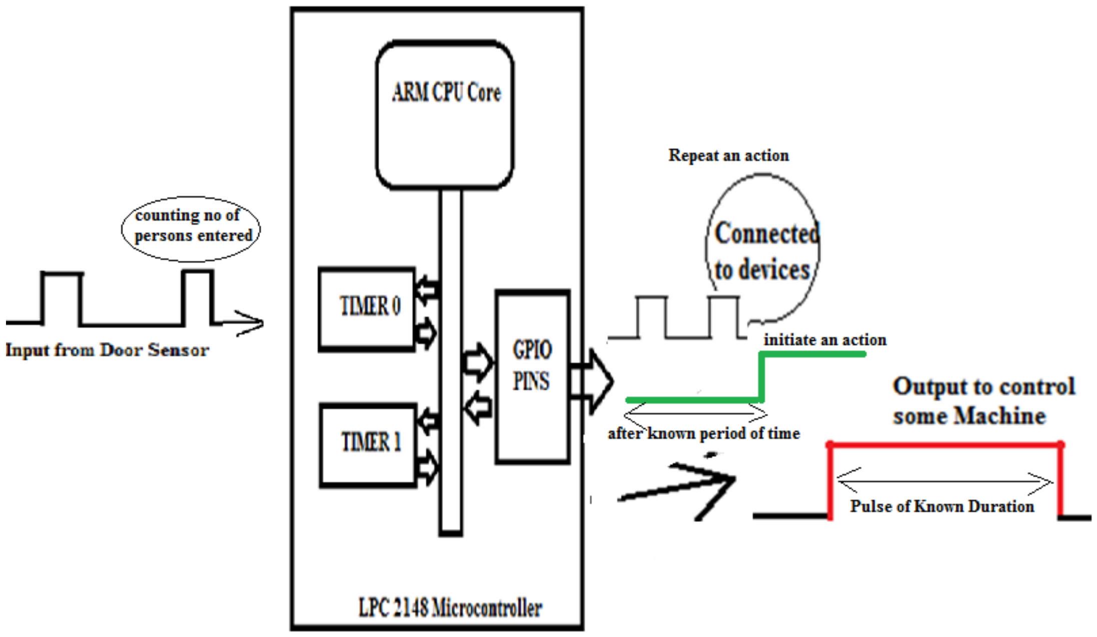
LPC 2148 Timers and Programming

Introduction

Necessity of Hardware Timers: Writing delay programs using for/while loops involve CPU getting engaged in execution (blocked). Producing accurate delays, requires proper calculation of number of iterations the loop should run. It depends on the compiler and CPU clock (CCLK), any change requires re-calculation. To overcome these disadvantages, hardware timers are provided inside the microcontrollers. Hence, timing related tasks like delay programs, generating waveforms, timing the events, counting the events can be delegated to timers to achieve accuracy, easy of programming and avoiding the blocking of CPU.

What is Timer/Counter : Timer/counter is programmable independent hardware block present inside the microcontroller [Programmable -> meaning the working of this timer circuit is configurable, based on the programmer requirements.]. The timer hardware can be used as timers or counters. Programmers can use the timers and GPIO pins to generate/time external events. LPC 2148 provides two independently working timers, timer0 and timer1.



Difference between Timer and Counter: The timer/counter hardware requires the clock to work. As we input the clock, timer/counter starts counting the clocks 0,1,2.. to N-1.(N refers to size of timer/counter). For 8 bit timer it counts from 00 to FF (255),once it reaches FF, overflows to 0 and again continue counting from 0. LPC 2148 provides 32 bit timers, hence counts from 00000000h to FFFFFFFFH.

Clock to the timer/counter hardware can be given from two sources. Based on the source of the clock they function as timer or counter,

Timer : the clock source is, the internal clock (PCLK)
Counter: the clock source is external (from any other external circuit/ peripheral/ device). Hence, Timers uses internal clock (PCLK) whereas the counters uses external clock. A **Timer** is used for producing precise time delays, used to repeat or initiate an action after / at a known period of time. A **Counter** is used to count the events happening outside the microcontroller.

