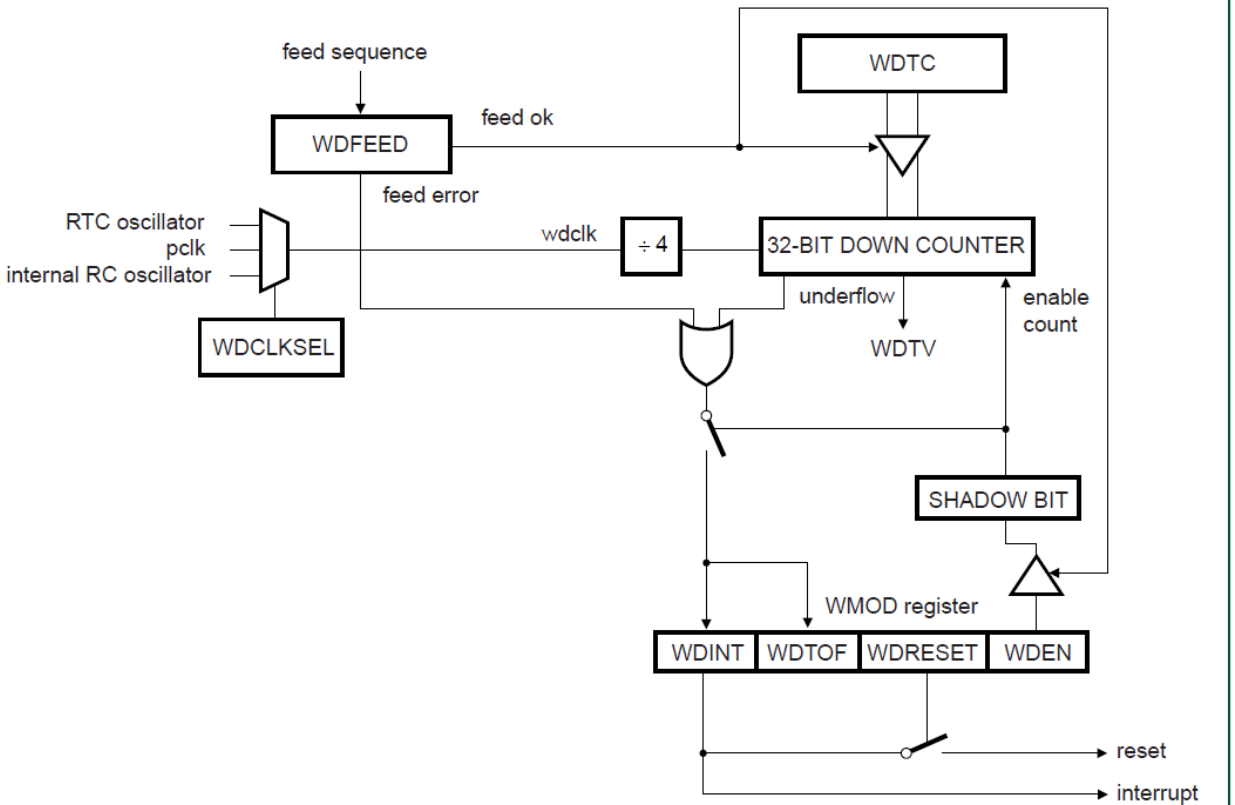
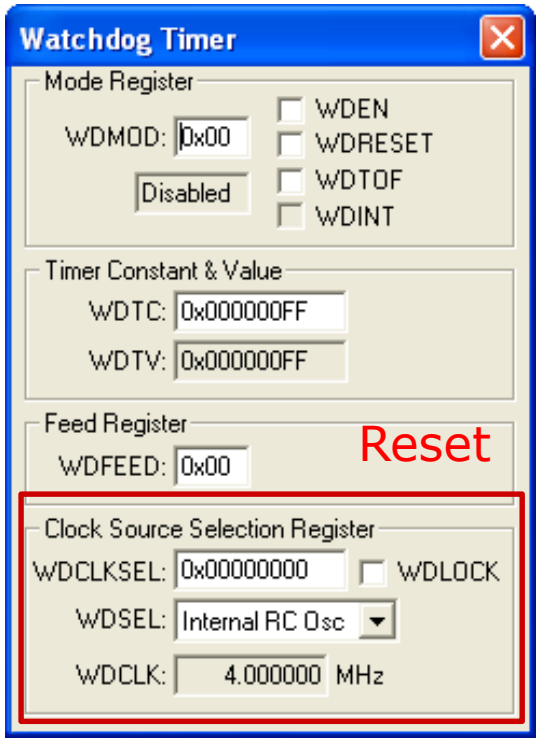


