System Timing

R. Ferrero , P. Bernardi

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

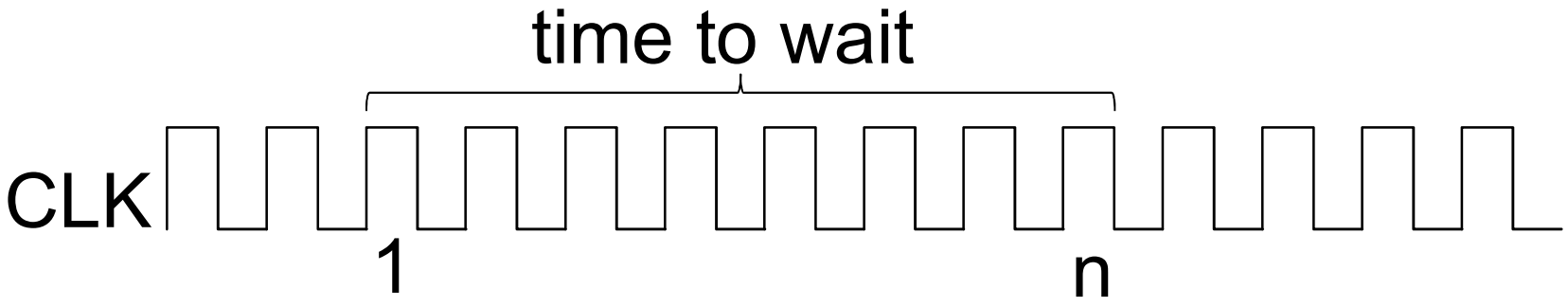
Torino - Italy

Timers

- Timers are peripheral cores for synchronizing the system, based on counting
  - to wait for a delay time
  - to perform operations at regular time.
- Usually, when a timer ends counting, the system needs to react.
  - Typically an interrupt handler is entered.
  - While timer counts, the CPU can enter a low-power mode.

Working principle

- Timers are supplied with a (dedicated) clock.
- Timer counting is based on the clock signal.
- Timers include registers to be programmed with the number of clock cycles to count.



COUNT = number of clock cycles

Counting modes

- Two philosophies exist for the timer design:
  - decreasing count: interrupt when count reaches 0
  - increasing count: interrupt when a match value is reached.
- Whatever the counting mode is, the number of clock cycles to count is computed as:

$$\text{count} = \text{time [s]} * \text{frequency [1/s]}$$

Timing computation examples

- How many clock cycles are needed to wait 10 seconds with a frequency of 25 MHz?

$$\text{count} = 10\ [\text{s}] * 25*10^{6}\ [1/\text{s}] =$$

$$= 25*10^{7} = 0\text{x}0\text{EE}6\text{B}280$$

- How many clock cycles are needed to wait 10 milliseconds with a frequency of 100 MHz?

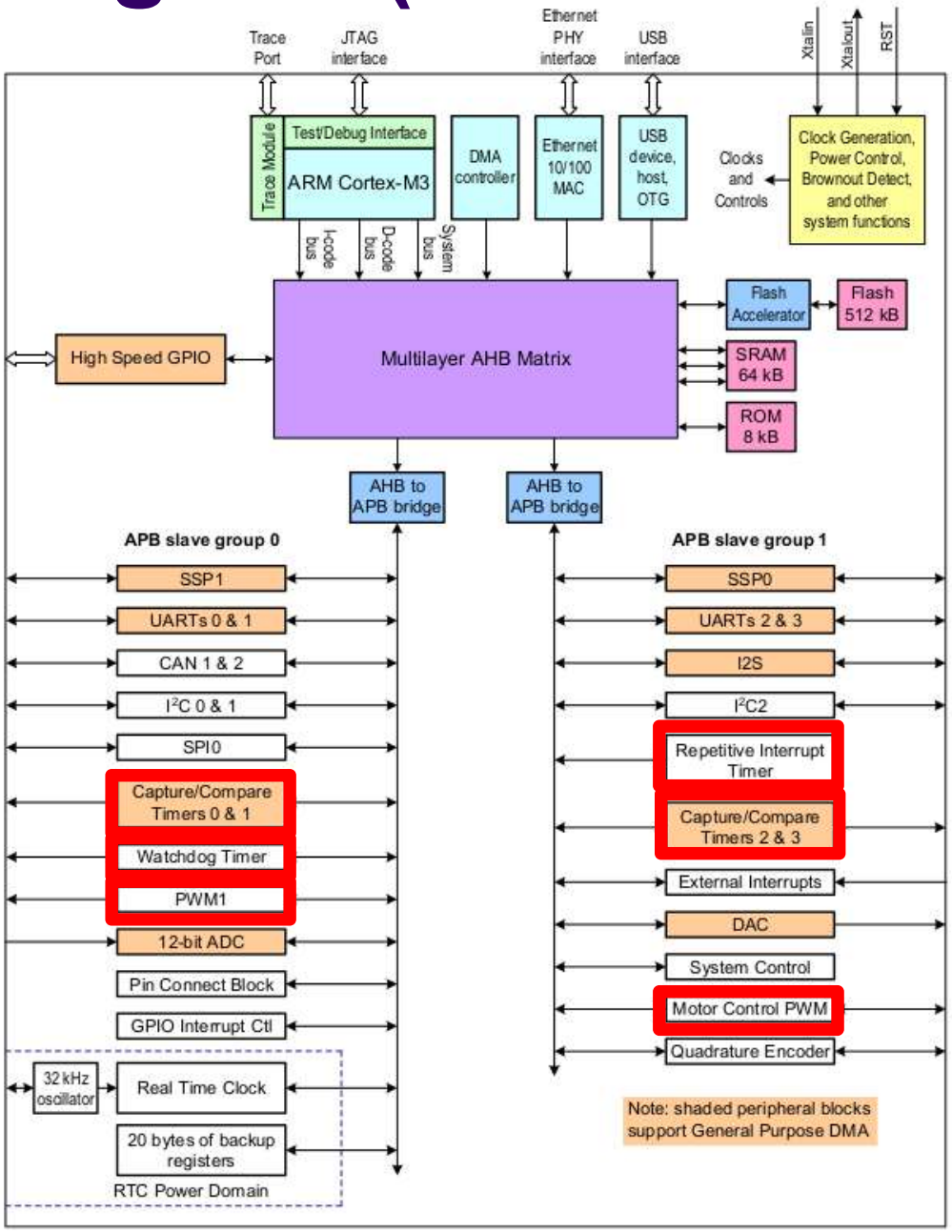
$$\text{count} = 10*10^{-3}\ [\text{s}] * 100*10^{6}\ [1/\text{s}] =$$

$$= 10^{6} = \text{count} = 0\text{x}000\text{F}4240$$

Timer count limits

- If a timing request is large, the count value could not fit in the timer register.
- Hardware and software features can be used to address this issue
  - HW – Cascade of counters
  - HW – Prescalers
  - SW – Handler software count of HW events.