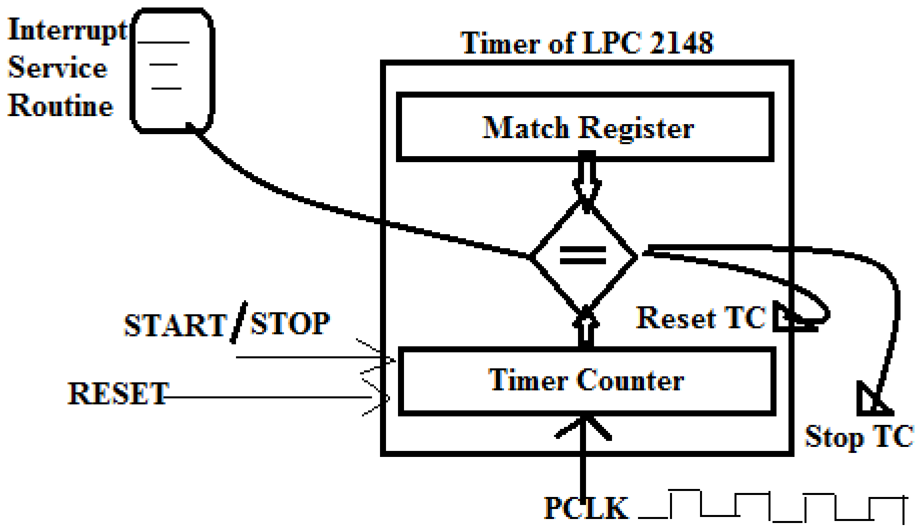
Applications of Timers and Counters

Some of the applications of timers are,

- To generate precise delays, Interval timer for counting internal events
- To repeat an action after a known period of time
- To initiate an action after a known period of time
- To generate precise pulse of given duration, to trigger external time critical events
- Operating systems use a timer to schedule the different processes/threads/tasks.
- As a waveform generator, to generate a PWM signal to drive the servo-motor or a dc-motor.
- If timers are used as counters: It can be used as pulse counter , Pulse width Demodulator. (As a pulse counter, to count pulses on a specific hardware pin, for each pulse the counter is counting one step, used to measure no of people/items moved, for door sensor. Combine this with a second timer one can measure pulses/second (velocity).

In general, timers are used for producing precise time delay. Secondly, it can be used to repeat or initiate an action after/at a known period of time. This feature is very commonly used in several applications. These enable to precisely time processes, generate signals and count events.

Working Principle of LPC 2148 Timers:



ARM LPC 2148 has two 32 bit timers. The basic block of timer is TC (TimerCounter). It works as up-counter, counting from 00000000h to FFFFFFFFh, for every clock cycle, supplied by PCLK internally. The timer can be enabled (START) or disabled (STOP) or RESET (TC=0) through program, using the bits provided in the TCR (Timer Control Register). Hence they are called as programmable timers.

How long i.e how many clock cycles should the TC count, it depends on how much time/delay the programmer has to produce. Let us assume for 12MHz crystal, CCLK = 60MHz (PLL is enabled), PCLK = 15MHz (default value VPB divider).

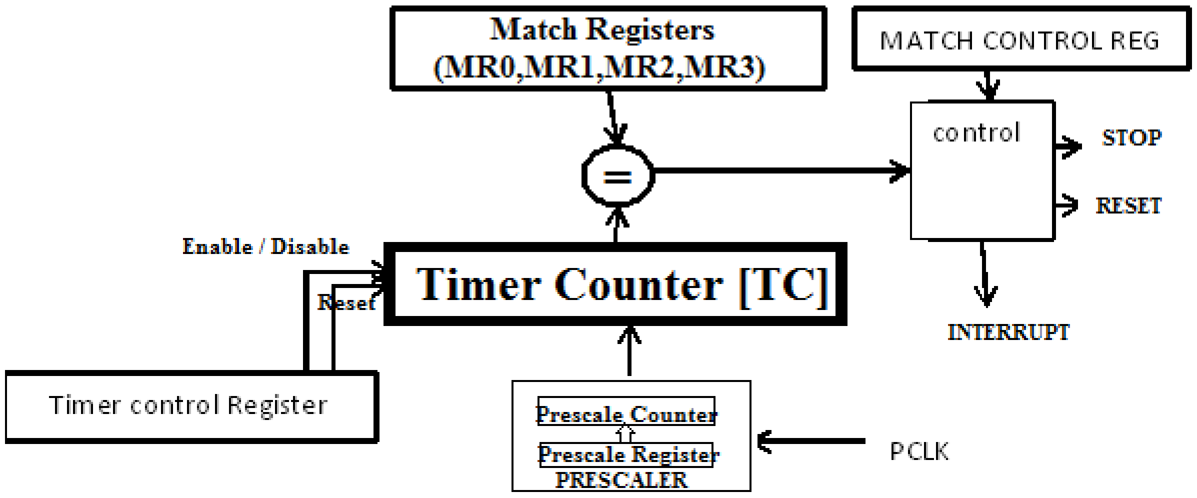
Time taken for counting one PCLK cycle (TC=1)
T = 1 / PCLK = 1 / (15MHz) = 0.067 micro seconds
Similarly, if TC = 15000 (i.e counted 15000 PCLK pulses), time taken by timer is
= 15000 x T = 15000 x [1 / (15MHz)] = 1 milli seconds.

