Sử dụng Timer Input Capture

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, các 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 Capture

Tương tự như Compare thì mode Capture cũng có 4 kênh.

Input Capture : Ở chế độ Input capture, thanh ghi TIMx_CCRx của kênh đầu vào tương ứng sẽ được sử dụng để lưu giá trị của CNT khi phát hiện sự thay đổi mức logic (sườn lên/ sườn xuống) như được cấu hình trước đó. Từ đó có thể biết được khoảng thời gian giữa 2 lần có sườn lên hoặc sườn xuống.

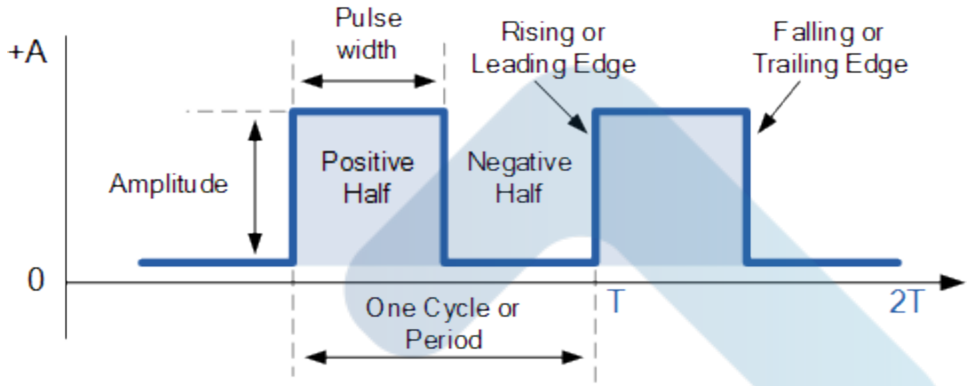
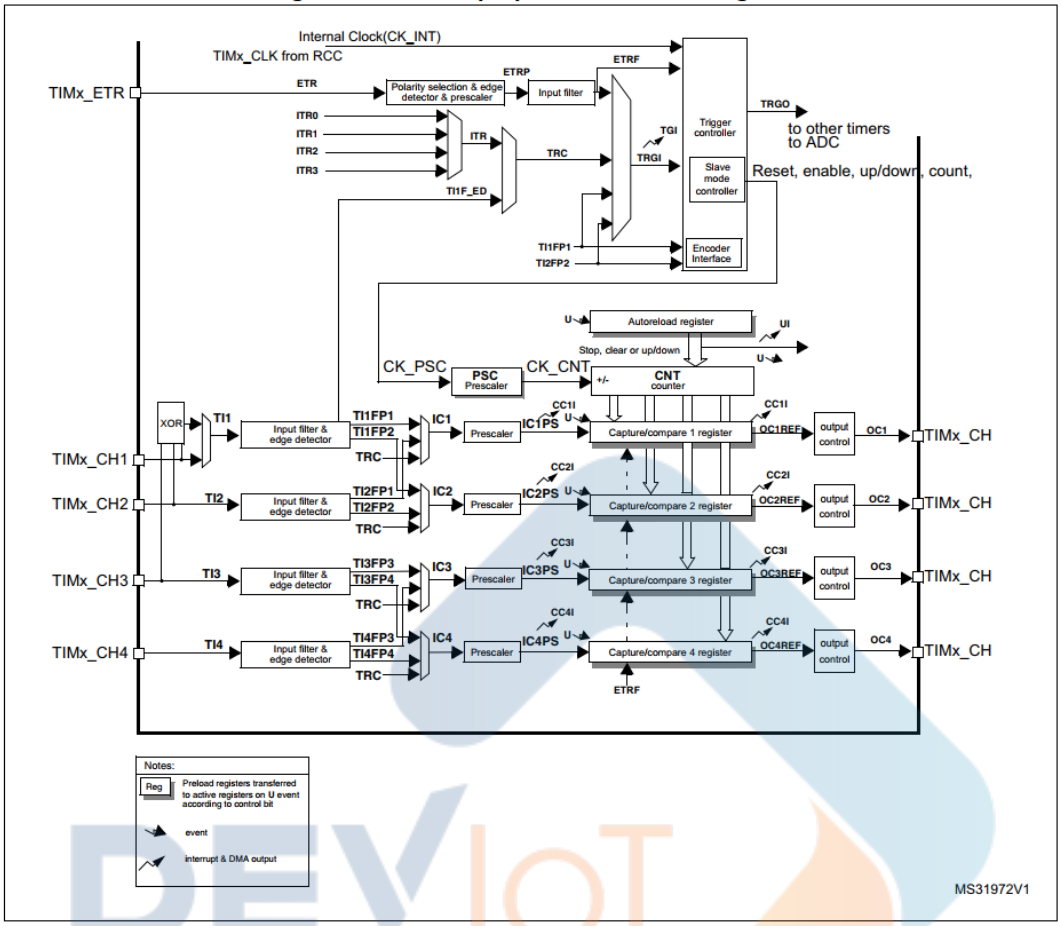