Block diagram (user manual p. 10)



Timers in LPC1768 (I)

捕获/对比 定时器

- Capture/Compare timers 0-3: standard timers programmed by the user to implement delays and regular intervals.

系统节拍定时器

- System Tick Timer: it generates a fixed 10 millisecond interrupt for use by an operating system or other system management software.

重复中断定时器：他产生重复的中断，在一个指定的时间间隔，不使用额外的标准定时器

- Repetitive Interrupt Timer: it generates repeating interrupts at specified time intervals, without using a standard timer.

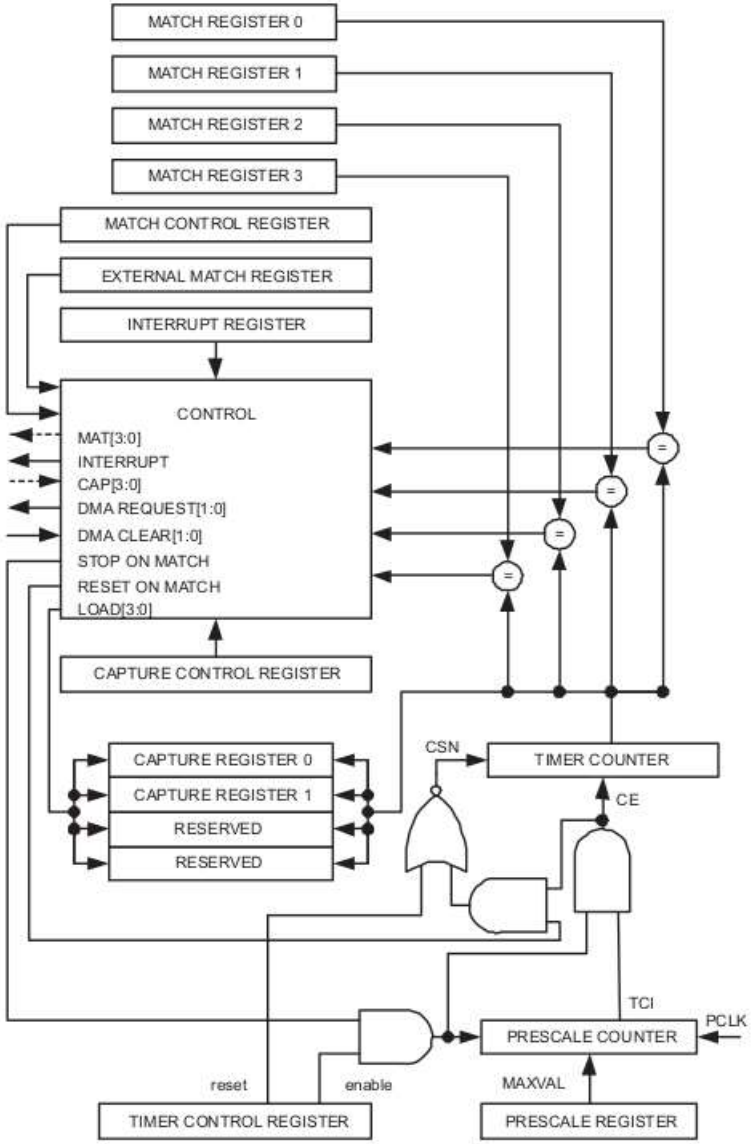
Timers in LPC1768 (II)

- PWM: it extends the standard timers for digital signal modulation.
- Motor Control PWM: optimized timer for three-phase AC and DC motor control applications. 优化的定时器用于 三相交流或者直流电机的控制应用
- Watchdog timer: it resets the microcontroller within a reasonable amount of time if it enters an erroneous state.

Standard Timers

- The Timer/Counter is designed to count cycles of the peripheral clock (PCLK) or an externally-supplied clock.
- It 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.

Timer block diagram (page 511)

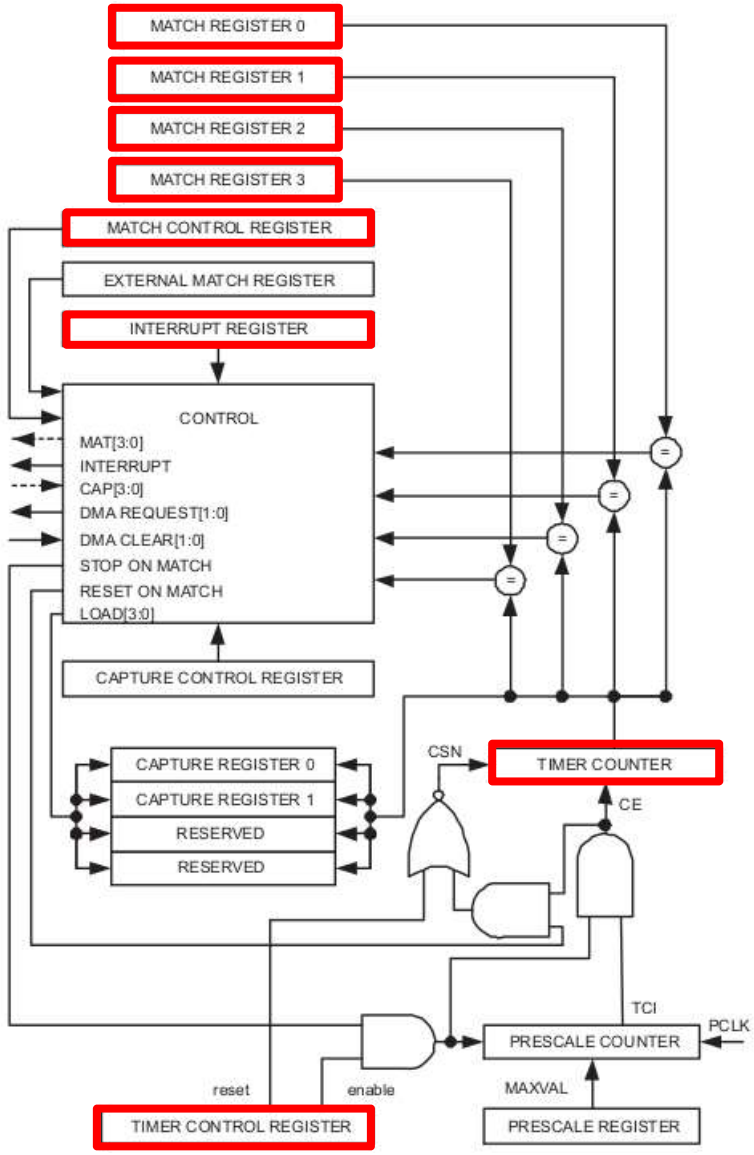


Main registers (page 503)

Table 426. TIMER/COUNTER0-3 register map

Generic Name	Description	Access
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
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
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W
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
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
MR1	Match Register 1. See MR0 description.	R/W
MR2	Match Register 2. See MR0 description.	R/W
MR3	Match Register 3. See MR0 description.	R/W

控制TC



Timer counter (TC)

- The Timer Counter counts clock cycles
  - it is incremented when the prescale counter reaches its terminal count.
- It counts up through the value 0xFFFF FFFF and then wrap back to 0x0000 0000.
- When it reaches the upper limit, it does not generate an interrupt 当他到达上限时，不会产生中断
  - a match register should be configured.
- It can be reset before reaching its upper limit.