To make the timer TC stops once it reaches required count, timer is provided with MR (Match Register). MR is initially loaded with the value/number by the programmer, indicating the number of clock cycles, TC should count.

When, the TC reaches the MR value (TC=MR), an action is initiated by the timer. Timer can be programmed to do the following actions: Make TC stops counting , Make TC get reset , Generate an Interrupt to execute the required task/program.

Programming LPC 2148 Timers

LPC 2148, has two 32 bit timer/counters named, Timer0 and Timer1. When used as Timer PCLK is used for its counting, when used as a counter it uses external clock source for its counting. PCLK is fed to Timer Counter through the programmable 32 bit pre-scaler, to further divide PCLK, to realise larger delays. By loading, value 0 to pre-scaler is equivalent to PCLK given directly to TC.



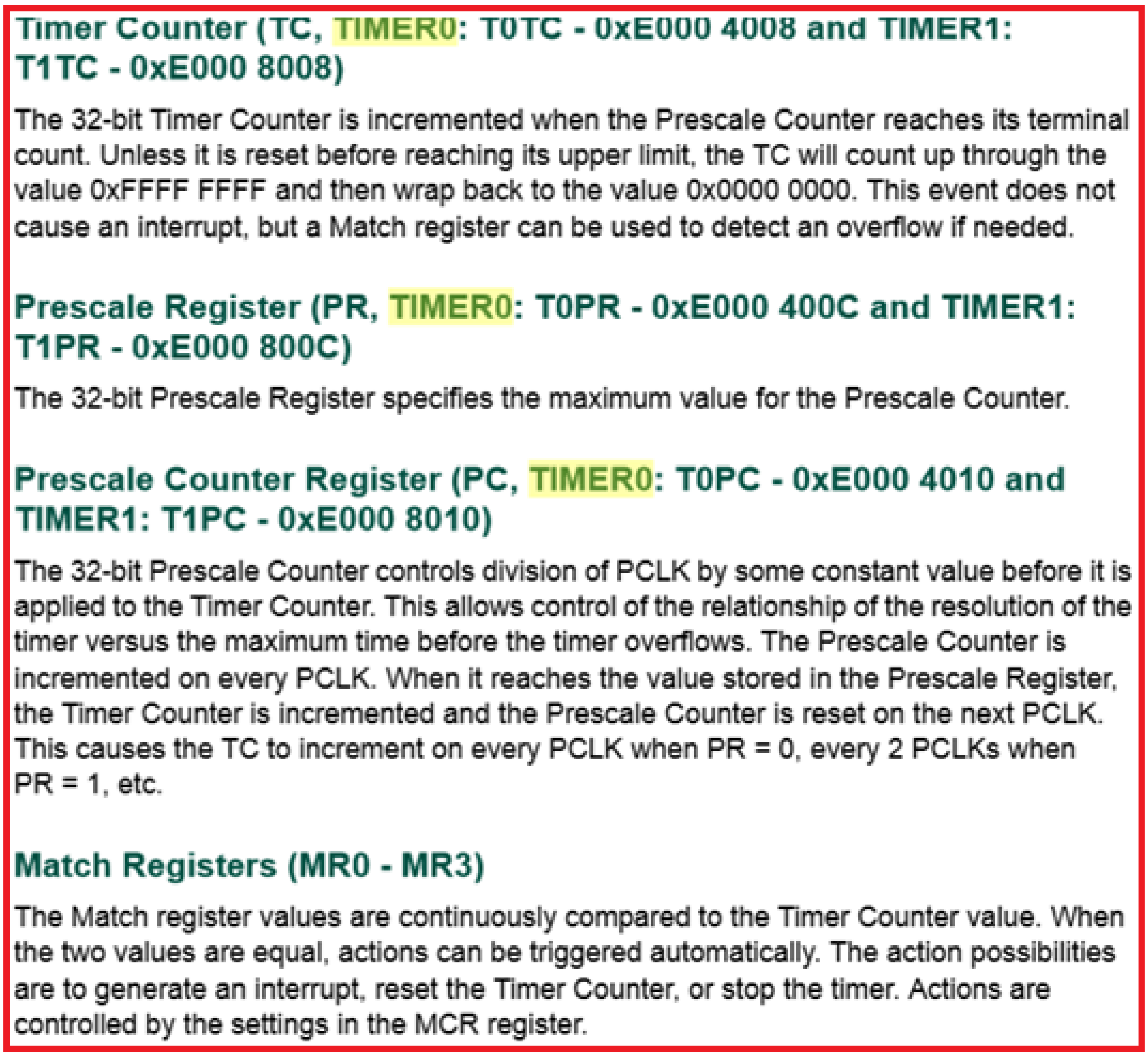
Each timer has following registers for its working,

