Sử dụng Timer mode Output Compare

I, Kiến thức cần chuẩn bị

1.Timer

Timer là một loại ngoại vi được tích hợp ở hầu hết các vi điều khiển, cung cấp cho người dùng nhiều ứng dụng như xác định chính xác một khoảng thời gian, đo – đếm xung đầu vào, điều khiển dạng sóng đầu ra, băm xung PWM.

STM32F411 có 8 bộ Timer, trong đó có 1 bộ Advanced – control timer (TIM1) thường được các bộ thư viện sử dụng để tạo bộ đếm thời gian chuẩn của hệ thống (như ngắt System Tick, hàm tạo Delay, TimeOut…), và 7 bộ General – purpose timer (TIM2 đến TIM5 và TIM9 đến TIM11).

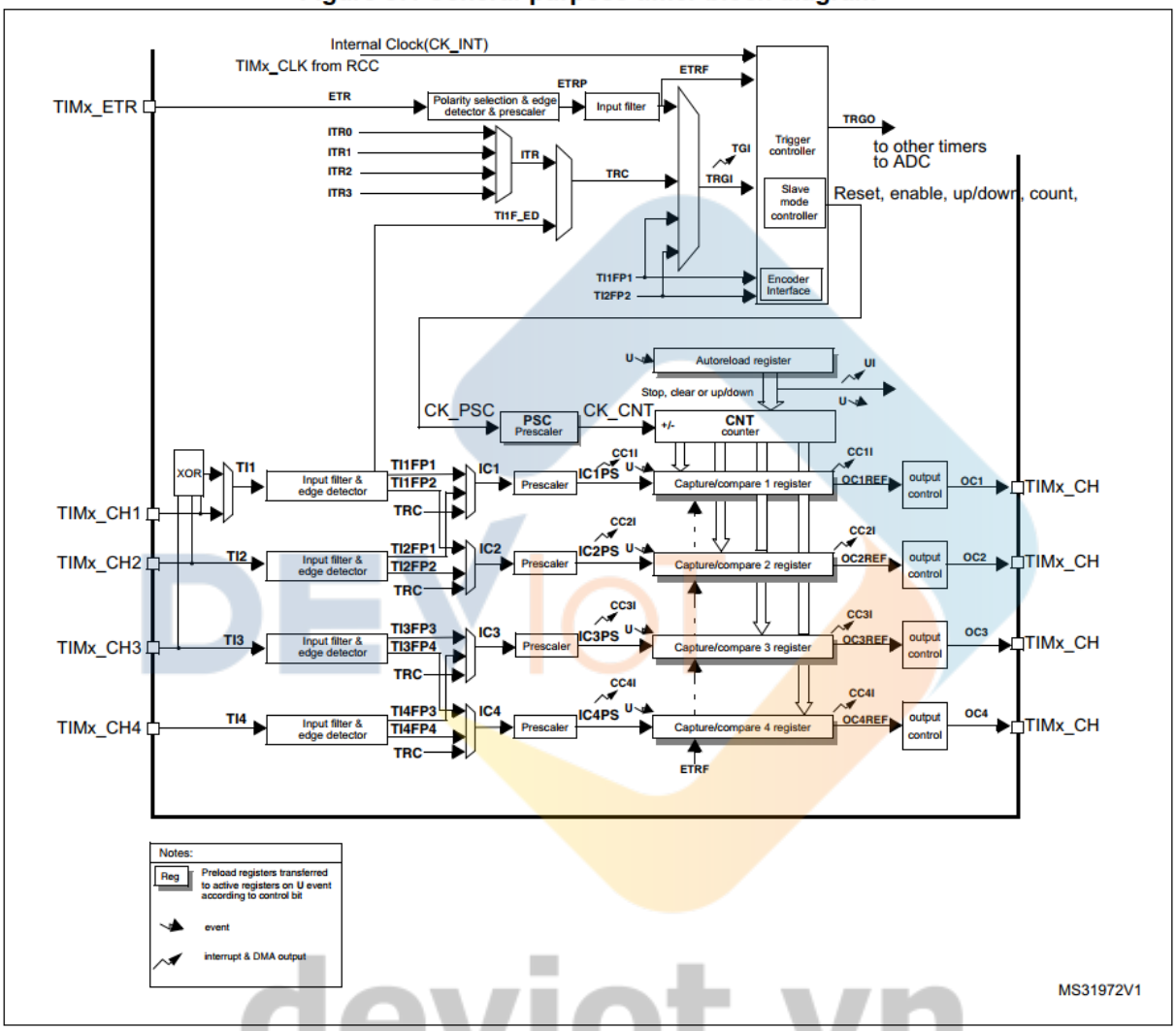
Table 4. Timer feature comparison

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complemen-tary output	Max. interface clock (MHz)	Max. timer clock (MHz)
Advanced -control	TIM1	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	100	100