他在到达上限前，可以被重置

Timer Control Register (TCR)

- The lowest two bits of TCR control the operation of the Timer/Counter.
- Bit 0 = 1 -> enable timer for counting.
- Bit 1 = 1 -> reset counter. The value of the timer is fixed until the value of this bit is 1.

Table 428. 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

Match Registers 匹配寄存器

- The Match register values are continuously compared to the Timer Counter value.
- When the two values are equal, actions can be triggered automatically:
  - to generate an interrupt
  - to reset the Timer Counter
  - to stop the timer

当MR和TC相等时，产生一个中断
重置TC
停止timer

以上动作由MCR控制

- Actions are controlled by the settings in the Match Control Register.

Match Control Register

- The Match Control Register controls which operations are performed when one of the Match Registers matches the Timer Counter.

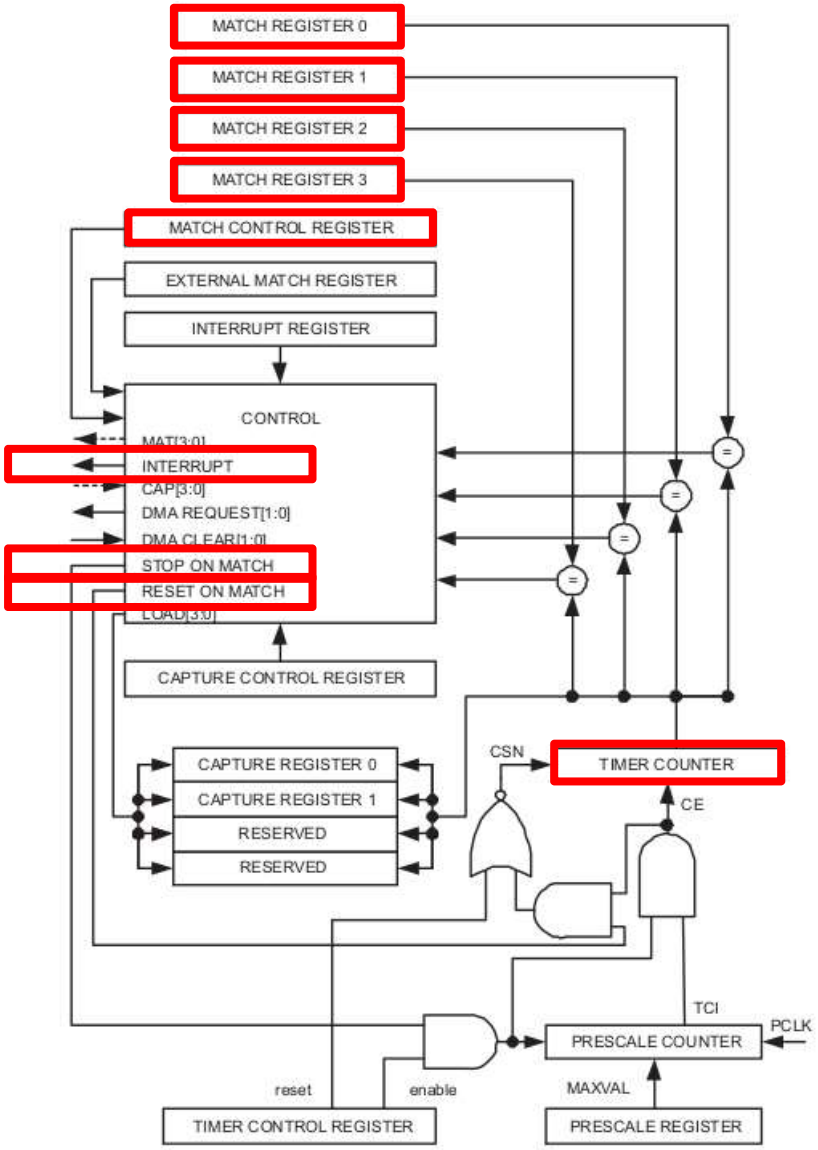
Table 430. Match Control Register (T[0/1/2/3]MCR - addresses 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014) bit description

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.	

- Bits 3-5, 6-8, and 9-11 behave in the same way for MR1, MR2, and MR3 respectively.

Three possible behaviors

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



Interrupt Register

- The Interrupt Register consists of
  - 4 bits for the match interrupts
  - 4 bits for the capture interrupts.

4bits 用来匹配中断

4 bits 用来捕获中断

- If an interrupt is generated, the corresponding bit in the IR is high, otherwise the bit is low.
- Writing 1 to the corresponding IR bit will reset the interrupt, writing 0 has no effect.

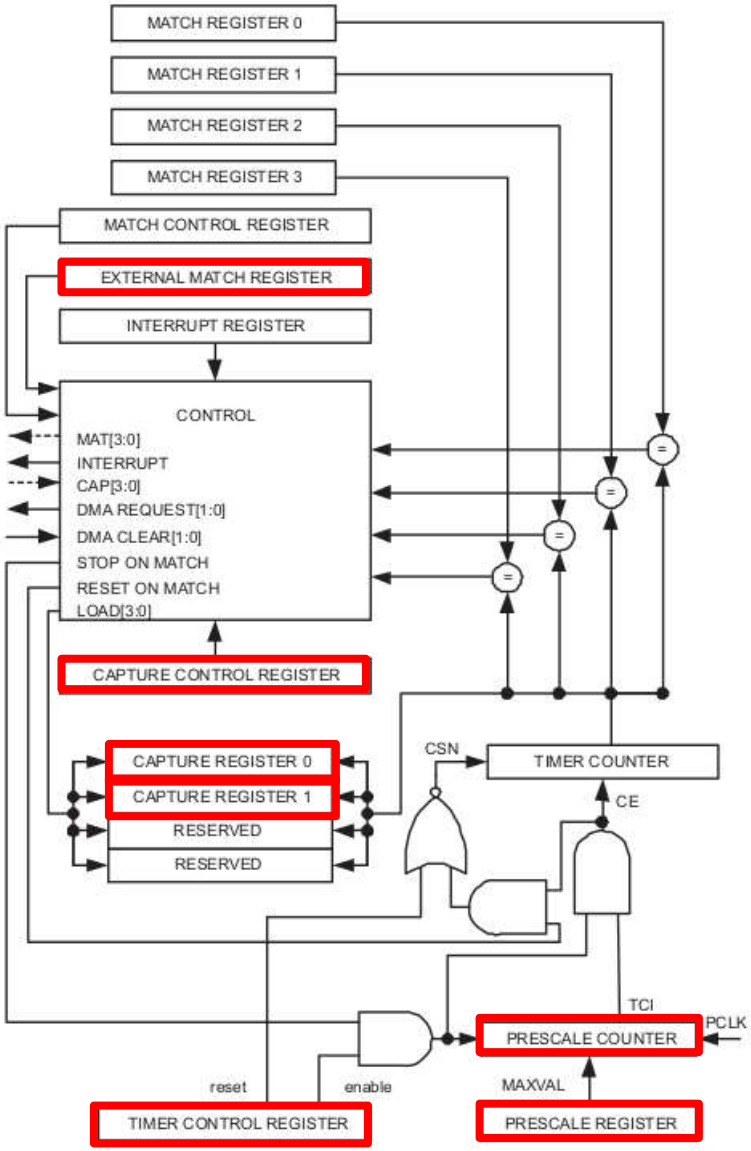
Table 427. Interrupt Register (T[0/1/2/3]IR - addresses 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000) bit description