TC – [Timer Counter]: It is a 32 bit register, counts from 00000000h to FFFFFFFFH. The 32 bit TC is incremented every PR+1 cycles of PCLK. (T0TC-for Timer0,T1TC-for Timer1)

Pre-scaler: The pre-scaler is used to further divide the PCLK, to realise higher delays.
The pre-scaler has PC and PR, they work as follows

1. **PC (Pre-scaler Counter)** – The 32 bit PC is a counter like TC, which gets incremented for every clock cycle and when it reaches the value in PR, the next clock increments TC by 1, and clears the PC. This process continues.
2. **PR Register (Pre-scaler Register)** – It holds the maximum value, the PC can reach.

Hence TC gets incremented for every PR+1 cycles of PCLK. By default, on reset, PR is loaded with 0, meaning PCLK is directly fed to TC (pre-scaler is disabled).
(T0PC,T0PR for Timer0 and T1PC,T1PR for Timer1)



Timer Counter (TC, TIMER0: T0TC - 0xE000 4008 and TIMER1: T1TC - 0xE000 8008)

The 32-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

Prescale Register (PR, TIMER0: T0PR - 0xE000 400C and TIMER1: T1PR - 0xE000 800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Prescale Counter Register (PC, TIMER0: T0PC - 0xE000 4010 and TIMER1: T1PC - 0xE000 8010)

The 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next PCLK. This causes the TC to increment on every PCLK when PR = 0, every 2 PCLKs when PR = 1, etc.

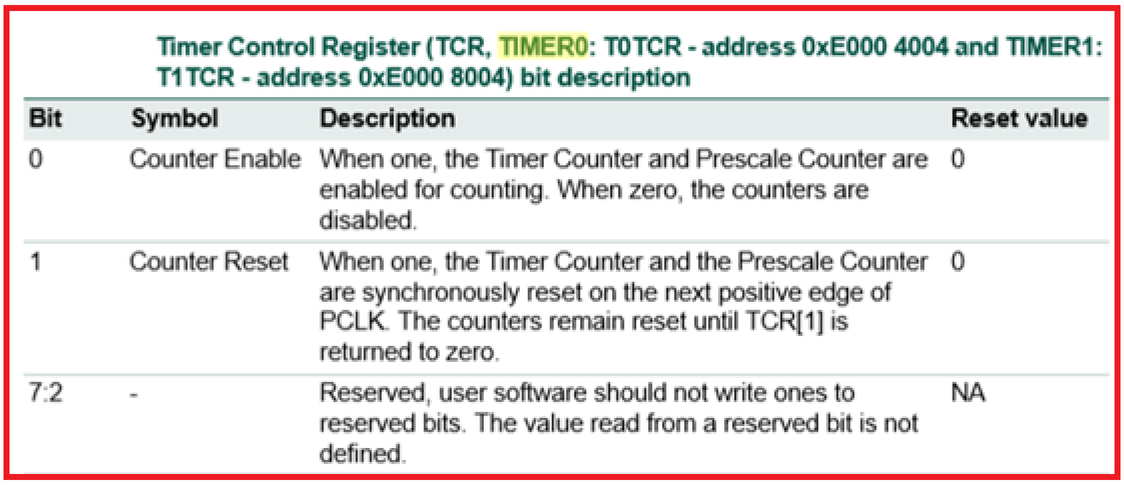
Match Registers (MR0 - MR3)

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

TCR- [Timer Control Register] , 8 bits, used to enable / disable and/or reset timer (TC).

D7	D6	D5	D4	D3	D2	D1	D0
						R	E

E (D0 bit) = writing ‘1’ to this bit, enable the timer , =0 to disable the timer.
R (D1 bit) = writing ‘1’ to this bit reset the TC to zero, (after resetting, write ‘0’)
(TOTCR for Timer0 and T1TCR for Timer1)


Timer Control Register (TCR, TIMER0: T0TCR - address 0xE000 4004 and TIMER1: T1TCR - address 0xE000 8004) 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
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

MR0,MR1,MR2,MR3 – 4 Match Registers, 32 bits, whose values are compared with TC value, on match certain action is performed based on the MCR (Match Control Register).