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	100	100
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	100	100

2.Các kênh Compare

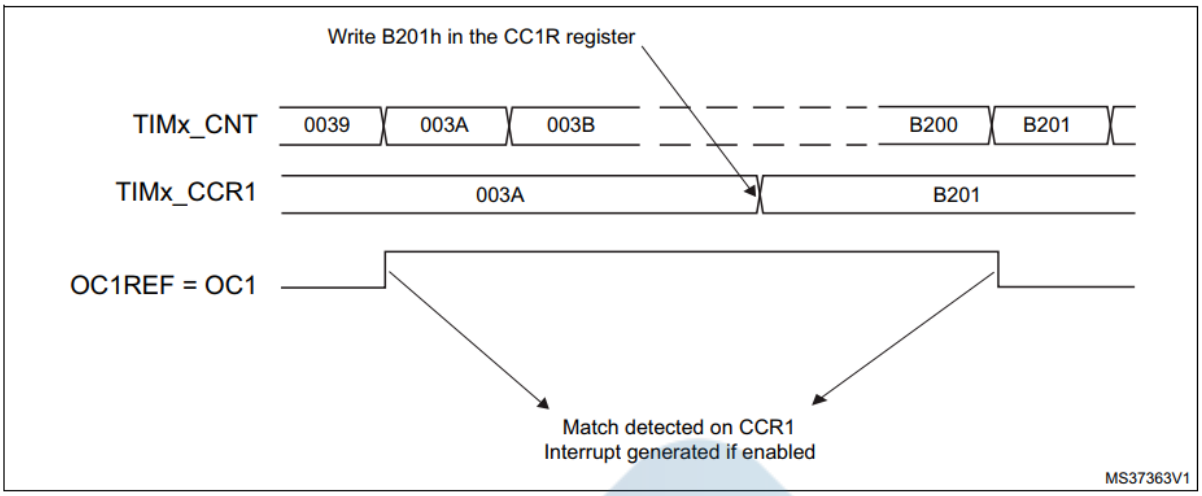
Output compare: Đây là một chức năng của bộ Timer với rất nhiều chế độ hoạt động được hỗ trợ. Cơ chế hoạt động chung là sẽ kiểm tra giá trị TIMxCNT đếm tới giá trị bằng với giá trị được nạp sẵn vào thanh ghi TIMx_CCRy thì một ngắt sẽ được tạo ra đồng thời có thể xuất xung đầu ra tại OCy output.

Figure 87. General-purpose timer block diagram



Lấy ví dụ ta chọn chế độ khi TIMx_CNT = TIMx_CCRy có ngắt xảy ra, chân OCy sẽ Toggle mức logic đầu ra.

Figure 116. Output compare mode, toggle on OC1



Còn lại chúng ta có tới 8 chế độ để sử dụng với bộ Output Compare của Timer.

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).