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	-

Advanced registers (page 503)

Table 426. TIMER/COUNTER0-3 register map

Generic Name	Description	Access
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
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
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
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
CR1	Capture Register 1. See CR0 description.	RO
EMR	External Match Register. The EMR controls the external match pins MATn.0-3 (MAT0.0-3 and MAT1.0-3 respectively).	R/W
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



Registers for prescaling

- The Prescale register stores a 32-bit value
- The Prescale Counter is incremented on every PCLK.
- When the Prescale Counter reaches the value stored in the Prescale register:
  - the Timer Counter is incremented
  - the Prescale Counter is reset on the next PCLK.
- E.g.: if Prescale register is set to 1, the Timer Counter is incremented every 2 PCLKs.

如果PR被设置成1, TC每两个PCLKs增加一次

External Match Register

- For every match register, the external match register has
  - an output bit
  - a pair of configuration bits.
- When a match register equals timer counter, the related bit in the external match register changes according to the configuration bits.

Table 433. External Match Control

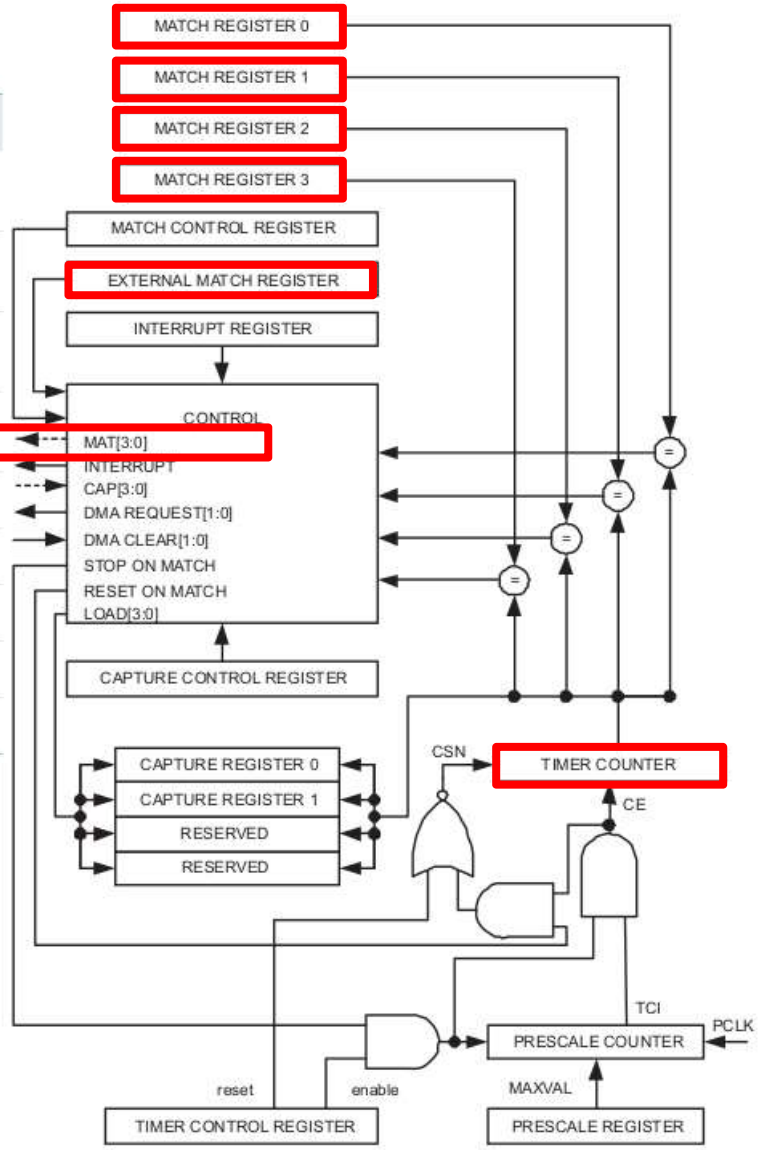
EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (MATn.m pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (MATn.m pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

External Match Register

Table 432. 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 433 shows the encoding of these bits.	00
7:6	EMC1	External Match Control 1. Determines the functionality of External Match 1. Table 433 shows the encoding of these bits.	00
9:8	EMC2	External Match Control 2. Determines the functionality of External Match 2. Table 433 shows the encoding of these bits.	00
11:10	EMC3	External Match Control 3. Determines the functionality of External Match 3. Table 433 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

- The 4 least significant bits of external match register are sent to MAT output.



