**Lab 1: Pulse Width Modulation and Motor Control**

**Instructor: Prof. Yifeng Zhu**

**Fall 2019**

## **Goals**

1. Understand the clock tree of STM32F4 micro-controllers
2. Understand the concept of Pulse Width Modulation (PWM)
3. Learn how to configure and start a timer
4. Use PWM to control the motors

## **Pre-Lab Assignment**

1. Read Textbook Chapter 15.3 PWM Output
2. Watch Youtube Tutorial
   * Timer PWM output <https://youtu.be/zkrVHIcLGww> (17 minutes)
3. Complete the pin and timer configuration tables

## **Lab Demo**

1. Initialize TIM4 to produce a PWM output:
   1. Initialize GPIOB pin 6 to be configured as push-pull, NO PUPD, and low speed, in AF mode, using AF2. (The timer 4 PWM alternate function)
   2. Configure timer 4 as done in the pre-lab assignment
2. Produce and measure a 500Hz output PWM signal on TIM4 Ch1
   1. Modify the main function as described
   2. Measure the output of TIM4 Ch1 using an oscilloscope to test said code

## **Post-Lab Assignment**

1. Complete the post lab report and write your answer in *readme.md*

# **STM32F4 Microcontroller**

The [STEVAL-FCU001V1](https://www.st.com/en/evaluation-tools/steval-fcu001v1.html) drone controller board uses an ARM Cortex-M4 (with DSP and FPU) microcontroller ([STM32F401CCU6TR](https://www.st.com/en/microcontrollers-microprocessors/stm32f401cc.html), UFQFPN48). The core runs at a frequency of up to 84 MHz.



Figure 1. ST drone kit



Figure 2. The STEVAL-FCU001V1 Drone controller

The key references for this microcontroller are listed below.

* RM0368 Reference manual STM32F401xB/C and STM32F401xD/E advanced Arm®-based 32-bit MCUs ([link](https://www.st.com/resource/en/reference_manual/dm00096844.pdf))
* STM32F401xB STM32F401xC Datasheet ([link](https://www.st.com/resource/en/datasheet/stm32f401cc.pdf))
* STM32 Cortex®-M4 MCUs and MPUs programming manual ([link](https://www.st.com/resource/en/programming_manual/dm00046982.pdf))

# **Clock Configuration**

There are two major types of clocks: **system clock** and **peripheral clock**.

* ***System Clock.*** To provide different tradeoffs between performance and energy-efficiency for different applications, the processor core can be driven by three different clock sources, including ***HSI*** (high-speed internal) oscillator clock, ***HSE*** (high-speed external) oscillator clock, and **PLL** clock. A faster clock provides better performance but usually consumes more power, which might not appropriate for battery-powered systems.
* ***Peripheral Clock.*** All peripherals require to be clocked to function. However, ***clocks of all peripherals are turned off by default to reduce power consumption***. Therefore, software has to enable the clock of the a peripheral or a GPIO port if a peripheral or a GPIO pin is used.

Figure 3 shows the clock tree of ***STM32F4xC***, the processor used in the drone controller board. The clock sources in the domain of Advanced High-performance Bus (***AHB***), low-speed Advanced Peripheral Bus 1 (***APB1***) and high-speed Advanced Peripheral Bus 2 (***APB2***) can be switched on or off independently when it is not used. Software can select various clock sources and scaling factors to achieve desired clock speed, depending on the application’s needs.