Figure 87. General-purpose timer block diagram



Một số thanh ghi quan trọng của 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

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

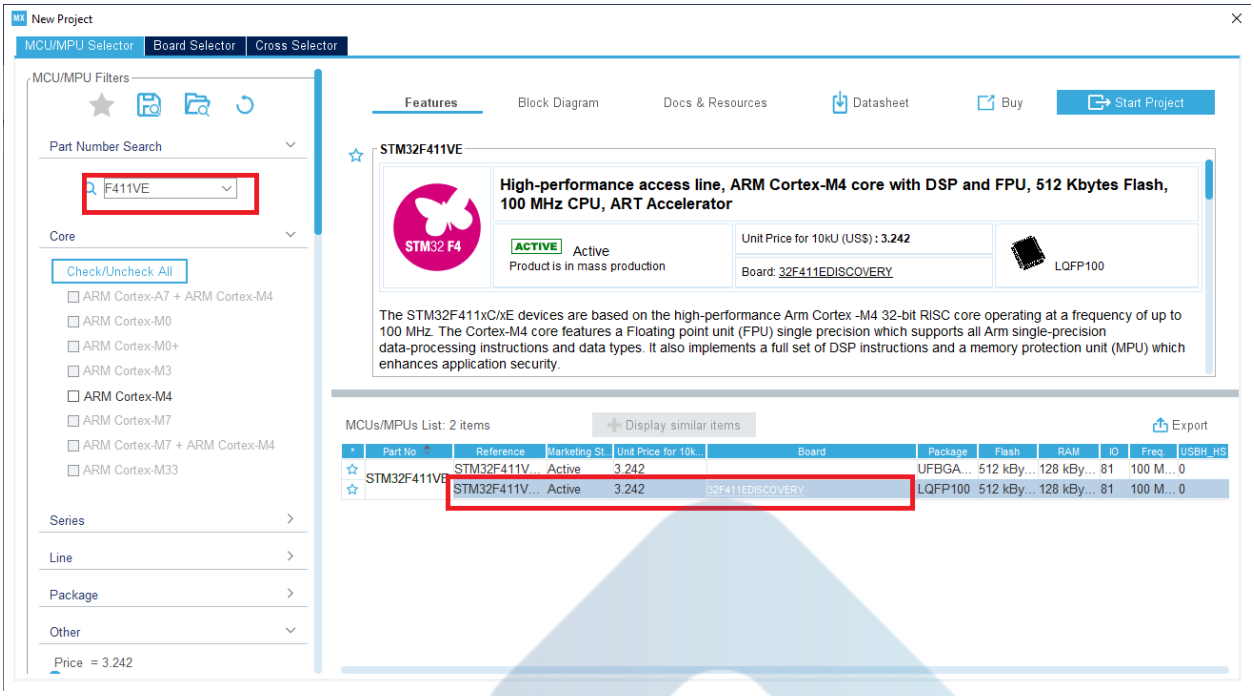
Thanh ghi ứng với các Channel sẽ lấy giá trị CNT khi tín hiệu đầu vào thay đổi theo mức logic 1 xuống 0 hay từ 0 lên 1.