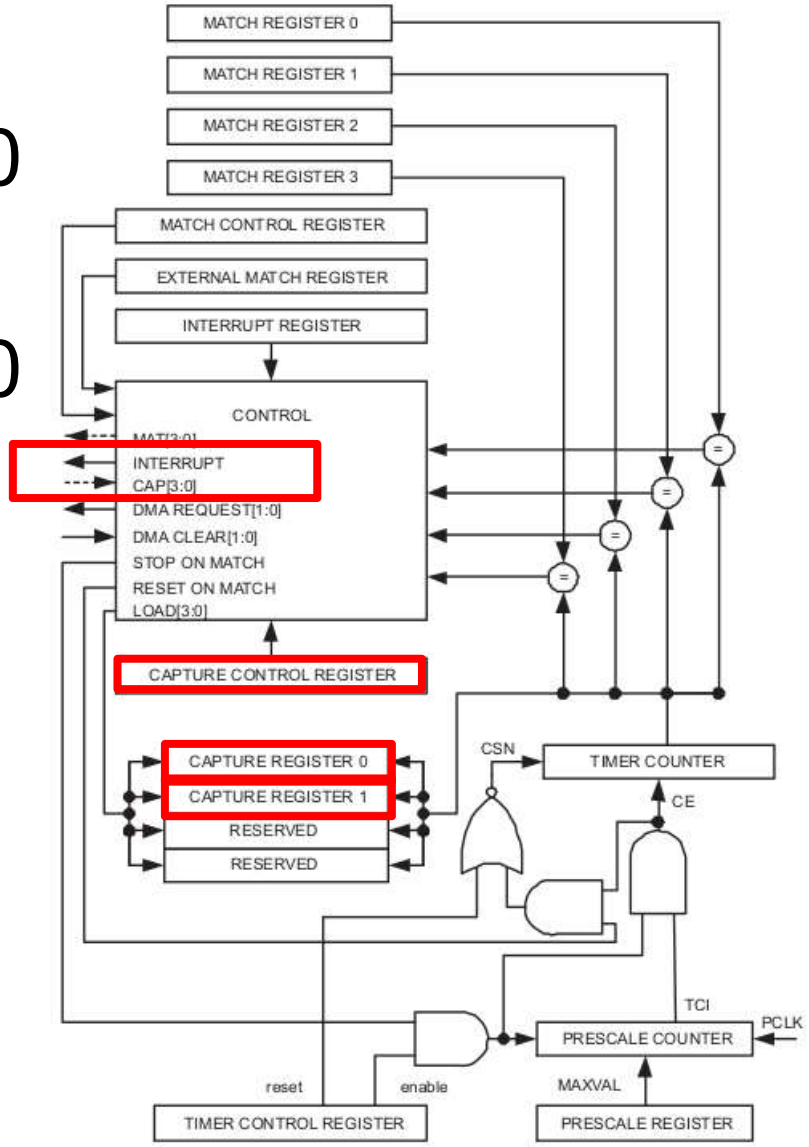
Registers for capturing an event

- 2 Capture Registers: CR0 and CR1.
- 4 Capture Control Registers (one for each pin of CAP input).
- Transition of single CAP pin can be monitored
  - rising edge and/or
  - falling edge
- In such a case, two actions are supported
  - Timer Counter value is stored in a capture register
  - an interrupt is generated.

TC 存储在CR中

Bits in Capture Control Register

- Bit 0 = 1: store TC on CR0 on rising edge of CAP pin
- Bit 1 = 1: store TC on CR0 on falling edge of CAP pin
- Bit 2 = 1: generate an interrupt with CR0 load
- Bit 3-5 are the same for CR1.

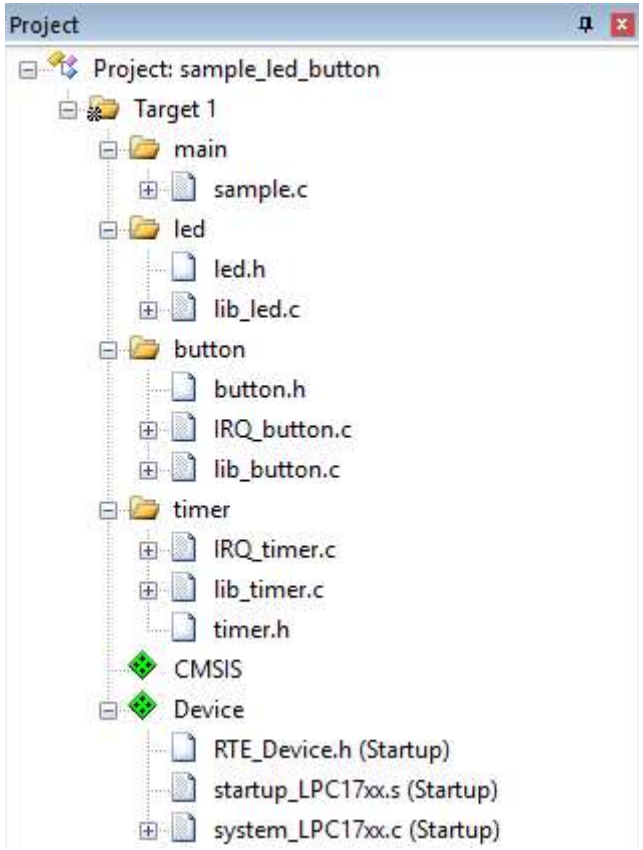


Count Control Register

- It selects between Timer and Counter mode.
- In Timer mode:
  - Prescale Counter is incremented on every PCLK.
  - when the Prescale Counter reaches the value of Prescale register, Timer Counter is incremented.
- In Counter mode:
  - a selected bit of CAP is sampled on every PCLK
  - is the expected transition of CAP bit is recognized (rising and/or falling edge or no change), then the Timer Counter is incremented.

Timer library

- lib_timer.c
  - init_timer(timer_num, timerInterval)
  - enable_timer(timer_num);
  - disable_timer(timer_num);
  - reset_timer(timer_num);
- IRQ_timer.c
  - TIMER0_IRQHandler();
  - TIMER1_IRQHandler();



Delay setup example

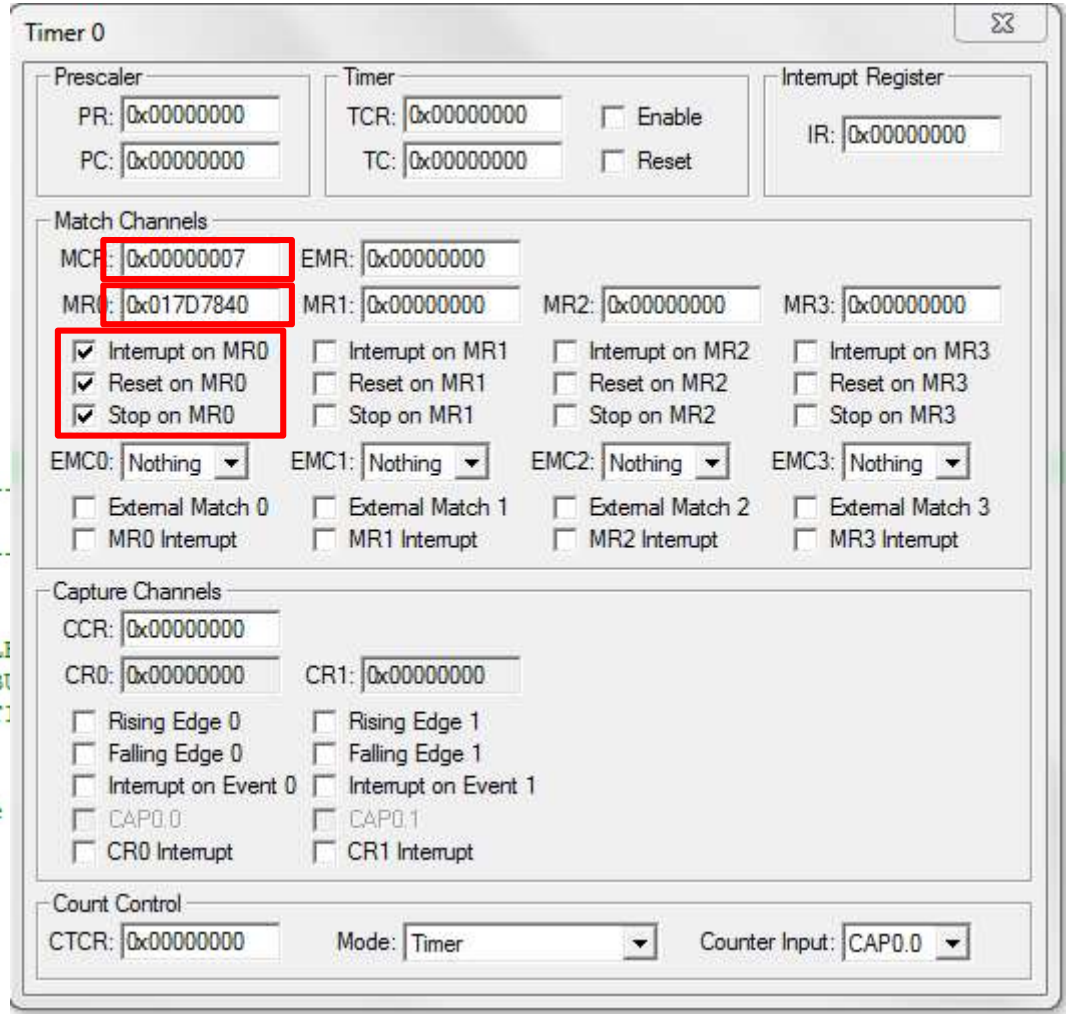
- Setup a 1 s delay with 25 MHz frequency
- Raise an interrupt, reset and stop TC

```
/*----------------------------------------------------------------
  Main Program
 *----------------------------------------------------------------*/
int main (void)
{
  LED_init();                           /* LED Initialization    */
  BUTTON_init();                        /* BUTTON Initialization */
  init_timer(0,0x017D7840);             /* TIMER0 Initialization */
  enable_timer(0);

  LPC_SC->PCON |= 0x1;     /* power-down mode */
  LPC_SC->PCON &= 0xFFFFFFFD;

  __ASM("wfi");
  /* program continues */

}
```

