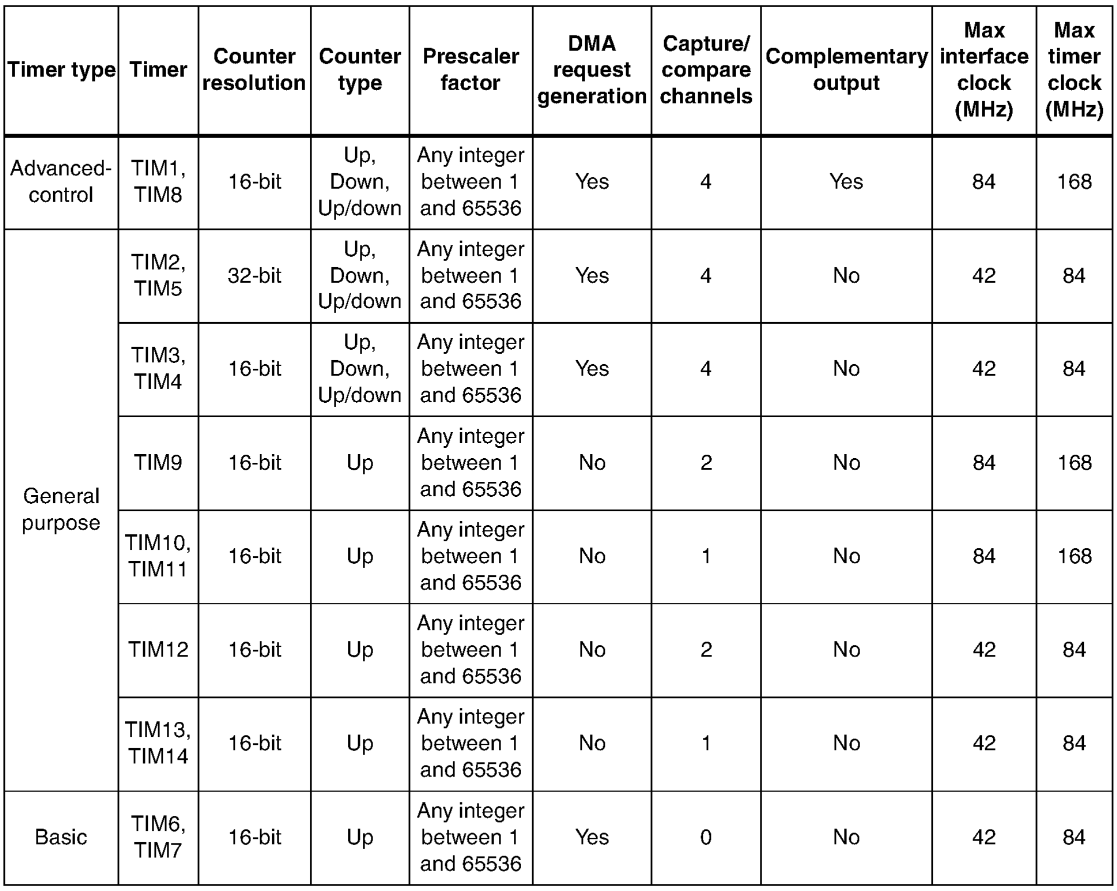
9. Counting pulses by Timer2

There are 14 timers included on the microcontroller chip. Their key properties are listed in Fig. 1 (DM00037051.pdf, p. 29). Detailed properties of each group of timers can be seen in RM0090. We will use Timer2 in this experiment to count pulses applied to suitable pin on the microcontroller and show current content of the counter on the LCD. There will be three pushbuttons used to initialize the content of the counter, to start counting, and to stop counting. Timer2 (RM0090, chapter 14) is selected for this example thanks to its input pin which is available at the edge of the board.



Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)
Advanced-control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	84	168
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	84	168
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	84	168
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	42	84
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	42	84
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	42	84

Figure 1: The timers included in STM32F407 and their key properties

Timers TIM1 and TIM8 use 16-bit counters and are the most complex timers of all timers included in the microcontroller. Timers TIM2 and TIM5 are 32-bit versions of TIM1/TIM8, but have less hardware included and therefore less options. Timers TIM3 and TIM4 are 16-bit versions of TIM2/TIM5. Timers TIM9, TIM10 and on are stripped-down versions of timer TIM4, and so on.

All timers are based on the block diagram given in RM0090, Fig. 65, page 294. We are going to use timer TIM2 for the experiment, so a simplified block diagram for timer TIM2 is given in Fig. 2 (reference manual RM0090, Figure 119, page 419).