111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

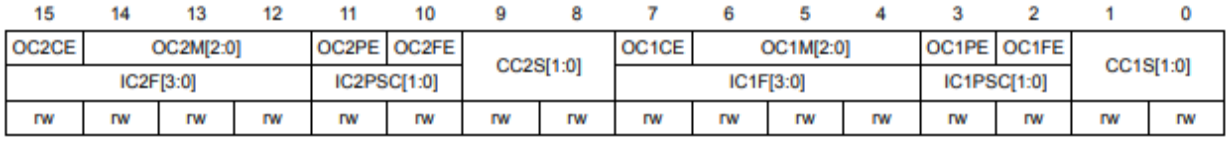
Vậy Output Compare và PWM khác nhau ở chỗ nào ?

Một số thanh ghi quan trọng cho các General Timer (TIM2 ,..,TIM5):

TIMx_CCMR1 (TIMx capture/compare mode register 1)

Address offset: 0x18

Reset value: 0x0000



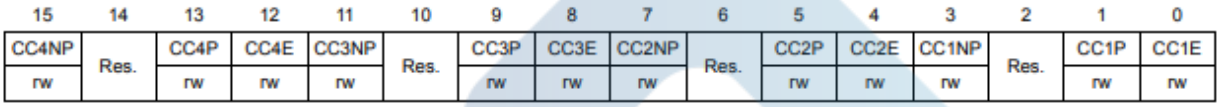
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Thanh ghi cấu hình chức năng hoạt động của Timer Mode Capture/Compare

TIMx_CCER (TIMx capture/compare enable register)

Address offset: 0x20

Reset value: 0x0000

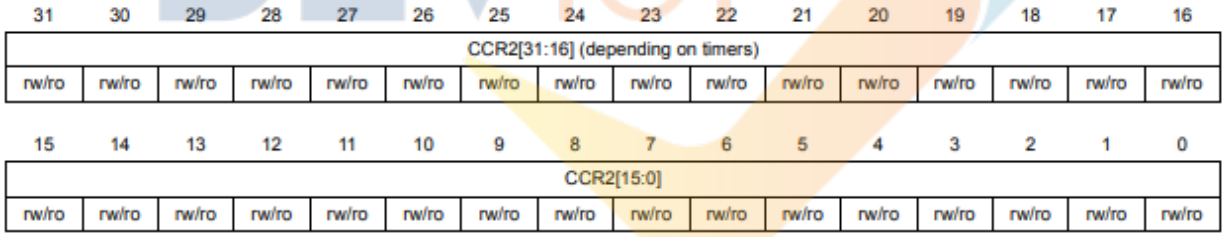


15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Thanh ghi enable chức năng Output Compare và cấu hình xung đầu ra của chân OCy.

TIMx_CCRy (TIMx capture/compare register , channel y với y từ 1 đến 4**)**

Reset value: 0x0000 0000



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16] (depending on timers)															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