Ý tưởng demo: Mình sẽ cấu hình cho Timer đếm từ 0→999999 và cấu hình Input Capture bắt xung sườn lên. Cứ mỗi lần mình bấm nút PA0 trên KIT, sẽ có sự thay đổi mức logic từ 0→1 và bộ Input Capture ghi giá trị thanh ghi *CNT* tới thanh ghi *TIMx_CCRx.* Cuối cùng mình sẽ đọc giá trị này ra để sử dụng cho các mục đích khác.

II, Lập trình

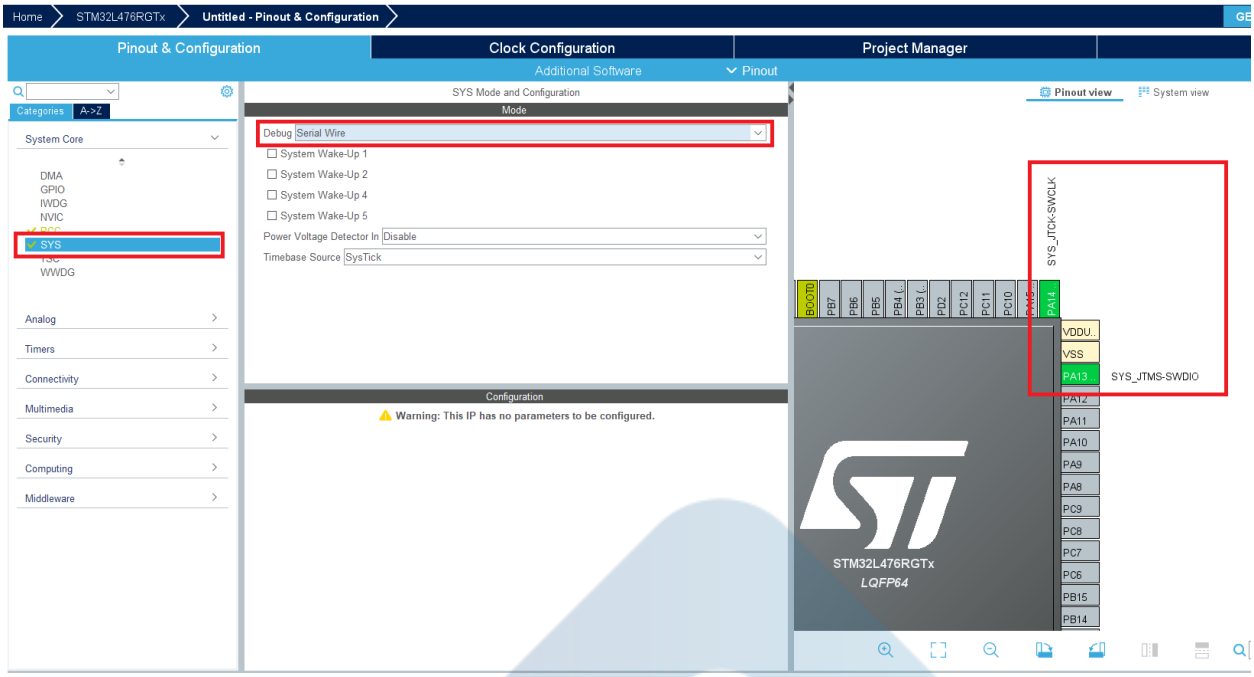
Mở phần mềm STM CubeMX, chọn dòng chip bạn sử dụng. Ở đây mình chọn chip STM32F411VE

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 hỗ trợ và cần chú ý phần này để cấu hình tần số hoạt động cho Chip.

