STM32 Timers

G V V Sharma*

CONTENTS

***Abstract*—This manual shows how to program timers in arm using STM32F103C8T6.**

1 COMPONENTS

Component	Value	Quantity
Breadboard		1
Resistor	220 Ω	1
STM32F103C8T6		1
Seven Segment Display	Common Anode	1
Jumper Wires		20

TABLE 1.0: Components

Problem 1.1. List all available clocks in the STM32F103C8T6 blue pill.

Solution: See Table 1.1.

Problem 1.2. List all the available timers in the STM32F103C8T6 blue pill.

Solution: See Table 1.2

*The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in.

Clock	Location	Type	Frequency
HSI	Internal	RC	8Mhz
LSI	Internal	RC	32.768 kHz
HSE	External	Crystal	8Mhz

TABLE 1.1: STM32F103C8T6 Clock Types

Timer	Type	Counter Resolution
Systick	Default	24 bit
Independent	Watchdog	12 bit
Window	Watchdog	7 bit
TIM1	Advanced	16 bit
TIM2	General Purpose	
TIM3		
TIM4		

TABLE 1.2: STM32F103C8T6 Timer Types.

2 SYSTICK TIMER

The Systick timer is the default timer available on all ARM chips.

Problem 2.1. Make connections as shown in Table 2.1.

STM32	Seven Segment Display
3.3V	COM (through resistor)
PA1	DOT

TABLE 2.1: Pin Connections

Problem 2.2. Execute the program in

https://github.com/gadepall/STM32F103C8T6/blob/master/examples/timer/blink_systick.c

Problem 2.3. The default clock is the HSI 8MHz RC. Find the number of clock cycles required for a 1 s delay.

Solution: The time period is

$$T = \frac{1}{8}\mu s = 1 \text{ cycle} \tag{2.3.1}$$

Thus, the number of cycles required for 1 s delay is

$$1 \text{ second} = 8000000 \text{ cycles} \quad (2.3.2)$$

Problem 2.4. List the SysTick registers.

Solution: See Table 2.4.

Register	Command	Purpose
SysTick Control and Status	SysTick->CTRL	Timer control
SysTick Reload Value	SysTick->LOAD	Timer Count
SysTick Current Value	SysTick->VAL	Timer Initialize
SysTick Calibration Value		

TABLE 2.4: Systick Registers

Problem 2.5. What do the following instructions do?

```
SysTick->LOAD = 4000000;
SysTick->VAL = 0;
```

Solution: See Table 2.4 for details. These two instructions ask the SysTick timer to count down from 4000000 to 0.

Problem 2.6. Explain the following instruction.

```
while(!(SysTick->CTRL & 0x00010000));
```

Solution: Fig. 2.6 shows the SysTick CTRL register. 0x00010000 is used in the above command to mask all the bits except for bit 16, which is the COUNTFLAG. The **while** loop will stop once COUNTFLAG = 0. The while loop is used for the delay.

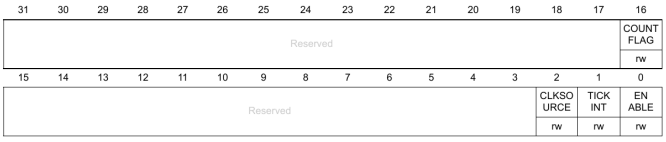

Fig. 2.6: Control Register (CTRL)

Problem 2.7. What does the following instruciton do?

```
SysTick->CTRL = 0x00000005;      // 8MHz clock
```

Solution: From Fig. 2.6, ENABLE = 1 enables the counter (for delay) and CLKSOURCE = 1 enables the 8 MHz internal RC clock.

Problem 2.8. Obtain a 1 MHz clock.

Solution: CLKSOURCE = 1 results in the $\frac{\text{Processor Clock}}{8}$ = 1 MHz clock.

```
SysTick->CTRL = 0x00000001;      // 1MHz clock
```

Problem 2.9. Obtain a delay of 1 second using the 1 MHz clock.

3 TIM1

Problem 3.1. Make the connections according to Table 2.1. Execute the following program

```
https://github.com/gadepall/STM32F103C8T6/blob/master/examples/timer/timer1_blink.c
```

Problem 3.2. Enable Timer1 through RCC.

Solution:

```
RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;
```

Problem 3.3. Select the HSI clock of 8 MHz as TIM1 clock.

Solution: See Fig. 3.3 for register. SMS=000 implies that the TIM1 will be controlled by the internal clock.

```
TIM1->SMCR = 0;
```

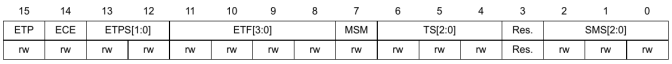

Fig. 3.3: Slave Mode Control Register (SMCR)

Problem 3.4. What does the following instruction do?

```
TIM1->CR1 = 0x0001;
```

Solution: Fig. 3.4 shows the control register 1 (CR1). CEN=1 enables the counter.

Fig. 3.4: Control Register 1 (CR1)

Problem 3.5. Make TIM1 clock = 2 KHz.

Solution: Through the following command,

```
TIM1->PSC          = 3999;
```

$$TIM1\_CLK = \frac{HSI\_CLK}{TIM1->PSC+1} = \frac{8000000}{4000} \quad (3.5.1)$$

Fig. 3.5 shows the TIM1− >PSC (prescalar) register.