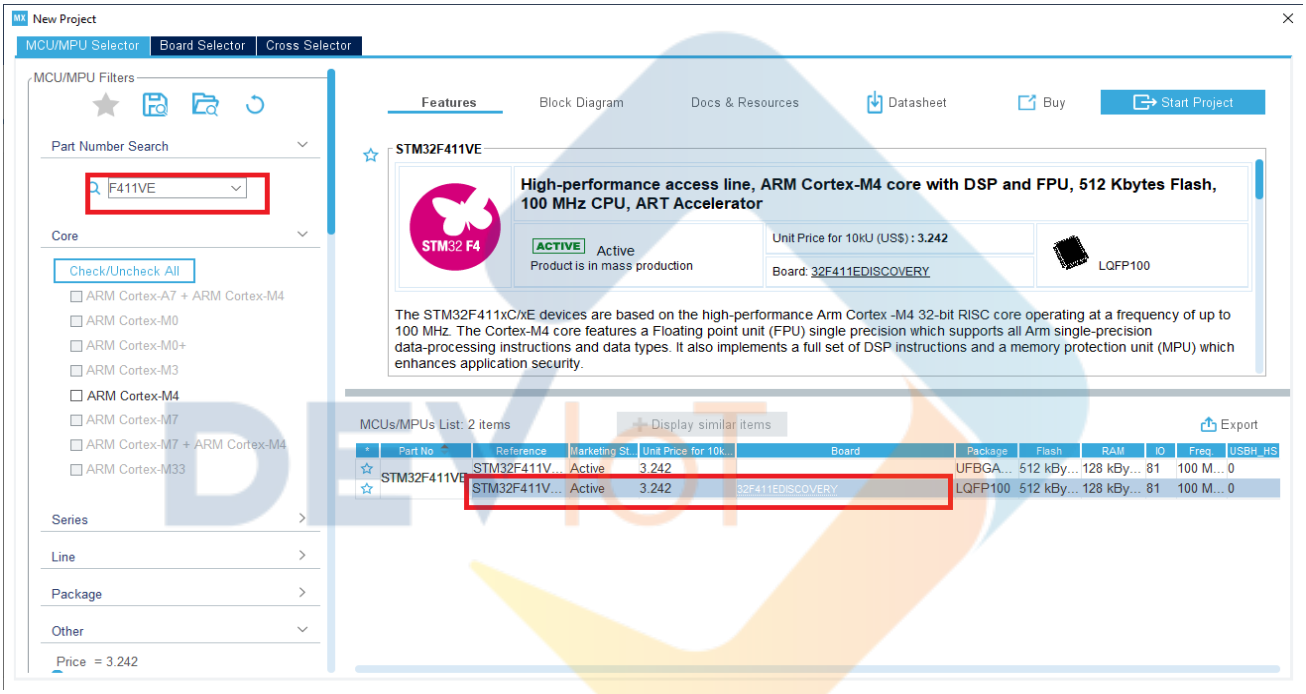
Thanh ghi ứng với các Channel chứa giá trị để so sánh với CNT.

Ý tưởng demo: Mình sẽ cấu hình Timer đến từ 0 → 9999, trong đó sẽ nạp sẵn giá trị 999 vào thanh ghi TIMx_CCRy và đảo mức logic ở chân OCy. Sau đó mình sẽ reset TIMx_CNT về 0 và bắt đầu lại chu trình. Các bạn sẽ nhìn thấy sự thay đổi xung nhờ việc nhấp nháy LED tại chân OCy nhé.

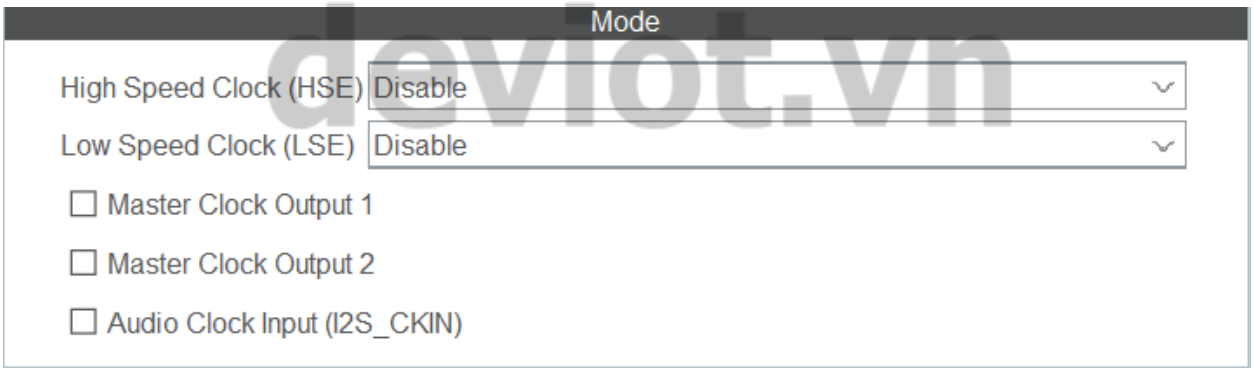
II, Lập trình

Mở phần mềm STMCubeMX, chọn dòng chip bạn sử dụng. Ở đây mình chọn chip STM32F411VE.