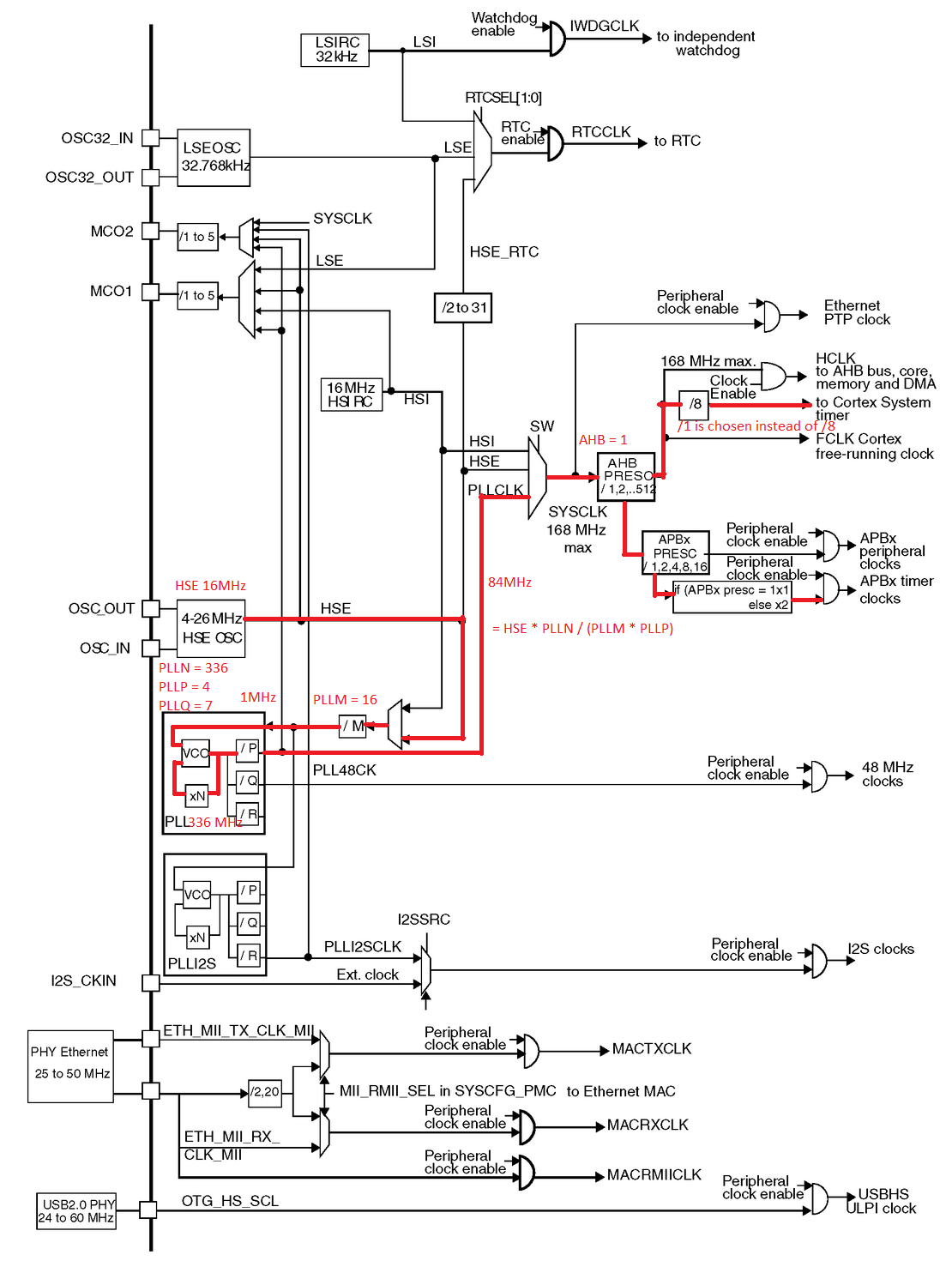
****

Figure 3. Clock tree of the microcontroller (STM32F401xC). The red lines are programmed by software shown later.