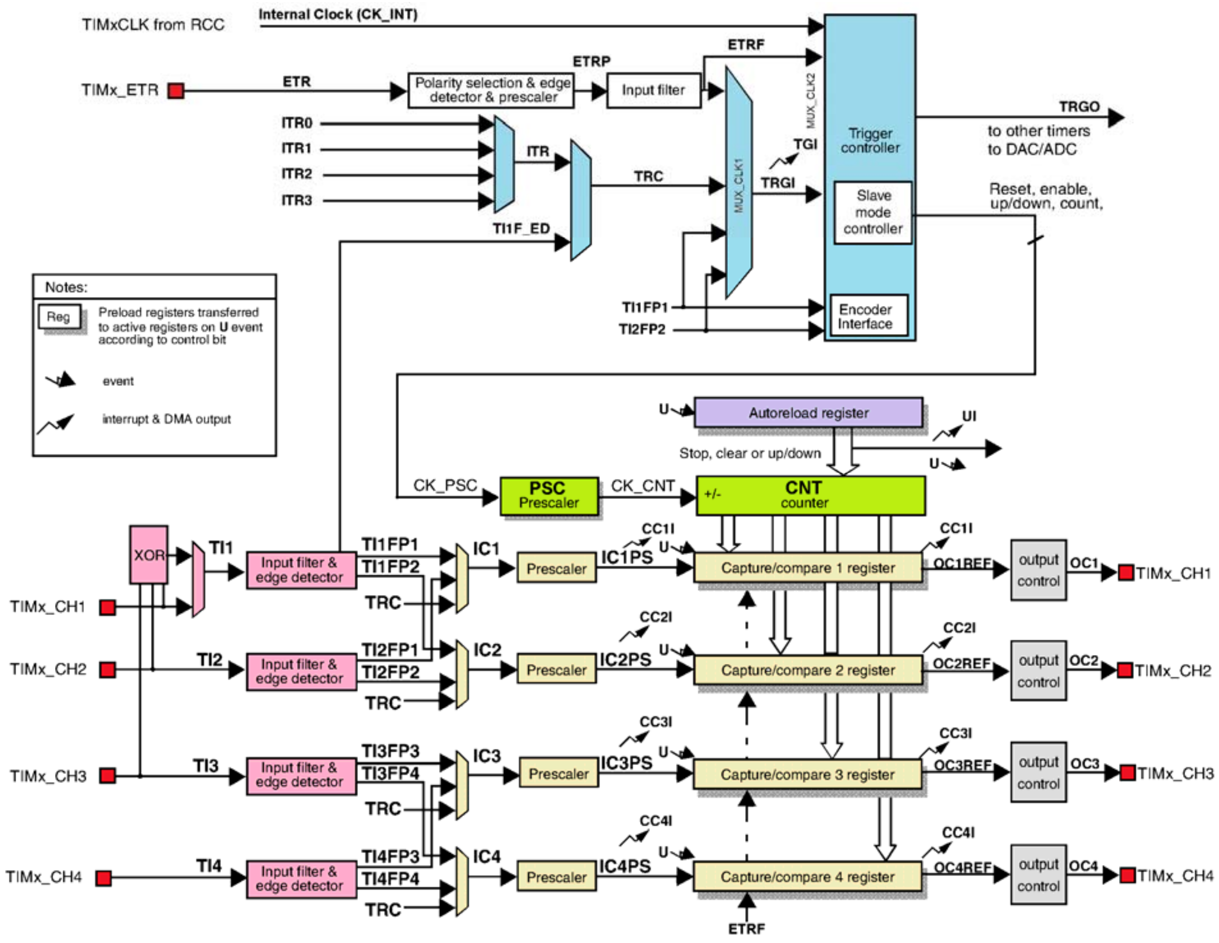


Figure 2: The block diagram of timer TIM2 & TIM5. Counter: green boxes, clock select: blue boxes, input signals conditioning: purple boxes, output control: grey boxes.

The content of the counter CNT (green) is available in register TIM2_CNT, and can be read from here to be displayed on the LCD screen. The content is 32 bits in width. The pre-scaler and auto-reload registers are available, but are not used for this experiment.

The counter needs a clock signal (CK_PSC) to count. The clock signal comes from a set of modules (blue boxes) from the top of Fig. 2. Several signals are available for counting, and multiplexers (two of them for this example) select the appropriate clock signal. We will use the timer input TIM2_CH2, as this line can be mapped to port A, bit 1 (PA01) inside of the microcontroller. The buffered version of line PA01 is available at the edge of the test board. A signal must be routed from port A, bit 1 to CK_PSC.

- One multiplexer is hidden in the rightmost blue module; its name shall be MUX_CLK2, and is controlled by three bits called SMS (Slave Mode Selection) in register TIM2_SMCR (Slave Mode Control Register). Putting all three SMS bits to one selects “External clock Mode 1”, which is equivalent to selecting the line TRGI (TRiGger Input) for clocking the counter, see RM0090, page 458 for detailed description.