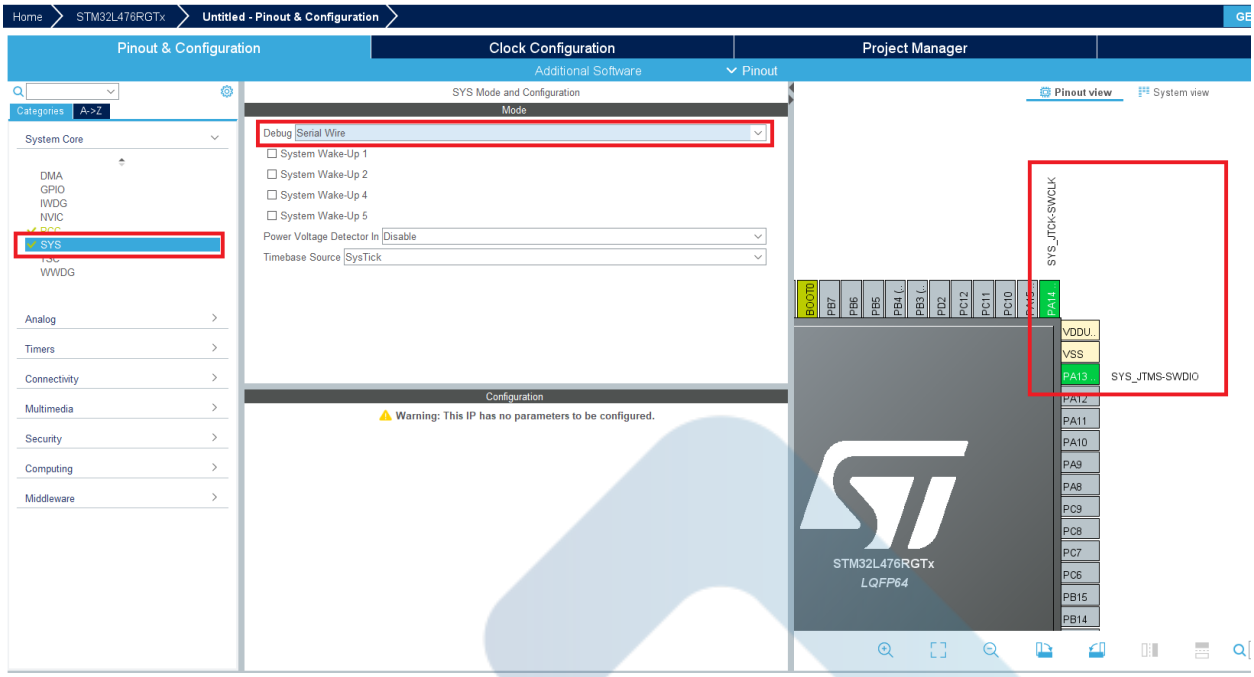
Đối với các dòng chip STM32 đời 4, tất cả mọi câu lệnh khi sử dụng thư viện HAL đều giống nhau. Chỉ khác nhau phần cấu hình Clock phụ thuộc riêng vào mỗi Chip.



Bài này mình sẽ sử dụng nguồn dao đông RC nội để cấp Clock cho Chip nhé.



Cấu hình Chip Debug sử dụng mode SWD.