The software provided in this lab uses the 16MHz HSE as the input to the PLL clock. Appropriate scaling factors have been selected to achieve the clock speed (84 MHz). See the function void **System\_Clock\_Init**() for details.

|  |
| --- |
| void System\_Clock\_Init(void){  ...    // Enable the Internal High Speed oscillator (HSI)  RCC->CR |= RCC\_CR\_HSEON;  while((RCC->CR & RCC\_CR\_HSERDY) == 0);  RCC->CR &= ~RCC\_CR\_PLLON;  while((RCC->CR & RCC\_CR\_PLLRDY) == RCC\_CR\_PLLRDY);    // Select clock source to PLL  RCC->PLLCFGR &= ~RCC\_PLLCFGR\_PLLSRC;  RCC->PLLCFGR |= RCC\_PLLCFGR\_PLLSRC\_HSE; // 00 = No clock, 01 = MSI, 10 = HSI, 11 = HSE    // Make PLL as 84 MHz  // f(VCO clock) = f(PLL clock input) \* (PLLN / PLLM) = 16MHz \* 336/16 = 168 MHz  // f(PLL\_R) = f(VCO clock) / PLLR = 168MHz/2 = 84MHz  RCC->PLLCFGR = (RCC->PLLCFGR & ~RCC\_PLLCFGR\_PLLN) | 336U << 6;  RCC->PLLCFGR = (RCC->PLLCFGR & ~RCC\_PLLCFGR\_PLLM) | 16U << 0;  RCC->PLLCFGR = (RCC->PLLCFGR & ~RCC\_PLLCFGR\_PLLQ) | 7U << 24;  RCC->PLLCFGR &= ~RCC\_PLLCFGR\_PLLP; // 00: PLLP = 2, 01: PLLP = 4, 10: PLLP = 6, 11: PLLP = 8  RCC->CR |= RCC\_CR\_PLLON;  while((RCC->CR & RCC\_CR\_PLLRDY) == 0);    // Select PLL selected as system clock  RCC->CFGR &= ~RCC\_CFGR\_SW;  RCC->CFGR |= RCC\_CFGR\_SW\_PLL; // 00: MSI, 01:HSI, 10: HSE, 11: PLL    // Wait until System Clock has been selected  while ((RCC->CFGR & RCC\_CFGR\_SWS) != RCC\_CFGR\_SWS\_PLL);  } |

Example 1. Setting the clock

**Timer PWM Output**

Timers are special hardware components that provide accurate timestamps, time-interval measurements, and timer-related periodic events for both hardware and software. This lab presents one example use of timers: generating output waveforms (output compare and PWM).

The timing information are controlled by three key registers:

* Counter Register (TIMx\_CNT)
* Prescaler Register (TIMx\_PSC)
* Auto-Reload Register (TIMx\_ARR)

Read the following two chapters to find out how to generate PWM outputs.

* Chapter 15 General-purpose Timers of the Textbook
* Chapter 13 of the STM32F401xC reference manual