Configuring MCR

1. init_timer()
2. GUI
3. configuration wizard.

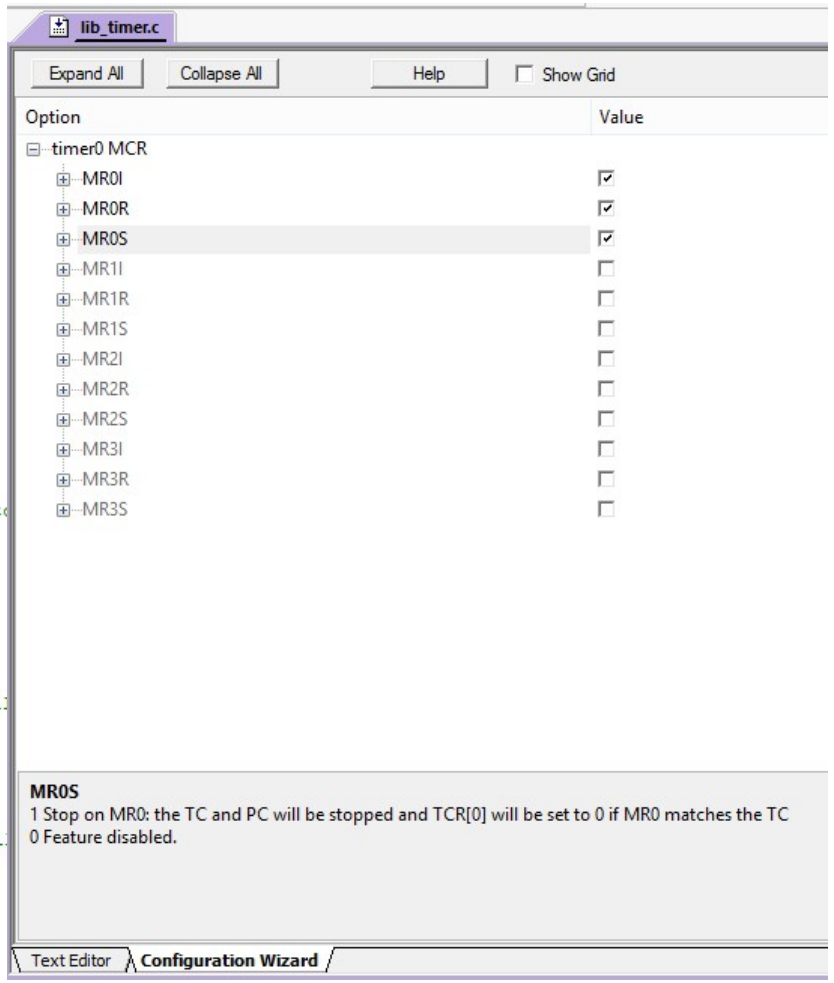
```
/*----------------------------------------
  Main Program
 *----------------------------------------
int main (void)
{
  LED_init();                          /* LE
  BUTTON_init();                       /* BU
  init_timer(0,0x017D7840);            /* TI
  enable_timer(0);

  LPC_SC->PCON |= 0x1;     /* power-down mode
  LPC_SC->PCON &= 0xFFFFFFFD;

  __ASM("wfi");
  /* program continues */

}
```