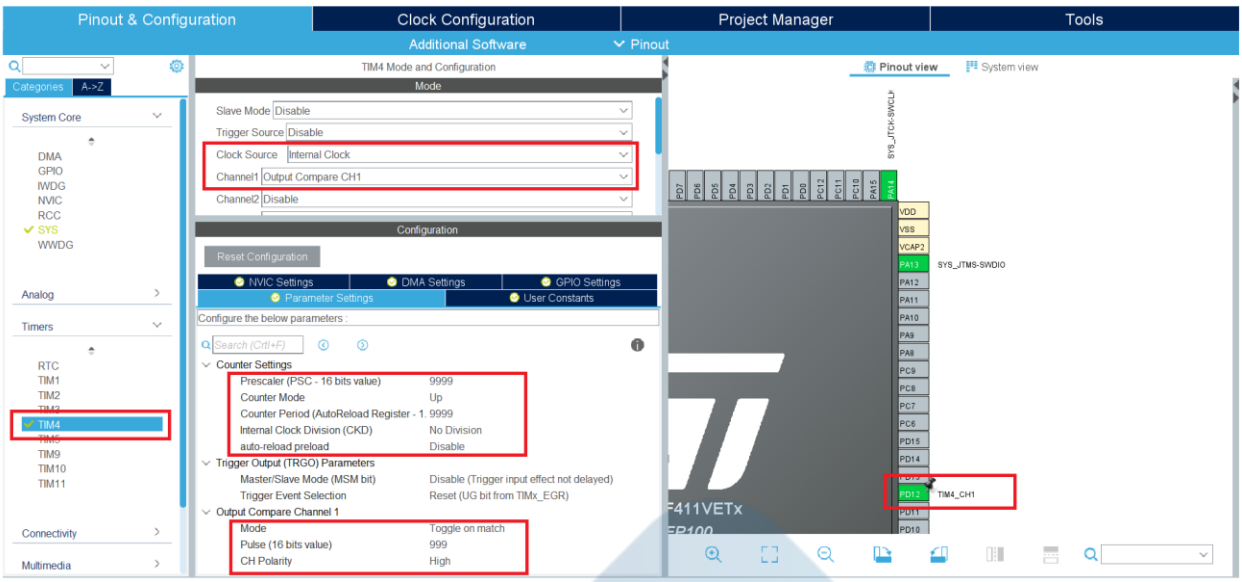


Ở đây mình sử dụng TIM4_CH1 để trùng với 1 chân LED trên DEV KIT, như vậy xung xuất ra trên chân OC1 sẽ làm cho LED sáng tối theo ý mình điều chỉnh.

Cấu hình chân là Output Compare lấy nguồn clock từ Clock nội (HSI) được chip hỗ trợ sẵn.

Cấu hình khi có ngắt xảy ra (TIM4_CNT = TIM4_CCR1) chân OC1 sẽ đảo mức logic (Toggle on match). Mình muốn nhấp nháy LED mà.

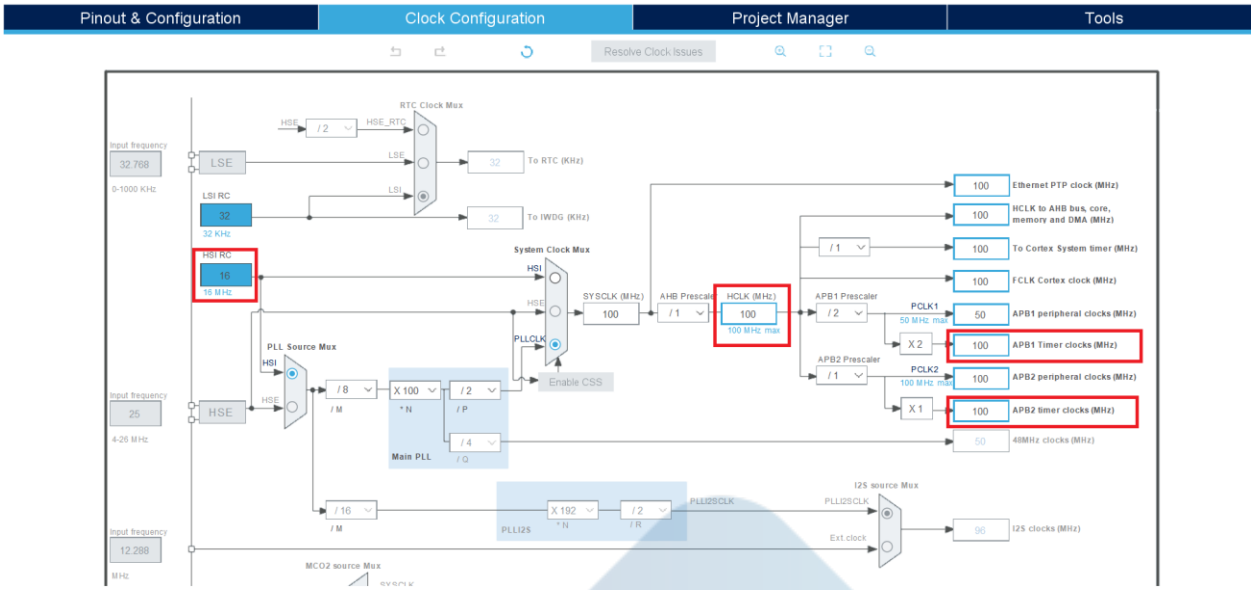
Chọn mức Polarity là mức High (cao) nhé.