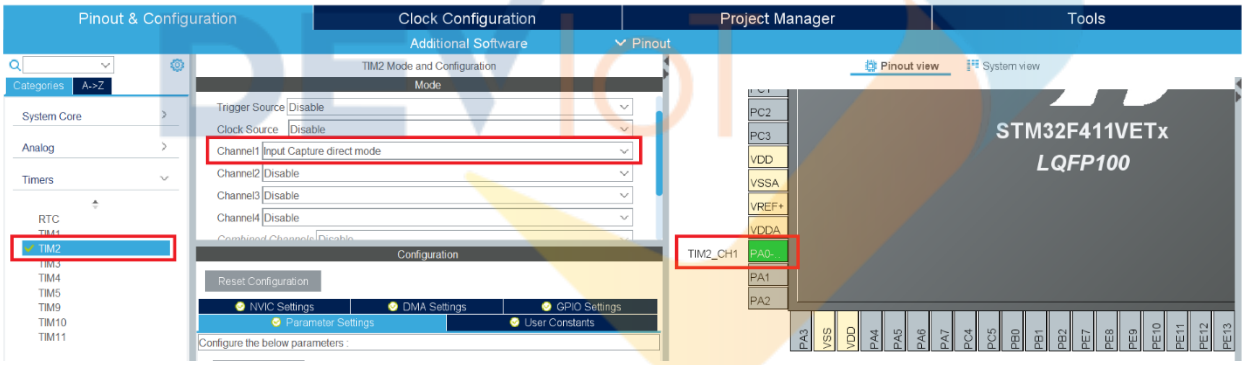


Sau đó cấu hình Chip sử dụng thạch anh ngoài hàn sẵn trên board mạch.

Cấu hình Chip Debug bằng mode SWD.

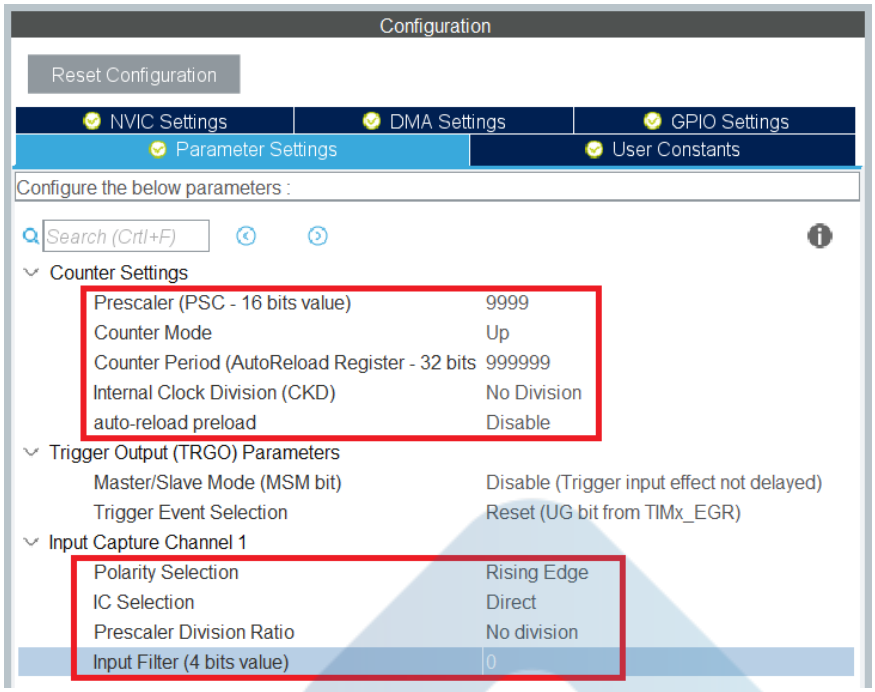


Cấu hình button trên board là TIM2_CH1 ứng với chân PA0 và Chọn nguồn clock là xung clock nội và mode là Input_Capture cho CH1.