lib_timer.c

- Text editor and configuration wizard

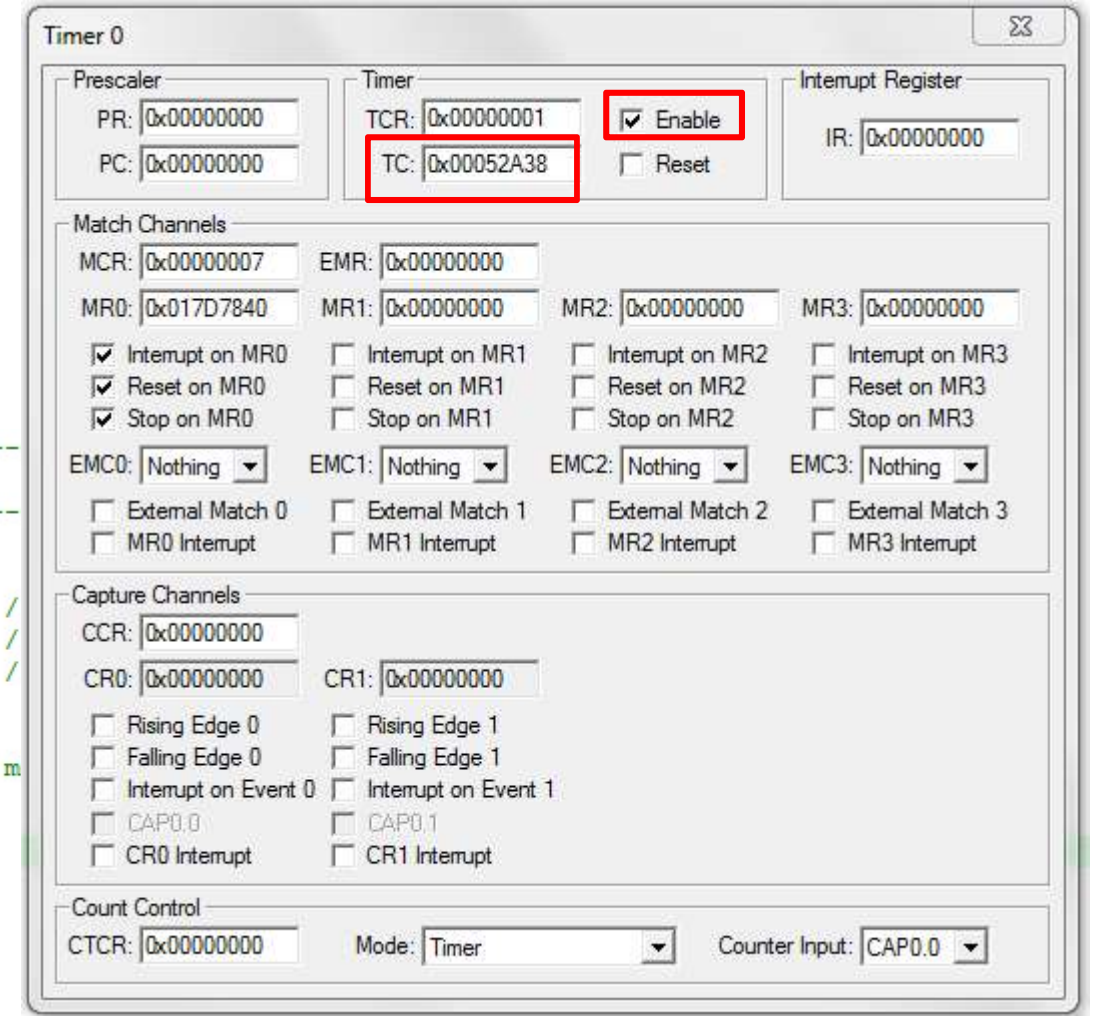
```
91  //*** <<< Use Configuration Wizard in Context Menu >>> ***
92  // <h> timer0 MCR
93  //   <e.0> MR0I
94  //   <i> 1 Interrupt on MR0: an interrupt is generated when MR0 mat
95  //   <i> 0 This interrupt is disabled
96  //   </e>
97  //   <e.1> MR0R
98  //   <i> 1 Reset on MR0: the TC will be reset if MR0 matches it.
99  //   <i> 0 Feature disabled.
100 //   </e>
101 //   <e.2> MR0S
102 //   <i> 1 Stop on MR0: the TC and PC will be stopped and TCR[0] wi
103 //   <i> 0 Feature disabled.
135 //   <i> 0 Feature disabled.
136 //   </e>
137 //   <e.11> MR3S
138 //   <i> 1 Stop on MR3: the TC and PC will be stopped and TCR[3] wi
139 //   <i> 0 Feature disabled.
140 //   </e>
141   LPC_TIM0->MCR = 7;
142 // </h>
143 //*** <<< end of configuration section >>>    ***
```



Enable_timer function

- The timer is started.

```
/*----------------------------------------
  Main Program
 *----------------------------------------
int main (void)
{
  LED_init();                           /
  BUTTON_init();                        /
  init_timer(0,0x017D7840);             /
  enable_timer(0);

  LPC_SC->PCON |= 0x1;     /* power-down m
  LPC_SC->PCON &= 0xFFFFFFFD;

  __ASM("wfi");
  /* program continues */

}
```



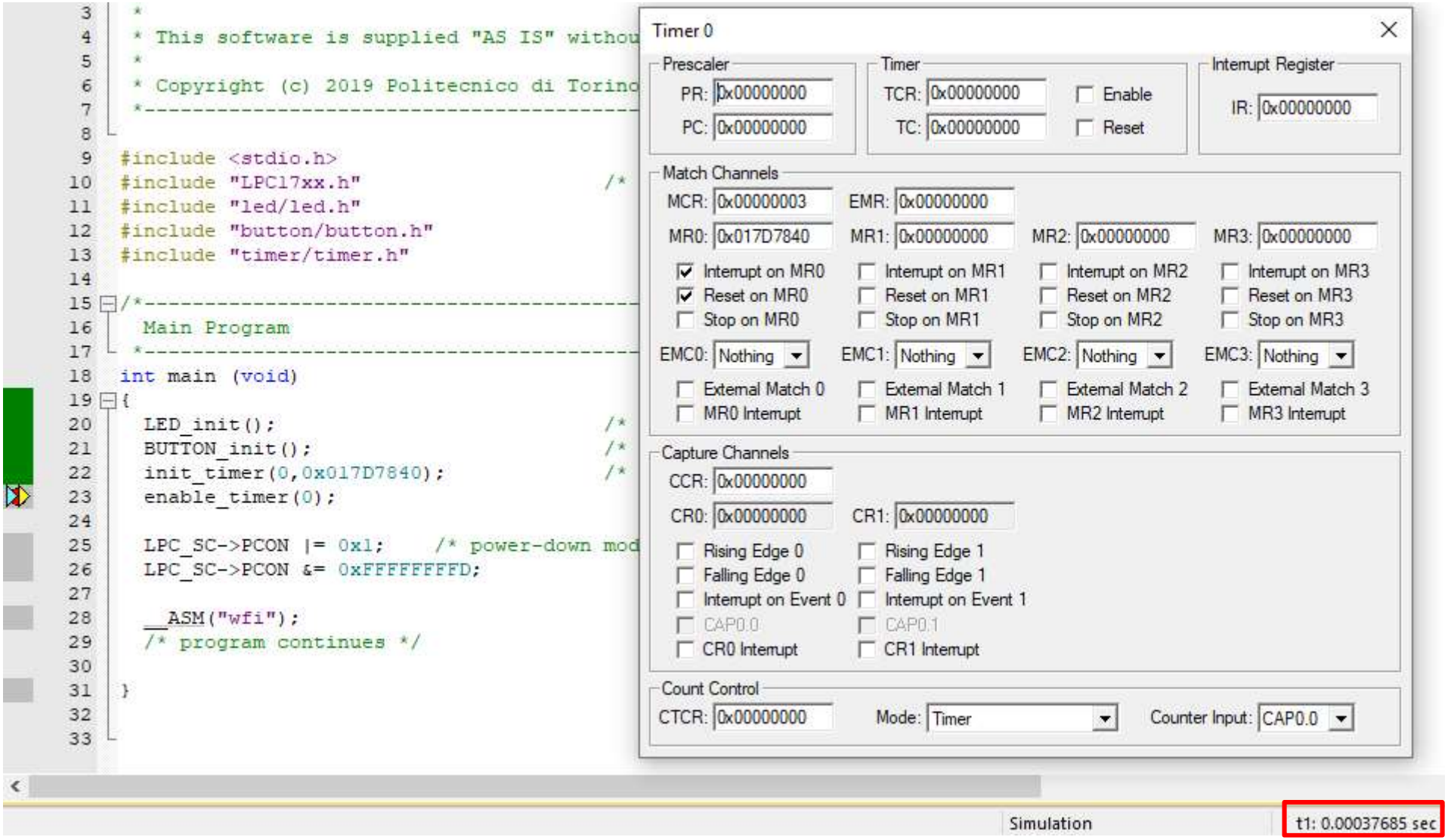
Timing check

- Measure time between breakpoints.