MCR – Match Control Register, 16 bits, 3 bits for each Match register:

D15	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	S	R	I	S	R	I	S	R	I	S	R	I
		MR3			MR2			MR1			MR0	

The lowest three bits (D2,D1,D0) are for controlling the operations related to the Match register 0, next three for MR1, MR2 and MR3, in that order. The meaning of the bits is explained below,
I - When ‘1’ , an interrupt is activated when match occurs, ‘0’ interrupt is disabled
R- When ‘1’, the timer count register is reset when match occurs,’0’-feature disabled
S- When ‘1’ , the TC and PC will be stopped when match occurs, also timer is disabled.

Hence, four 32 - bit match registers (MR0-3) allows:

- ➢ Continuous operation with optional interrupt generation on match.
- ➢ Stop timer on match with optional interrupt generation.
- ➢ Reset timer on match with optional interrupt generation.

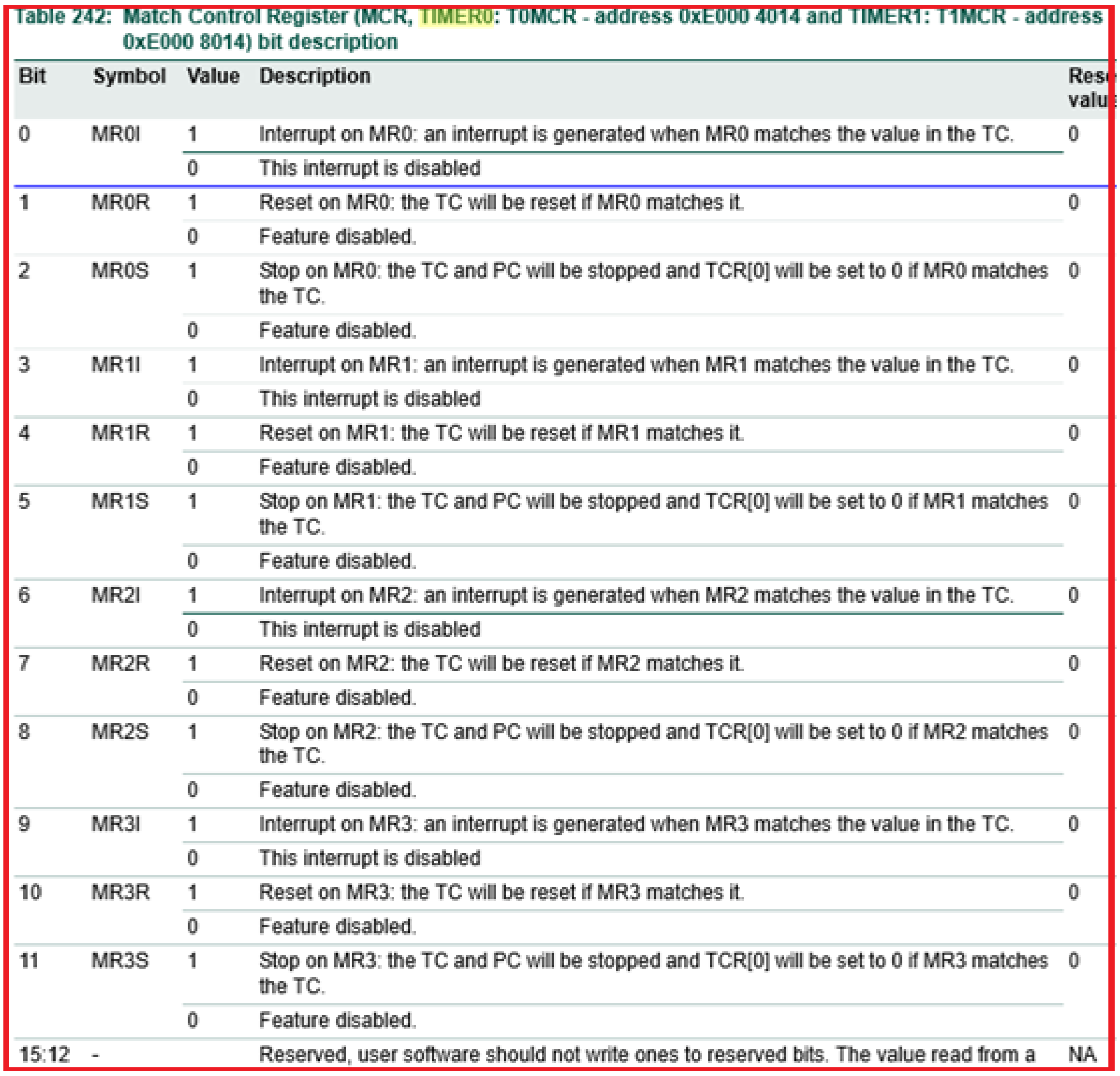
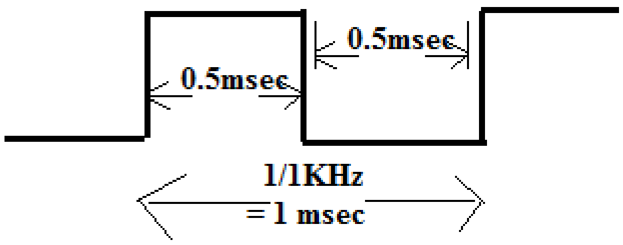