- One multiplexer is used to select one of possible sources for trigger signal, and is controlled by three bits called TS, also within the register TIM2_SMCR. Setting these bits to '110' selects the line TI2FP2 (Timer Input 2, Filtered, Polarity selected line 2). This signal is available as a result of conditioning of the signal at the input TIM2_CH2 at the output from a pink box.

The clock signal can be conditioned in a module enclosed in a pink box before the use. Here signal coming from input line TIM2_CH2 can be filtered and the correct edge (rising or falling) to cause the counting can be selected. Both settings are available in the upper half of the register TIM2_CCMR1 (Capture Compare Mode 1, this register is responsible for inputs/outputs TIM2_CH1 and TIM2_CH2, there is also a register TIM2_CCMR2 for other two inputs/outputs); corresponding bits are called CCS2 (Capture/Compare Selection 2) and IC2F (Input Channel 2 Filtering), details are given in RM0090, page 466.

The next thing to do is to map the timer input line TIM2_CH2 to port PA01. To do it we have to go to registers responsible for port A, and change the one that affects so called "alternate functions". There are 16 possible alternate functions available for each bit of a port (see Figure 14, page 190, RM0090), therefore 4 bits are required to select one of 16 possibilities. There are 16 bits in a port, therefore 16 x 4 = 64 bits are needed to select alternative functions for a complete port. These 64 bits fit into two 32-bit registers named AFR[x], where x stands for either 0 or 1. Four bits for port bit PA00 go into lower four bits of a register AFR[0], four bits for PA01 go into next four bits of register AFR[0], and so on for other bits of a port (RM0090, page 202 and 203). Alternative functions for timers TIM1 and TIM2 are selected by '0001', and since we are changing the alternative function for PA01, the register AFR[0] must become 0x00000010.

Once the correct alternative function is selected, it has to be enabled by changing the mode setting for the bit in a port. A bit can be defined as input (00), output (01), alternative function (10), or analog (11). The bits to define the mode are stored in a register called GPIOA_MODER, two bits per bit of a port. Since we are changing the mode setting for PA1, the register GPIOA_MODER should become 0x00000008 (0000 0000 0000 0000 0000 0000 0000 1000 binary).

The last thing to set is to enable the counting; this is accomplished by setting the bit CEN (Counter Enable, least significant bit) located in register TIM2_CR1.

Figure 3 gives the listing of the demo program. The program starts with two include statements to define the standard names used in program and to include LCD functions. The initialization of peripherals starts with turning on the clock for ports A (here the input signal is connected), E (here the pushbuttons are connected), and timer TIM2. Liquid crystal display is initialized. Next two lines activate the alternate mode at port A, bit 1, and select the alternate function as TIM1/TIM2. A message is written at LCD.

The timer TIM2 gets initialized next. There are three steps:

- Ch.2 is defined as input channel, mapped to TI2 by writing '01' to bits 9..8 of register TIM2_CCMR1,
- External Clock Mode is selected by writing '111' to lower three bits of register TIM2_SMCR,
- Filtered Timer Input is selected by writing '110' to bits 6..4 of register TIM2_SMCR.

The counting is enabled by writing 1 to bit 0 of register TIM2_CR1..

Within the infinite loop the content of the counter is transferred to the LCD, then some time is wasted. Nest pushbuttons are checked, and actions are taken on demand. The actions are:

- Reset the content of the counter to zero.
- Enable counting
- Disable counting

```
#include "stm32f4xx.h"
#include "LCD2x16.c"

void main (void){

  RCC->AHB1ENR  |= 0x01 + 0x10; // Clock for PortA, E
  RCC->APB1ENR  |= 0x01;        // Clock for Timer2
  LCD_init();                   // init LCD

  GPIOA->MODER  |= 0x00000008;    // all inputs but: PA1 => AF mode
  GPIOA->AFR[0] |= 0x00000010;    // select AF1 (TIM2) for PA01 -> TIM2_CH2

  LCD_string("Timer2:", 0x00);          // display title string

  TIM2->CCMR1   |= 0x0100;        // Ch. 2 as TI2
  TIM2->SMCR    |= 0x0007;        // Ext. clk mode 1
  TIM2->SMCR    |= 0x0060;        // TI2FP2 as ext. clock
  TIM2->CR1     |= 0x0001;        // enable counting

  while (1) {
    LCD_uInt32(TIM2->CNT,0x40,1);       // write position to LCD
    for (int i = 0; i<100000; i++) {};     // waste some time
    if (GPIOE->IDR & 0x01)      TIM2->CNT = 0;          // reset counter
    if (GPIOE->IDR & 0x02)      TIM2->CR1 |=  0x01;     // enable counter
    if (GPIOE->IDR & 0x04)      TIM2->CR1 &= ~0x01;     // disable counter
  };
}
```

Figure 3: A listing of the program to count incoming pulses using timer TIM2. The input signal is applied to PA1 (AIN1), the result is displayed on the LCD.