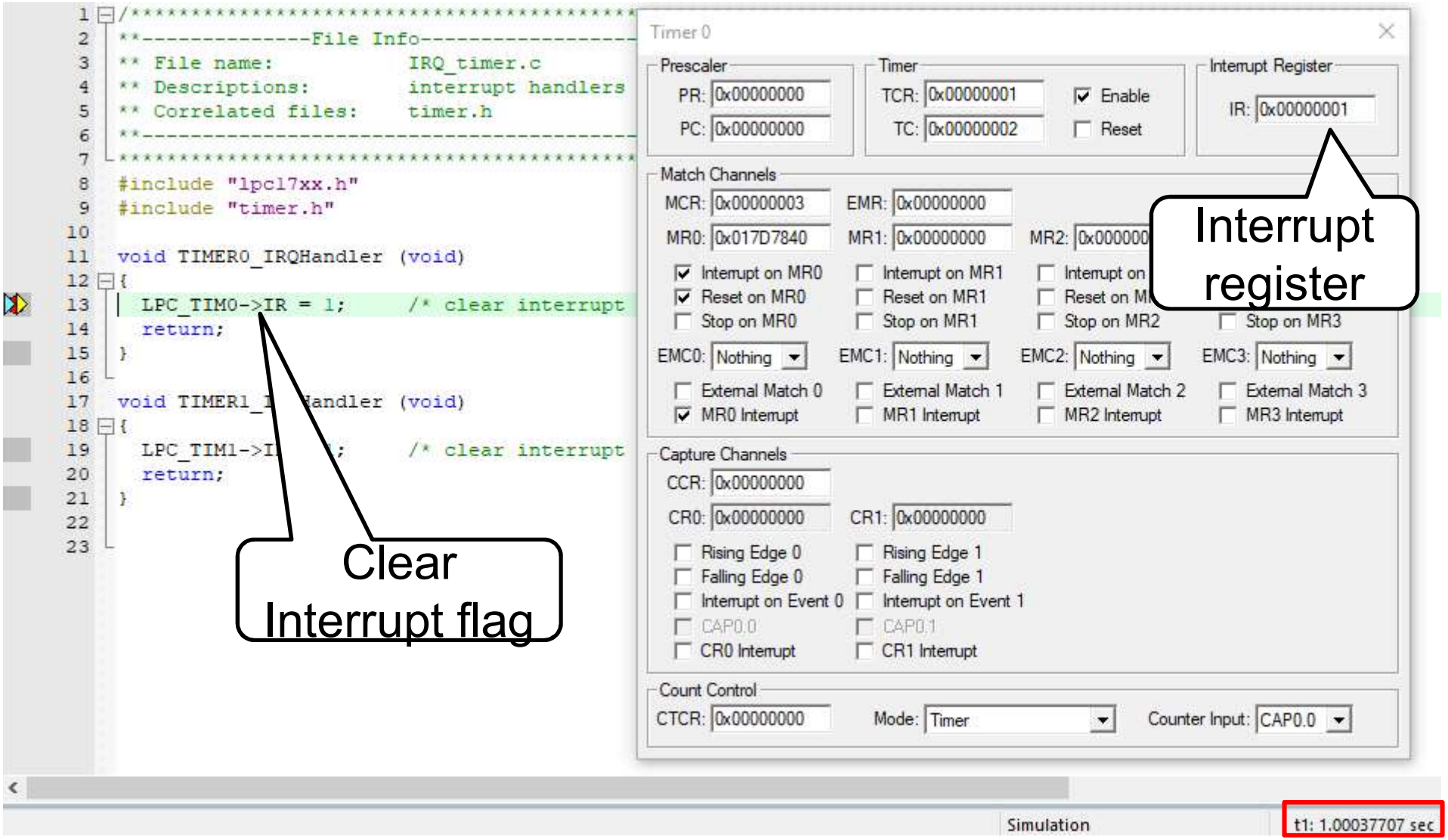
```
10
11  void TIMER0_IRQHandler (void)
12  {
13    LPC_TIM0->IR = 1;      /* clear interrupt flag */
14    return;
15  }
16
17  void TIMER1_IRQHandler (void)
18  {
19    LPC_TIM1->IR = 1;      /* clear interrupt flag */
20    return;
21  }
```

```
15  /*-------------------------
16    Main Program
17    *-------------------------------------------------------------------*/
18  int main (void)
19  {
20    LED_init();                          /* LED Initialization    */
21    BUTTON_init();                       /* BUTTON Initialization */
22    init_timer(0,0x017D7840);            /* TIMER0 Initialization */
23    enable_timer(0);
24
25    LPC_SC->PCON |= 0x1;     /* power-down mode */
26    LPC_SC->PCON &= 0xFFFFFFFD;
27
28    __ASM("wfi");
29    /* program continues */
30
31  }
```

Initial time



Final time


/***
**--------------File Info-----------------
** File name: IRQ_timer.c
** Descriptions: interrupt handlers
** Correlated files: timer.h
**---
***/
#include "lpc17xx.h"
#include "timer.h"
void TIMER0_IRQHandler (void)
{
LPC_TIM0->IR = 1; /* clear interrupt
return;
}
void TIMER1_I Handler (void)
{
LPC_TIM1->I ; /* clear interrupt
return;
}
Clear
Interrupt flag
Timer 0
Prescaler
PR: 0x00000000
PC: 0x00000000
Timer
TCR: 0x00000001
Enable
TC: 0x00000002
Reset
Interrupt Register
IR: 0x00000001
Interrupt
register
Match Channels
MCR: 0x00000003
EMR: 0x00000000
MR0: 0x017D7840
MR1: 0x00000000
MR2: 0x000000
Interrupt on MR0
Reset on MR0
Stop on MR0
Interrupt on MR1
Reset on MR1
Stop on MR1
Interrupt on
Reset on M
Stop on MR2
Stop on MR3
EMC0: Nothing
EMC1: Nothing
EMC2: Nothing
EMC3: Nothing
External Match 0
MR0 Interrupt
External Match 1
MR1 Interrupt
External Match 2
MR2 Interrupt
External Match 3
MR3 Interrupt
Capture Channels
CCR: 0x00000000
CR0: 0x00000000
CR1: 0x00000000
Rising Edge 0
Falling Edge 0
Interrupt on Event 0
CAP0.0
CR0 Interrupt
Rising Edge 1
Falling Edge 1
Interrupt on Event 1
CAP0.1
CR1 Interrupt
Count Control
CTCR: 0x00000000
Mode: Timer
Counter Input: CAP0.0
Simulation
t1: 1.00037707 sec

Exercises

- Setup regular interval time interruptions; in the meanwhile the system enters in a low-power state.
- Raise another interrupt with a different period (using another match register).
- Figure out the maximum delay that can be obtained when using 25 MHz. Try to overcome this limitation using
  - HW resources available in the peripheral
  - SW methods.