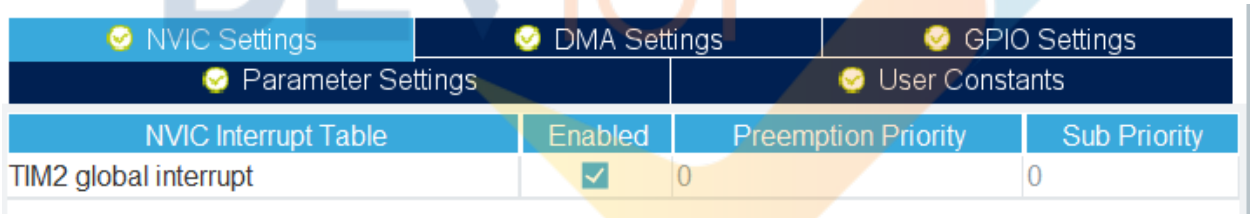
Tiếp theo ta sang cấu hình thông số cho Timer. Ở đây mình chọn Prescaler = 9999, Period = 999999. Chế độ Counter Up.

Đối với cấu hình như trên $F_{timer} = \frac{100,000,000}{(9999+1)} = 10,000$ (Hz). Tức cứ sau $\frac{1}{10,000}$ (s) CNT lại tăng lên 1 giá trị. Và CNT sẽ tăng tới giá trị 999999 (đếm từ 0 → 999999) sẽ được reset về 0.

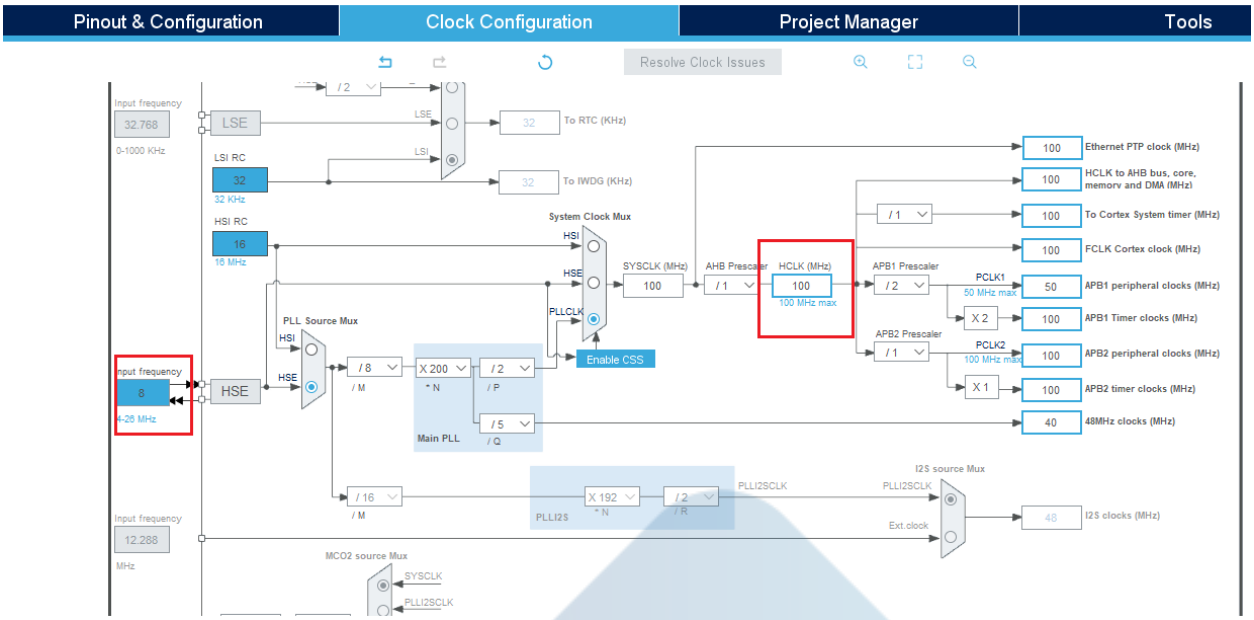


Polarity Selection : Rising Edge. Tức là khi chân PA0 nhận được 1 xung sườn lên (từ 0→1) nó sẽ ghi giá trị CNT vào thanh ghi TIMx_CCRx.

