

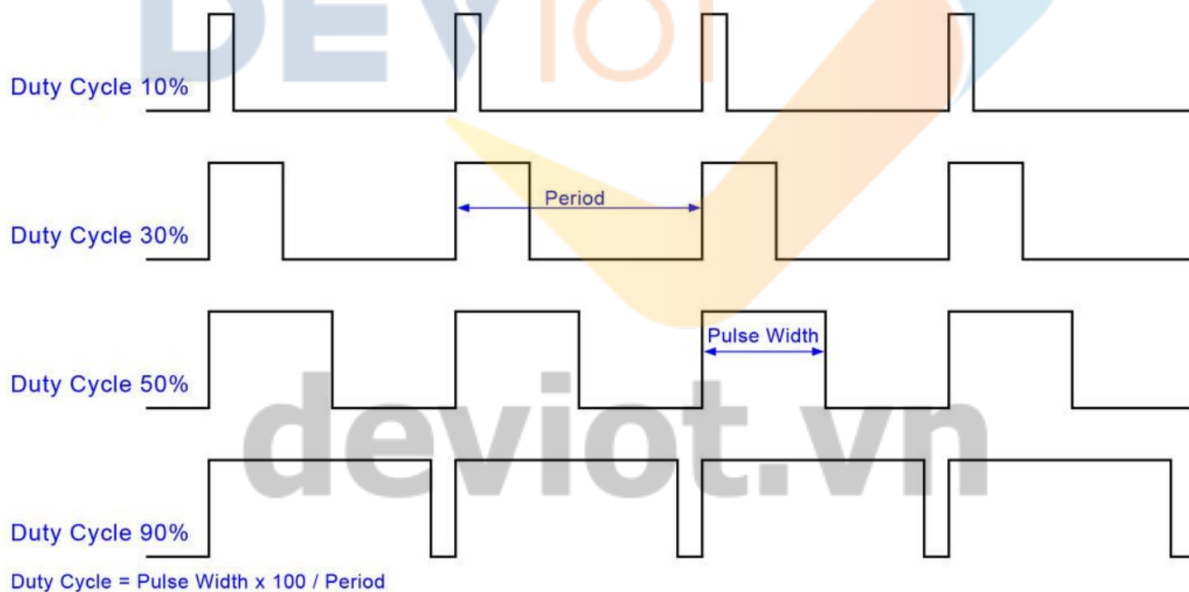
Bài 17: PWM với STM32

Giới thiệu

PWM (Pulse Width Modulation) – Điều chế độ rộng xung/Bấm xung: là phương pháp điều chỉnh giá trị điện áp trung bình ra tải như các thiết bị như động cơ, đèn LED,... từ đó có thể làm thay đổi công suất thiết bị (tốc độ động cơ, độ sáng của đèn,...). Điều này được thực hiện bằng cách thay đổi Duty Cycle của xung tín hiệu điện áp ra, là phương pháp dễ dàng và ít tốn kém hơn việc điều chỉnh các thông số của dòng điện.

I, Lý thuyết

- Để hiểu đơn giản PWM là gì, bạn hãy tưởng tượng ta cần thay đổi độ sáng của một đèn LED. Khi đó, nếu ta cho LED sáng tắt liên tục với tần số cao, mắt chúng ta sẽ mất khả năng phân tích sự sáng tối của LED(do hiện tượng lưu ảnh), từ đó thấy được độ sáng khác nhau của đèn. Đèn sáng tỏ tức là thời gian sáng của đèn cao, đèn sáng mờ tức là thời gian sáng của đèn thấp.



- **Duty Cycle** là tỷ lệ phần trăm mức cao.
- **Period** là chu kỳ xung.
- **Pulse Width** là thời gian mức cao trong một chu kì.

$$\text{Duty Cycle} = \frac{\text{Pulse width}}{\text{Period}} \times 100\%$$

Khi làm việc với STM32, tính năng PWM nằm trong khối Timer của vi điều khiển.

Một số thanh ghi quan trọng trong chế độ PWM Generation:

TIMx counter register (TIMx_CNT): Thanh ghi lưu giá trị của Counter (CNT).

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CNT[15:0] (Bit 15:0): Dãy bit lưu giá trị CNT.

TIMx auto-reload register (TIMx_ARR): Trong chế độ PWM, thanh ghi này lưu giá trị độ phân giải của xung PWM (ví dụ độ phân giải là 100 thì LED có thể có 100 mức độ sáng khác nhau)

Address offset: 0x2C

Reset value: 0xFFFF

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

ARR[15:0] (Bit 15:0): Dãy bit lưu giá trị ARR

TIMx capture/compare register x (TIMx_CCRx): Giá trị của thanh ghi này được dùng để làm mốc so sánh với Counter (sẽ được nói thêm ở phần dưới).

Reset value: 0x0000

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

CCR[15:0] (Bit 15:0): Dãy bit lưu giá trị CCR

TIMx prescaler (TIMx_PSC): Thanh ghi chứa giá trị chia xung clock được cấp vào từ bus APB.

Address offset: 0x28

Reset value: 0x0000

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

PSC[15:0] (Bit 15:0): Dãy bit lưu giá trị chia xung PSC.

$$F_{timer} = \frac{F_{hệ\ thống}}{(Prescale + 1)}$$

Vi điều khiển STM32 có hỗ trợ 2 chế độ tạo xung PWM như sau:

Mode 1

Nếu sử dụng chế độ đếm lên thì ngõ ra sẽ ở mức logic 1 khi CNT < CRR và ở mức 0 nếu CNT > CRR.