Table 242: Match Control Register (MCR, TIMER0: T0MCR - address 0xE000 4014 and TIMER1: T1MCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Res valu
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.	
15:12	-		Reserved, user software should not write ones to reserved bits. The value read from a	NA

Example: Generate the square wave of frequency 1KHz using the timer, on P1.16pin using timers. (By connecting Buzzer to this pin, we can here the beep sound)

Assume Timer0 is used to generate this frequency.



Steps to implement the Delay Program using Timer:

1. Calculate the number/count to be loaded in match register, MR0
   Let us assume **PCLK = 15 MHz** & **Prescaler is loaded with Zero (T0PR = 0)**
   (for crystal frequency= 12MHz, CCLK - 60MHz if PLL enabled and default value of VPBR (=0) gives PCLK = CCLK %by 4 = 60MHz/4 = 15MHz)

   count = **Td / T**
   = Time period of required output(1KHz) / time period of input frequency(PCLK)

   Td = 1/1KHz = 1 msec , half of it is **0.5msec**;
   T = 1/15MHz = 0.067µsec
   = **0.5 msec / 0.067µsec = 7462** (TC counting this many times produces 0.5msec)
   Load this number into MR0, MR0 = 7462
2. Load the MCR for stopping the timer on match & disable the interrupt
3. Start the timer, by enabling the 'E' bit in TCR
4. Now TC starts counting, when it matches with the MR value, it stops counting
5. Stop the timer

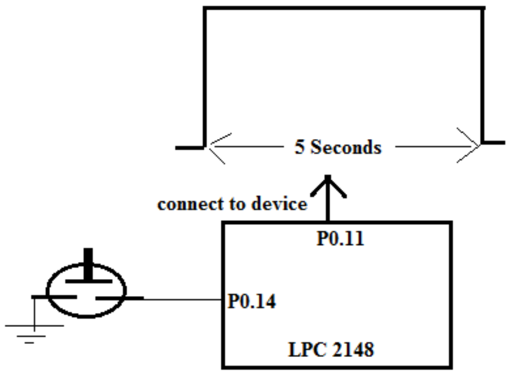
```
//Program to generate 1KHz on  P1.16 using delay produced by the timer

#include  <LPC214x.h>

void   delay(void)
{
 T0TCR = 1;  //start the timer
 While (!(T0TC == T0MR0));
 T0TCR = 2; // reset the counter  and stop the timer
}
```

```
int main(void)
{
      T0MR0 =  7462;  //use the Timer0 and load the MR0 with count
      T0MCR = 0X0004;  //   0000….100 – Stop the timer, after match
      I0DIR1 = 0X00010000; //make P1.16 as output
      while(1)  // program to produce square waveform  of 1 KHz
         {
            I0SET1   =  1 << 16;  //set P1.16 to 1
            delay( );
            I0CLR1  =  1 <<16;  //clear P1.16 to 0
            delay( );
         }
}
```

Question: Interface one AC gadget (say bulb or some machine) to LPC 2148. Write a program to make it on for 5seconds, each time a key is pressed using the timers. (OR question can be as follows)
Generate a 5 second pulse on P0.11 pin of LPC 2148



Assumptions:

1. P0.11 is connected to AC gadget through the RELAY, P0.14 is connected to switch (if switch is pressed it generates logic ‘0’, else logic ‘1’
2. **Assume CCLK = 60MHZ, PCLK = 60000 (PLL enabled, VPBDIV=1)**

Delay Calculation:

Let us use Timer0 to produce 5 sec delay. Let us use Prescaler to implement the delay. If Prescaler (T0PR) is loaded with 99, it will generate one clock cycle for every 99+1=100 PCLK clock cycles.
Now for PCLK of 60MHz, Load **PR = 60000-1 = 59999.**
Total time delay produced by Prescaler = 60000 x (1 / 60MHz) = 1msec
(for every 1msec, prescaler produces 1 clock pulse to TC, so every count of TC means, 1msec is over, if TC=1000, means delay produced = 1000 x 1msec= 1sec)
Required Delay = 5sec = 5000 milli-seconds.
Total Count = 5000 milli-seconds / 1milli-second = 5000 (load this to MR0)

```
#include  <LPC214x.h>
#define SW2 (IO0PIN & (1 << 14))
delayHW(void)
 {
   T0PR = 60000-1; // produces 1msec delay
   T0MR0 = 5000;   // produces 5000 x 1msec delay = 5seconds
   T0TCR = 1;   // Start the timer
   while (!(T0TC == T0MR0)); // wait for TC to reach MR0
   T0TCR = 2;  // Stop the timer, reset the TC,PC
 }
```

```
int main(void)
 {
     IODIR0  =  (1<<11);    // P0.11 as OUTPUT
     while(1)
         {
           if(!SW2) // wait for key press, if yes produces 5 sec pulse
                 {
                  I0SET0  =  1 << 10;  //set P0.10 to 1
                     delayHW( );
                     I0CLR0  =  1 <<10;  //clear P0.10 to 1
                   }
         }
}
```

Not in Syllabus:
Extra Information (Other facilities & registers provided by timers)

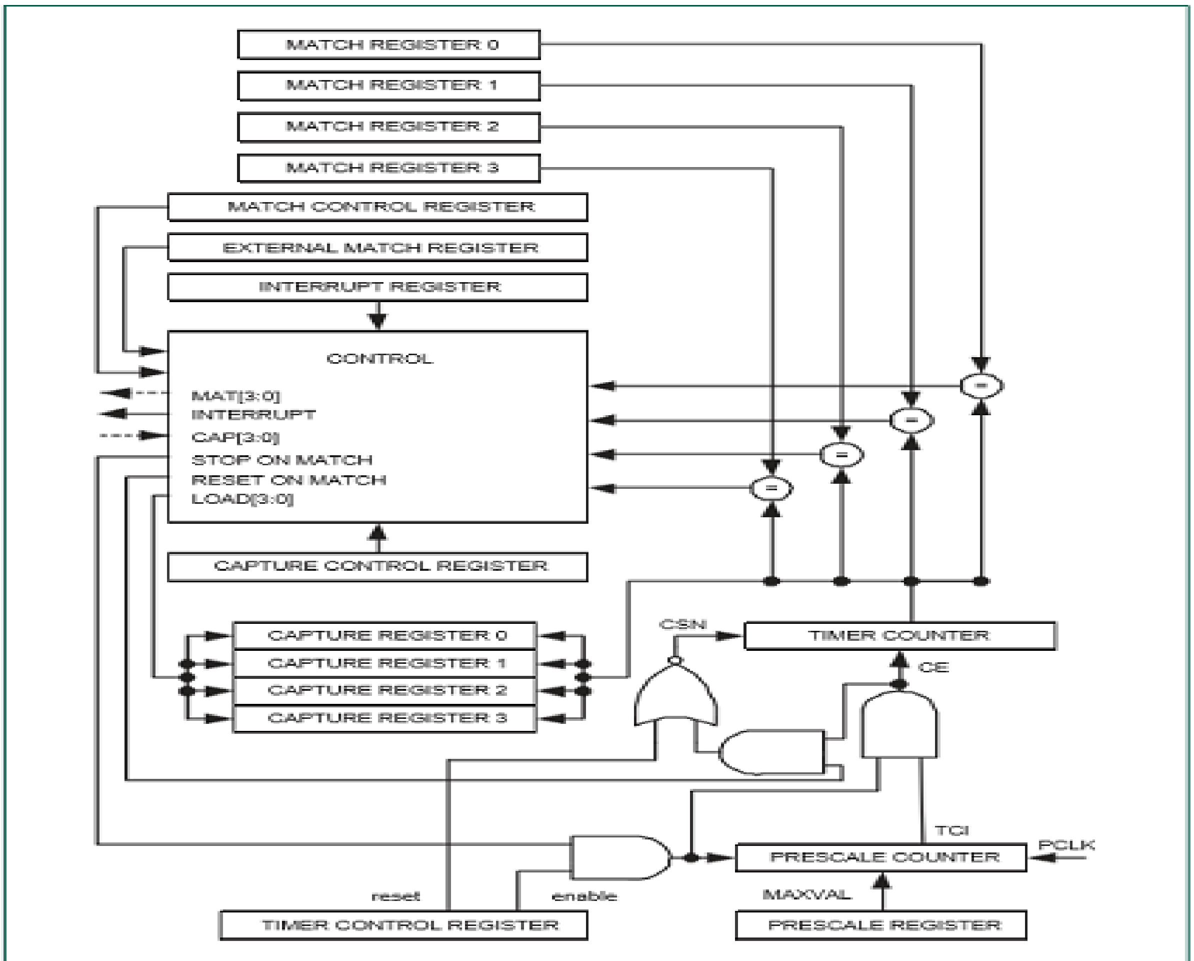


Fig 58. Timer block diagram

The timer / counter is designed to count cycles of the peripheral clock (PCLK) or an externally-supplied clock, and can optionally generate interrupts or perform other actions at specified timer values, based on four match registers. It also includes four capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