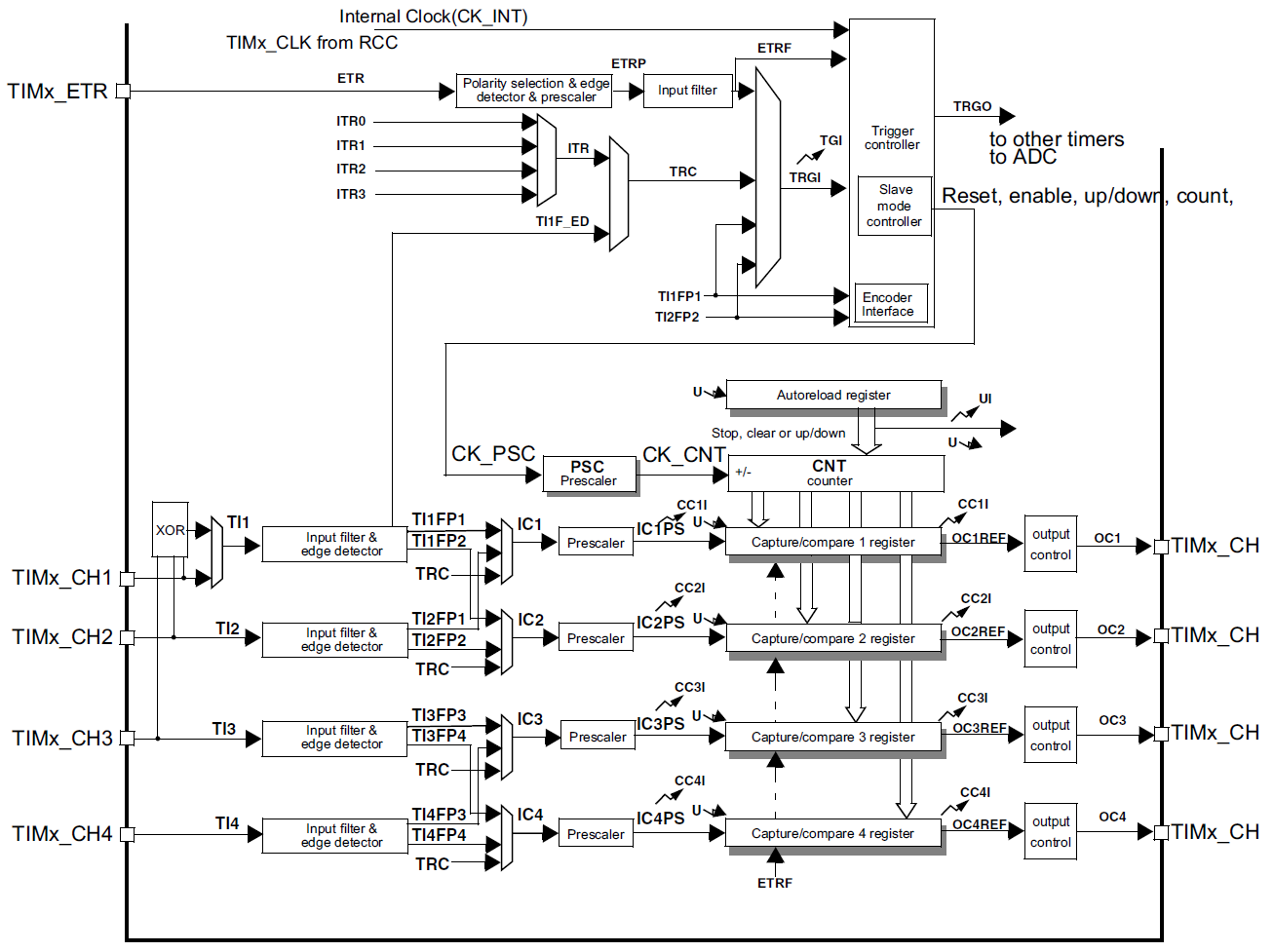


Figure 4. General-purpose timer block diagram (coped from the STM32F401xC reference manual).

**Lab 1:** **Pre-Lab Assignment**

Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Note: For both this assignment and future ones, please refer to [this document](https://www.st.com/content/ccc/resource/technical/document/reference_manual/5d/b1/ef/b2/a1/66/40/80/DM00096844.pdf/files/DM00096844.pdf/jcr:content/translations/en.DM00096844.pdf) for information regarding the mapping of register values to their associated settings.)

1. **Complete the following table to configure the PWM output for Channel 1 of Timer 4 with the following settings:**