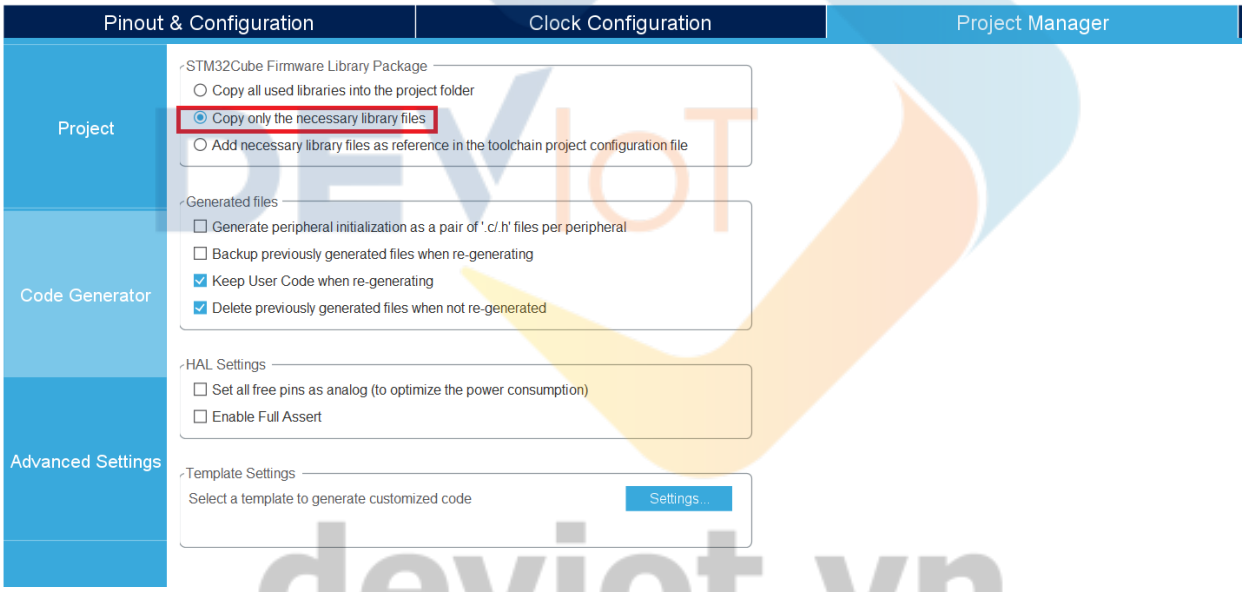
Tiếp theo ta sẽ Enable ngắt Timer lên



Giờ thì chuyển sang Tab Clock Configuration. Chọn đầu vào Clock thạch anh 8MHz. Chọn HCLK là 100MHz, tần số tối đa mà Chip hỗ trợ đạt được.



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.

Ở KeilC ta sẽ có các hàm khởi tạo cho Timer 2 và Channel input Capture tương tự như các bài trước.

```
166  static void MX_TIM2_Init(void)
167  {
168    TIM_MasterConfigTypeDef sMasterConfig = {0};
169    TIM_IC_InitTypeDef sConfigIC = {0};
170
171    htim2.Instance = TIM2;
172    htim2.Init.Prescaler = 9999;
173    htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
174    htim2.Init.Period = 999999;
175    htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
176    htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
177    if (HAL_TIM_IC_Init(&htim2) != HAL_OK)
178    {
179      Error_Handler();
180    }
181    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
182    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
183    if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
184    {
185      Error_Handler();
186    }
187    sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_FALLING;
188    sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
189    sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
190    sConfigIC.ICFilter = 0;
191    if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_1) != HAL_OK)
192    {
193      Error_Handler();
194    }
195  }
196
```

Cube đã tự sinh ra hàm khởi tạo cho chúng ta và bạn có thể sửa lại các thuộc tính ngay tại đây.