Capture Signals/Pins : Each timer is provided with four capture pins. A transition on a capture pin can be configured to load one of the capture registers with the value in the timer counter (TC) and optionally generate an interrupt. CAP0.0,CAP0.1,CAP0.2,CAP0.3 are the pins for timer0 and CAP1.0, CAP1.1, CAP1.2, CAP1.3 are the pins for timer1. Ex: CAP0.0 is P0.2 pin.

Capture Registers (T0CR0 – TCR3 , T1CRO – T1CR3)

Each capture register is associated with a device pin and may be loaded with the TC value when a specified event occurs on that pin. The settings in the capture control register determine whether the capture function is enabled and whether a capture event happens on the rising edge of the associated pin, the falling edge, or both edges.

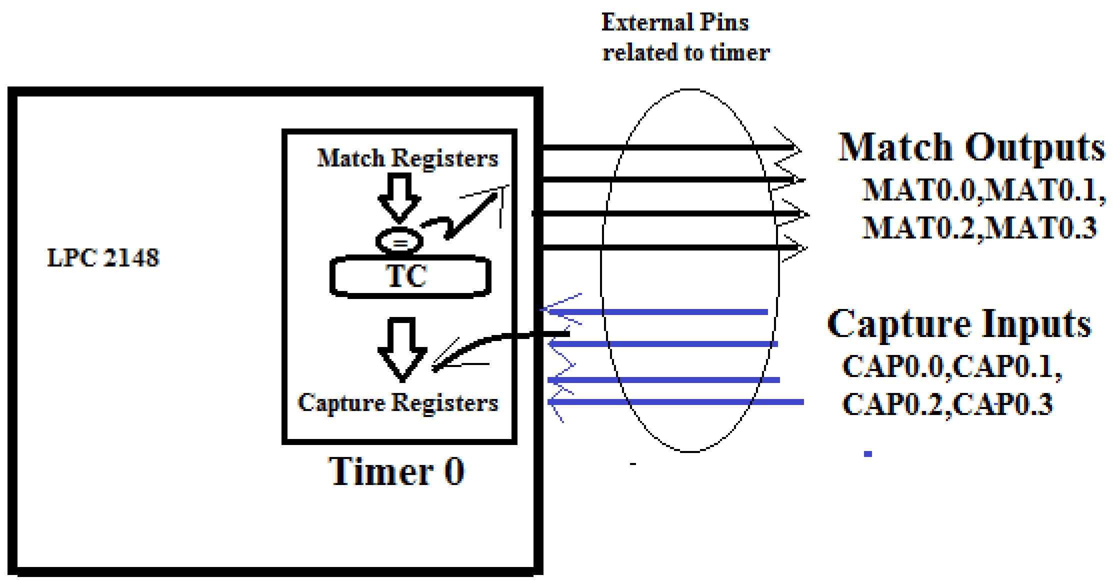
Capture Control Register (T0CCR,T1CCR)

The capture control register is used to control whether one of the four Capture Registers is loaded with the value in the TC when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges.

Match Outputs : (External Match outputs) When a match register equals the timer counter (TC), this output can either toggle, go low, go high, or do nothing. The EMR controls the functionality of this output. MAT0.0,MAT0.1,MAT0.2,MAT0.3 are the match output pins for timer0 and MAT1.0,MAT1.1,MAT1.2,MAT1.3 are the match output pins for timer1. Ex: MAT0.0 is P0.3.

External Match Register [T0EMR,T1EMR] : The external match register provides both control and status of the external match pins MAT(0-3). Ex: Bit[0] –External Match0 bit reflects the state of output MATn.0, whether or not this output is connected to its pin. When a match occurs between the TC and MR0, this output of the timer is can either toggle, go low, go high, or do nothing. Bits [5:4] control the functionality of this output [00-do nothing, 01-set the output, 10-clear the output,11-toggle the output]

IR Register : Interrupt register. This provides which of the eight possible interrupt sources are pending. This can be written to clear interrupts. The interrupt register consists of four bits for the match interrupts and four bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic '1' to the corresponding IR bit will reset the interrupt.


External Pins related to timer
LPC 2148
Match Registers
=
TC
Capture Registers
Timer 0
Match Outputs
MAT0.0,MAT0.1,
MAT0.2,MAT0.3
Capture Inputs
CAP0.0,CAP0.1,
CAP0.2,CAP0.3