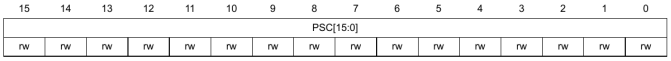

Fig. 3.5: Prescalar (PSC)

Problem 3.6. What is the maximum value that can be stored in TIM1− >PSC?

Problem 3.7. Make TIM1 count 1000 cycles of the 2 KHz TIM1 clock.

Solution:

```
TIM1->ARR          = 999;
```

Like the PSC, the ARR (auto reload register) is also of length 16 bits and used for factoring the clock.

Problem 3.8. What do the following instructions do?

```
if(TIM1->SR & 0x0001)//check if
   ARR count complete
{
        TIM1->SR &= ~0x0001;//
           clear status register SR
        GPIOA->ODR ^= (1 << 1);//
           blink LED through PA1
}
```

Solution: Once the TIM1 counter counts from 0 to TIM1− >ARR=999, it resets and starts counting again to 999. At the time of reset, the LSB of TIM1− >SR = 1. The **if** command checks this and when this condition is satisfied, TIM1− >SR is cleared and PA1 is toggled. This process keeps repeating. This results in a PA1 output of 1 and 0 with frequency

$$\frac{HSI\_CLK}{(TIM1->PSC+1)(TIM1->ARR+1)} = \frac{8000000}{4000\times 1000} = 2 \text{ Hz} \quad (3.8.1)$$

Problem 3.9. How would you use the repetition counter (RCR) to do the above?

Solution: The following instructions

```
        TIM1->SMCR  = 0;          //
           Internal clock, 8MHz
        TIM1->PSC          = 999; //
           Prescalar, dividing
           clock by 1000
        TIM1->CR1          = 0x0001;
                  //enable Timer1
        TIM1->ARR          = 999; //
           Load Count
        TIM1->RCR          = 3;   //
           Load Repetition Count
```

lead to

$$\frac{HSI\_CLK}{(TIM1->PSC+1)(TIM1->ARR+1)(TIM1->RCR+1)} = \frac{8000000}{4000\times 1000} = 2 \text{ Hz} \quad (3.9.1)$$

TIM1− >RCR keeps track of the number of times the counter has overflowed. Fig. 3.9 shows the repetition counter register.

Fig. 3.9: Repition Counter (RCR)

Problem 3.10. What is the maximum value of TIM1− >RCR?

Problem 3.11. What is the function of TIM1− >SR?

Solution: The status register (SR) is shown in Fig. 3.11. The UIF flag is 1 if the RCR overflows.

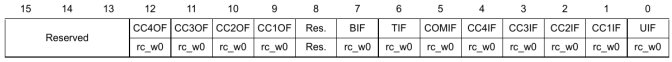

Fig. 3.11: Staus Register (SR)

4 TIM2

Problem 4.1. Blink an LED with TIM2.

Solution:

```
https://github.com/gadepall/
   STM32F103C8T6/blob/master/
   examples/timer/timer2_blink.c
```

Problem 4.2. In the above code,

```
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;
```

is used for enabling TIM2. Note that for TIM1, APB2 was used instead of APB1 in Problem 3.2. Explain.

Solution: Advanced Peripheral Bus 1 (APB1) and Advanced Peripheral Bus 2 (APB2) are connected to the Direct Memory Access (DMA) module, SRAM, peripherals like GPIOs, ADC, timers, etc. and Cortex core. APB1 has a maximum operating frequency of 36MHz while the maximum operating frequency of APB2 is 72MHz. That is why GPIOs are connected to APB2 bus instead of APB1 or any other bus and STM32 GPIOs can achieve 50MHz switching speed. APB1 bus mostly serves communication and timer modules of the STM32.

Problem 4.3. Mention any major difference between TIM1 and TIM2.

Solution: TIM1 is an advanced timer while TIM2 is a general purpose timer. One major difference between the two is that TIM2 does not have RCR.

5 Master-Slave Configuration

5.1 Blink

Problem 5.1. Execute the following program.

```
https://github.com/gadepall/
    STM32F103C8T6/blob/master/
    examples/timer/
    master1_slave2_blink.c
```

Problem 5.2. List the instructions for setting up TIM1 as master and TIM2 as slave. TIM1 should be a prescalar for TIM2.

Solution: The MMS bits can be seen in the CR2 register is shown in Fig. 5.2. The TS and SMS bits are visible in the SMCR register in Fig. 3.3.

```
TIM1->CR2  = 0x0020; //MMS =
    010
TIM2->SMCR = 0x0007; //TS =
    000, SMS = 111
```

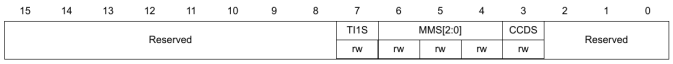

Fig. 5.2: Control Register 2 (CR2)

5.2 Decade Counter

Problem 5.3. Execute the following program.

```
https://github.com/gadepall/
    STM32F103C8T6/blob/master/
    examples/timer/decade_counter.c
```

Problem 5.4. If TIM2->ARR = 9, TIM2->CNT = ?

Solution: TIM2->CNT = 0, 1, ..., 9, 0,..., 9,... The counting rate depends on the PCR, ARR, RCR registers of TIM1 and the PCR and ARR registers of TIM2.