Trong main.c khai báo cáo biến cần thiết để lấy được thời gian khi nhấn nút.

```
32  /* USER CODE END PTD */
33  uint32_t counter = 0; // Get Value CNT resgister.
34  uint32_t input_capture = 0; // Get value TIMx_CCRx resgister.
```

Trong hàm main mình sẽ khởi tạo các Module cần thiết. Và liên tục đọc về giá trị CNT để hiển thị lên Watch.

```c
int main(void)
{

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM2_Init();

  /*Start TIMER2 with interrupt*/
  HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);

  while (1)
  {
    counter = __HAL_TIM_GetCounter(&htim2);    //read TIM2_CNT counter value
  }
  /* USER CODE END 3 */
}
```

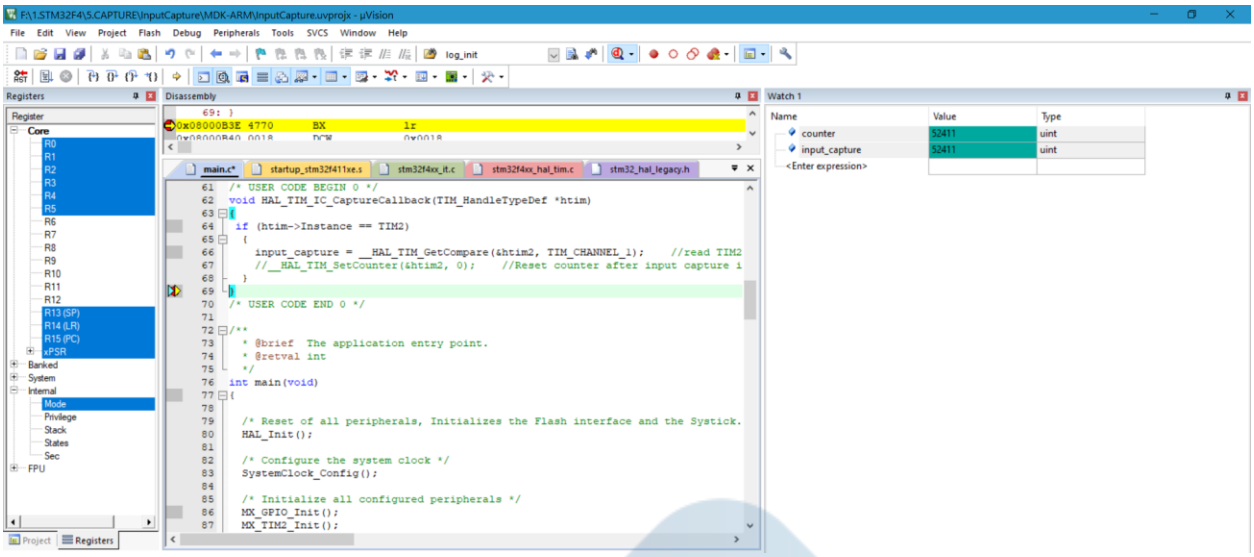
Tiếp theo mình sẽ khai báo 1 hàm xử lý ngắt khi có Input Capture xảy ra.

```c
/* USER CODE BEGIN 0 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
 if (htim->Instance == TIM2)
  {
    input_capture = __HAL_TIM_GetCompare(&htim2, TIM_CHANNEL_1);    //read TIM2 channel 1 capture value
    //__HAL_TIM_SetCounter(&htim2, 0);    //Reset counter after input capture interrupt occurs
  }
}
/* USER CODE END 0 */
```

Như vậy mỗi khi mình bấm nút, mình sẽ đọc lại giá trị của thanh ghi TIM2_CCR1 và in lên Watch.

Tiến hành Build Code và Nạp chương trình xuống KIT.

Kết quả Demo



Như các bạn thấy. Khi mình bắt đầu nạp chương trình, thanh ghi CNT bắt đầu đếm từ 0 sau đó tăng dần, khi mình bấm nút. Một ngắt Input Capture xảy ra và nó ngay lập tức bắt lấy giá trị của thanh ghi CNT và ghi vào TIM2_CCR1. Kết quả như trên hình.

Các bạn có thể ứng dụng chức năng này để tạo ra các mạch đếm độ rộng xung.

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

deviot.vn