Fig 132. Watchdog block diagram



WDT: Description

- The Watchdog consists of a divide by 4 fixed prescaler and a 32-bit counter.
- The timer decrements when clocked.
- The minimum value from which the counter decrements is 0xFF.
  - The minimum Watchdog interval is ($\mathbf{T}_{WDCLK} \times 256 \times 4$)
  - The maximum Watchdog interval is ($\mathbf{T}_{WDCLK} \times 2^{32} \times 4$) in multiples of ($\mathbf{T}_{WDCLK} \times 4$).
- The Watchdog should be used in the following manner:
  - **Set the Watchdog timer constant reload value in WDTC register.**
  - **Setup the Watchdog timer operating mode in WDMOD register.**
  - **Enable the Watchdog by writing 0xAA followed by 0x55 to the WDFEED register:**

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

  - **The Watchdog should be fed again before the Watchdog counter underflows to prevent reset/interrupt.**

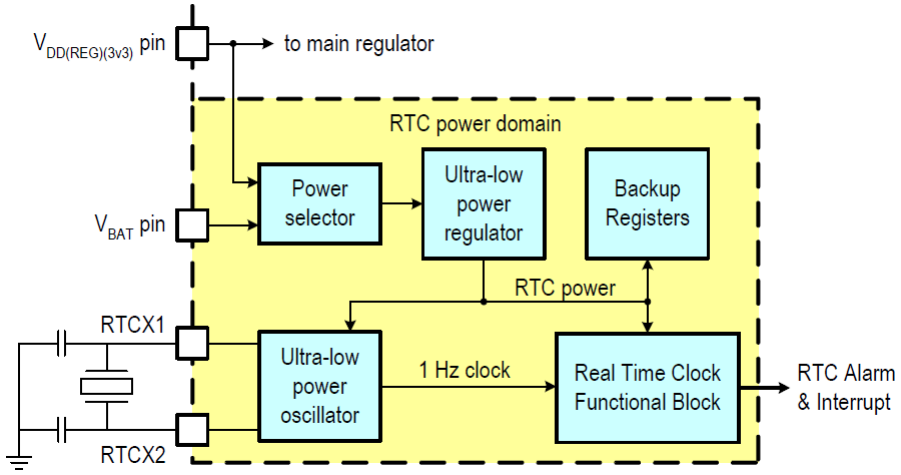
WDT: Registers description

Table 522. Watchdog register map

Name	Description	Access	Reset Value[1]	Address
WDMOD	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	R/W	0	0x4000 0000
WDTC	Watchdog timer constant register. This register determines the time-out value.	R/W	0xFF	0x4000 0004
WDFEED	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	WO	NA	0x4000 0008
WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	RO	0xFF	0x4000 000C
WDCLKSEL	Watchdog clock source selection register.	R/W	0	0x4000 0010

RTC: Real-Time Clock

- Ultra-low power 32 kHz oscillator provides 1 Hz clock to the RTC.
- Separate battery power supply.Uses CPU power supply, when present.
- Calibration mechanism.
  - ± 1 second per day
- Battery-backed registers.



- Alarm function generates interrupts
  - Wakes CPU from reduced power modes.
  - 1 second resolution.
- Extremely low power consumption
  - 390 nA (typical @ 25º C)
- Calendar function does not require CPU involvement.
  - RTC works with Vbat as low as 2.1 V.