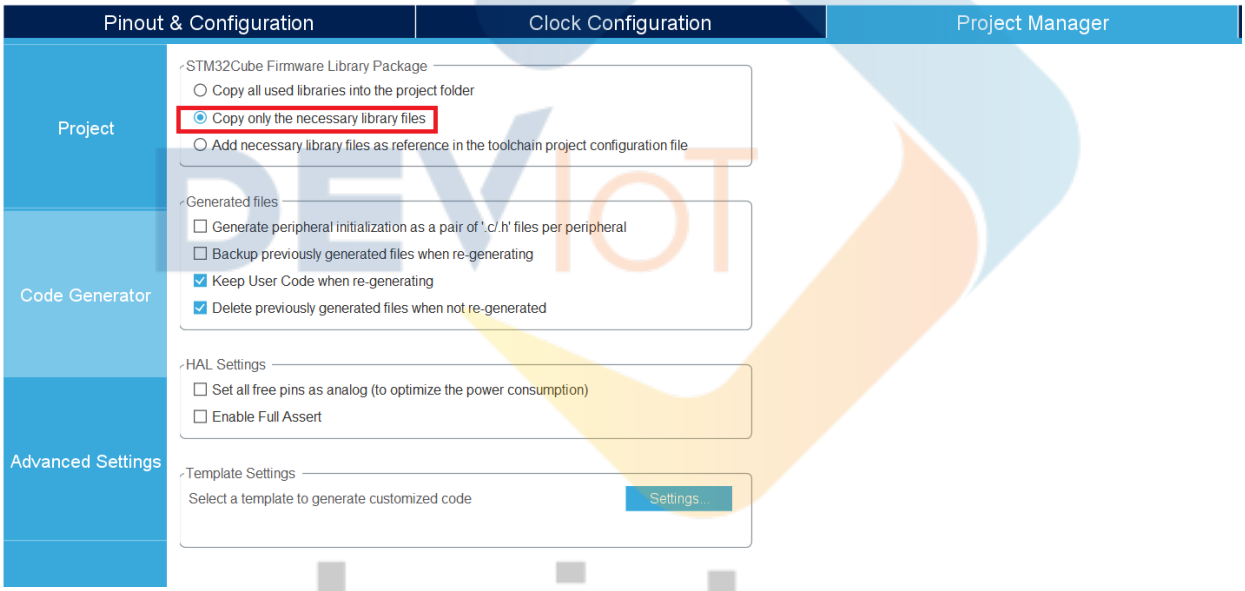
Ở đây mình chọn Prescaler = 9999 và Period = 9999 → $T_{timer} = (9999 + 1) * \frac{100,000,000}{(9999+1)} = 1(s)$.

Ở đây mình cài đặt giá trị TIM4_CCR1 là 999. Nghĩa là khi TIM4_CNT đếm từ 0 → 999 sẽ xảy ra ngắt Output Compare.