Upcounting mode, edge-aligned mode, 1:1 clock division, an auto-reload value of 1999, a prescaler value of 84, update generation enabled, slave mode disabled, internal trigger 0 selected, TRGO reset mode, ETR noninverted, external clock mode 2 disabled, external trigger prescaler disabled, external trigger filter disabled, output compare PWM mode 1, CC1 channel configured as output, CC1 active high, OC1 preload register enabled, OC1 fast mode disabled, CC1 output enabled.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Offset** | **Register** | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | | **14** | | **13** | | **12** | | **11** | | **10** | | **9** | | **8** | | **7** | | **6** | | **5** | | **4** | | **3** | | **2** | | **1** | | **0** | |
| 0x00 | **TIMx\_CR1** | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CKD [1:0] | | | | ARPE | | CMS [1:0] | | | | DIR | | OPM | | URS | | UDIS | | CEN | | |
| Reset value |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |
| 0x04 | **TIMx\_CR2** | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TI1S | | MMS[2:0] | | | | | | CCDS | | Reserved | | | | | | |
| Reset value |  | |  | |  | |  | |  | |
| 0x08 | **TIMx\_SMCR** | Reserved | | | | | | | | | | | | | | | | | ETP | | ECE | | ETPS [1:0] | | | | ETF[3:0] | | | | | | | | MSM | | TS[2:0] | | | | | | Reserved | | SMS[2:0] | | | | | | |
| Reset value |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| 0x10 | **TIMx\_SR** | Reserved | | | | | | | | | | | | | | | | | | | | | | | CC4OF | | CC3OF | | CC2OF | | CC1OF | | Reserved | | | | TIF | | Reserved | | CC4IF | | CC3IF | | CC2IF | | CC1IF | | UIF | |
| Reset value |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| 0x14 | **TIMx\_EGR** | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TG | | Reserved | | CC4G | | CC3G | | CC2G | | CC1G | | UG | | |
| Reset value |  | |  | |  | |  | |  | |  | | |
| 0x18 | **TIMx\_CCMR1**  Output Compare mode | Reserved | | | | | | | | | | | | | | | | | OC2CE | | OC2M [2:0] | | | | | | OC2PE | | OC2FE | | CC2S [1:0] | | | | OC1CE | | OC1M [2:0] | | | | | | OC1PE | | OC1FE | | CC1S [1:0] | | | | |
| Reset value |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| **TIMx\_CCMR1**  Input Capture mode | Reserved | | | | | | | | | | | | | | | | | IC2F[3:0] | | | | | | | | IC2  PSC [1:0] | | | | CC2S [1:0] | | | | IC1F[3:0] | | | | | | | | IC1  PSC [1:0] | | | | CC1S [1:0] | | | | |
| Reset value |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| 0x20 | **TIMx\_CCER** | Reserved | | | | | | | | | | | | | | | | | CC4NP | | Reserved | | CC4P | | CC4E | | CC3NP | | Reserved | | CC3P | | CC3E | | CC2NP | | Reserved | | CC2P | | CC2E | | CC1NP | | Reserved | | CC1P | | CC1E | |
| Reset value |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
| 0x28 | **TIMx\_PSC** | Reserved | | | | | | | | | | | | | | | | | PSC[15:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset value |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |

1. **Calculating the timer registers**

What is the clock frequency used to drive the timer?

Clock source: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Select one: HSI/HSE)

Clock frequency: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ MHz

**Motor control**

Suppose the PWM signal period is 2 ms (500 Hz).

TIM4\_PSC = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TIM4\_ARR = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Calculate the value of CCR1:

**Lab 1: Pulse Width Modulation and Motor Control**

**Instructor: Prof. Yifeng Zhu**

**Fall 2019**

**Lab Demo**

Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Lab demo:**

1. Inside the indicated areas of MX\_TIM4\_Init() within main.c, initialize TIM4 to produce a PWM output:
   1. Initialize GPIOB pin 6 to be configured as push-pull, NO PUPD, and low speed, in AF mode, using AF2. (The timer 4 PWM alternate function)
   2. Configure timer 4 as done in the pre-lab assignment. Don’t forget to configure the ARR and PSC registers to produce the signal required for part 2.
2. Produce and measure a 500Hz output PWM signal on TIM4 Ch1
   1. Add code to the end of the main function in main.c that sets the CCR1 register so that the output PWM signal has a duty cycle of 25%. To do this, you will also need to enable the “rc\_enable\_motor” and “fly\_ready” flags.
   2. Measure the output of TIM4 Ch1 using an oscilloscope to test that your code works correctly.

**Lab 1: Pulse Width Modulation and Motor Control**

**Instructor: Prof. Yifeng Zhu**

**Fall 2019**

**Post-Lab Assignment**

Suppose the 16-MHz HSE (high-speed external clock) is selected as the input clock of timer 1. Answer the following questions and show your calculation process clearly. Write your answer in README.md and submit it to gitlab.

1. To generate 1 Hz square wave with a duty cycle of 50%, how should we set up the timer? Indicate your counting mode and show the value of ARR, CRR, and PSC registers.
2. What is the smallest PWM frequency that can be generated?