RTC: Config. registers

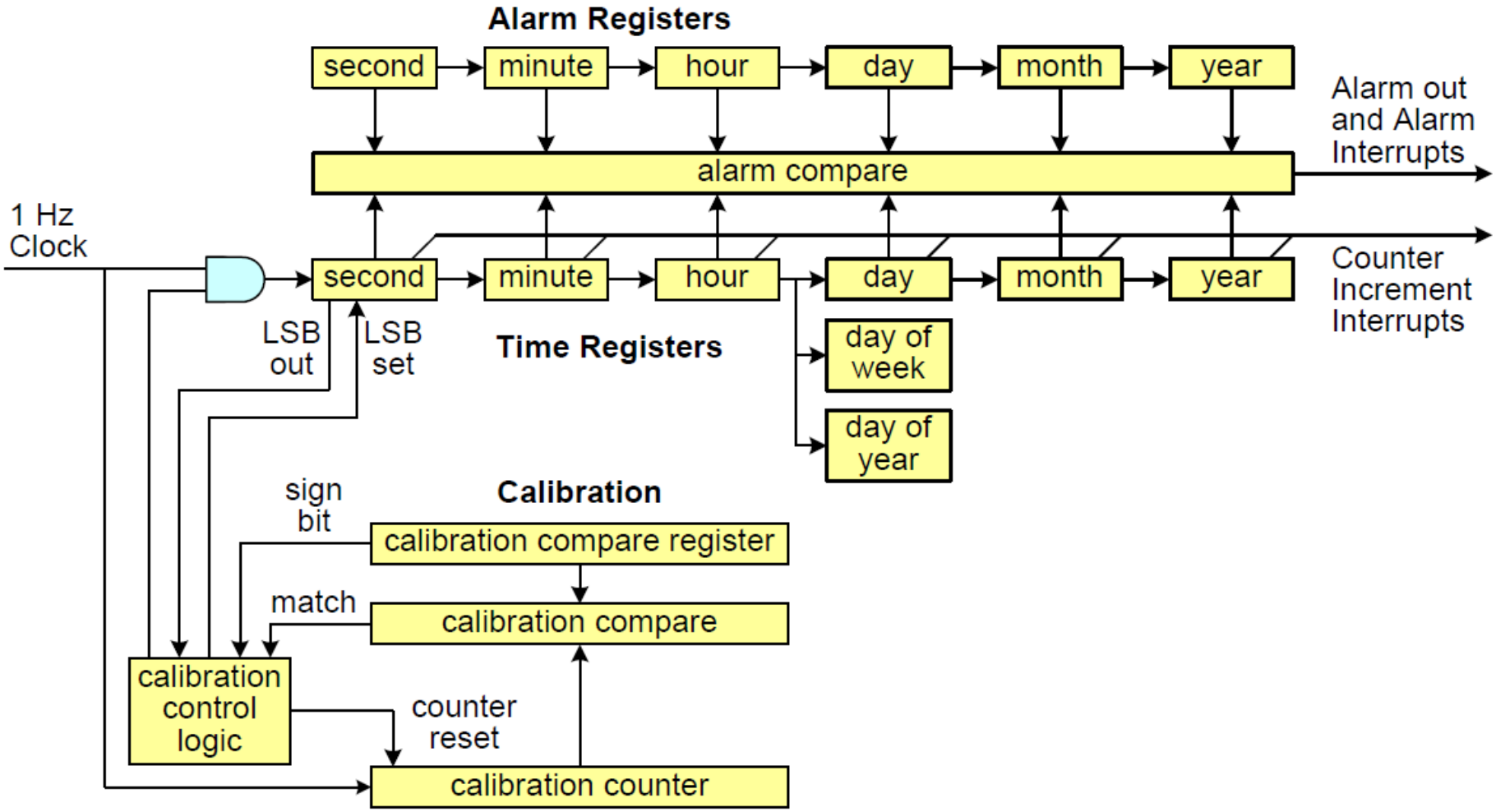
Table 508. Interrupt Location Register (ILR - address 0x4002 4000) bit description

Bit	Symbol	Description	Reset value
0	RTCCIF	When one, the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.	0
1	RTCALF	When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.	0
31:21	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 509. Clock Control Register (CCR - address 0x4002 4008) bit description

Bit	Symbol	Value	Description	Reset value
0	CLKEN		Clock Enable.	NC
		1	The time counters are enabled.	
		0	The time counters are disabled so that they may be initialized.	

RTC: Block Diagram



RTC and WDT: program example (I)

- RTC→Periodical Interrups at 1 seg. (incremens of 1SEG register).
- WDT Timeout = 1 and 2 seg. → Efects?

```
void init_RTC(void)
{
  LPC_SC->PCONP|= (1<<PCRTC);           // POwer ON
  LPC_RTC->CCR|=(1<<CLKEN)|(1<<CCALEN); // Calib. OFF
  LPC_RTC->CIIR|=(1<<IMSEC);            // Interrup incremento segundos

  set_time();
}

void RTC_IRQHandler(void)
{
  flag_segundo=1;
  LPC_RTC->ILR|=(1<<RTCIF);
}

void WDT_Feed(void)
{
  LPC_WDT->WDFEED=0xAA;
LPC_WDT->WDFEED=0x55;

}
void init_WDT(void)
{
  LPC_WDT->WDTC=F_wdclk*2; // Timeout=1seg.--> Reset CPU no se ejecuta
  LPC_WDT->WDCLKSEL=0x01;  // Clock=PCLK
  LPC_WDT->WDMOD=0x03;    // Enable y Reset if Timeout
  LPC_WDT->WDFEED=0xAA;
  LPC_WDT->WDFEED=0x55;
}
```

RTC and WDT: program example (II)

```
int main(void)
{
LCD_Initialization();
NVIC_SetPriorityGrouping(2);
NVIC_EnableIRQ(RTC_IRQn);
init_RTC();
LCD_Clear(Black);
init_WDT();
while(1){
        while(flag_segundo==0); // Espera interrupción cada 1 segundo
        flag_segundo=0;
        sprintf(buffer,"%02d/%02d/%04d   %02d:%02d:%02d",LPC_RTC->DOM,LPC_RTC->MONTH,LPC_RTC->YEAR,LPC_R
        GUI_Text(10, 10, buffer, White, Black);
        WDT_Feed();
        }
}
```