Tiếp theo mình sẽ cấu hình tần số Clock cho Chip như mọi bài mình chọn sử dụng nguồn Clock nội (HSI) được chip hỗ trợ sẵn, như vậy mình sẽ không cần hàn thêm thạch anh ngoại mà chương trình vẫn chạy được. Clock đi qua bộ nhân tần PLLCLK để đạt được tần số hoạt động tối đa mà chip hỗ trợ HCLK = 100MHz. Việc còn lại Cube MX sẽ tự cấu hình cho các bạn.



Cuối cùng chọn File và sinh code cho Project.



Chọn những thư viện cần thiết để sinh code nhanh hơn và giảm dung lượng Project nhé.

Nào bắt đầu đi vào sửa code nhé !!!

Ở hàm main, mình khai báo các hàm quan trọng.

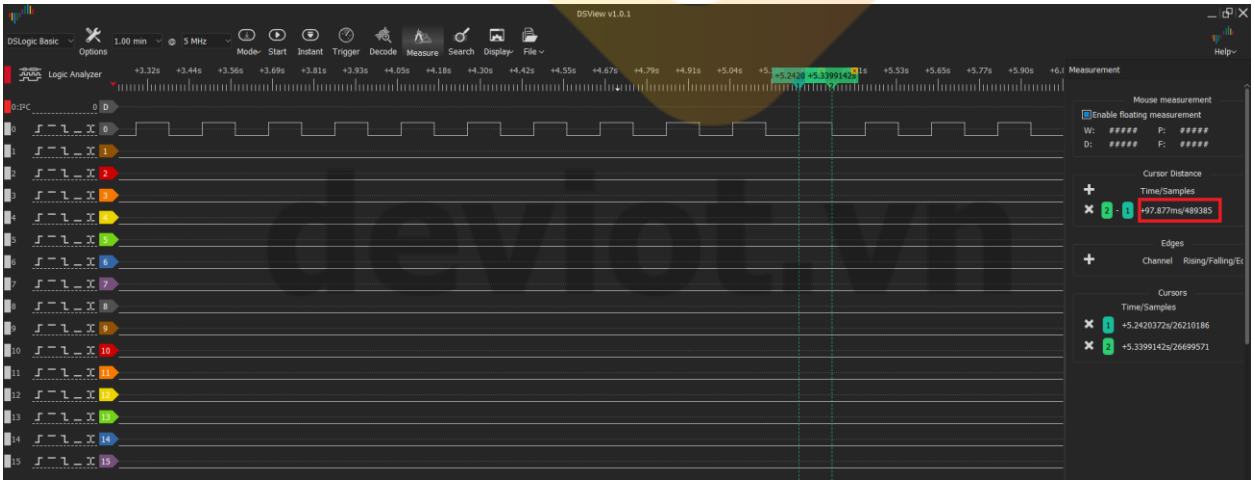
```c
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM4_Init();

  /*Start TIM4 Chanel 1 with interrupt*/
  HAL_TIM_OC_Start_IT(&htim4, TIM_CHANNEL_1);

  while (1)
  {

  }

}
```

Khi Output Compare xảy ra ngắt, mình sẽ reset biến đếm CNT về 0 và chân OC1 sẽ đảo mức logic làm thay đổi trạng thái LED.

Các bạn thử tính xem chu kì nhấp nháy LED sẽ là bao nhiêu nhé ?

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(htim->Instance == TIM4)
  {
    __HAL_TIM_SetCounter(&htim4,0);   // reset TIM4_CNT
  }
}
/* USER CODE END 0 */
```

Kết quả Demo trên chân xuất Output của TIM4



Đây chỉ là 1 trong 8 chế độ hoạt động của bộ Output Compare thôi nhé các bạn. Chúc các bạn thành công !!!

DEVIOT - CÙNG NHAU HỌC LẬP TRÌNH IOT

📌 Website: deviot.vn

📌 FanPage: Deviot - Thời sự kỹ thuật & IoT

📌 Group: Deviot - Cùng nhau học lập trình IOT

📌 Hotline: 0969.666.522

📌 Address: Số 101C, Xã Đàn 2

📌 Đào tạo thật, học thật, làm thật