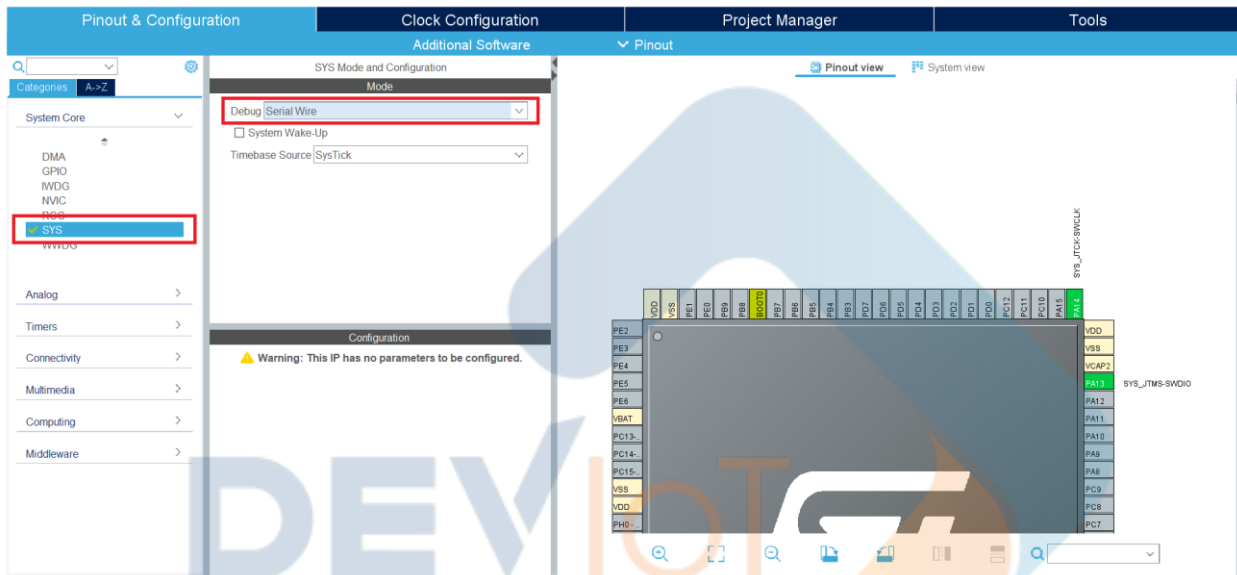
Mode 2

Nếu sử dụng chế độ đếm lên thì ngõ ra sẽ ở mức logic 0 khi CNT < CRR và ở mức 1 nếu CNT > CRR.

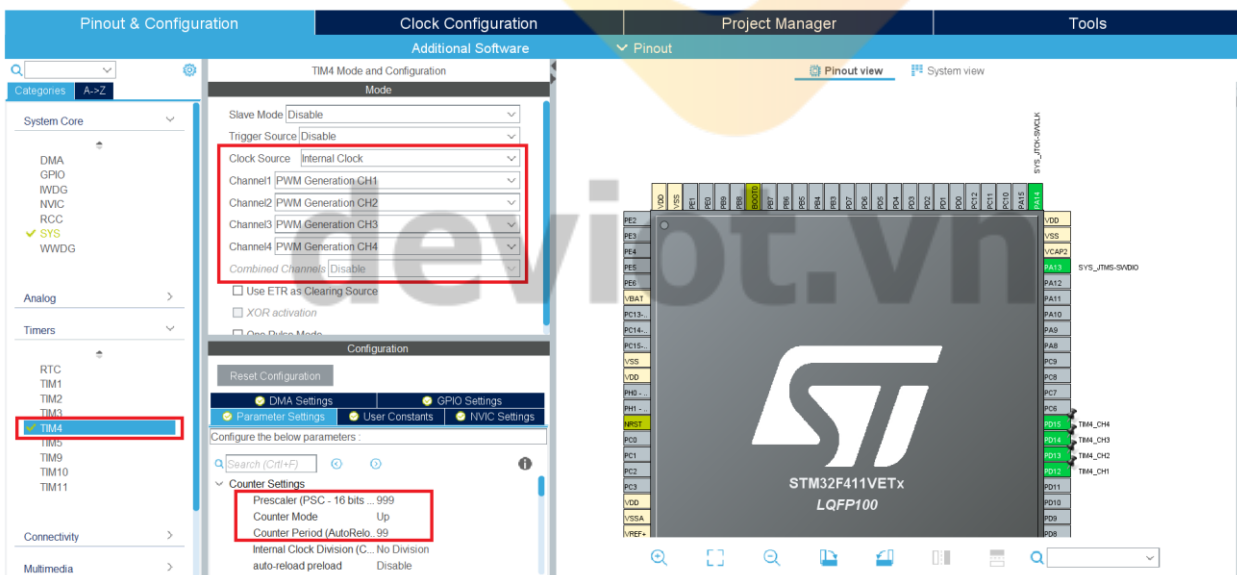
II, Lập trình

Ý tưởng demo: Mình sẽ sử dụng 4 kênh PWM của TIM4 trùng với 4 con LED trên KIT STM32F411VE của mình nhé. Sau đó sẽ xuất 4 xung PWM có duty cycle tăng dần làm và quan sát 4 LED trên KIT.

Đầu tiên cấu hình chế độ Debug Serial Wire

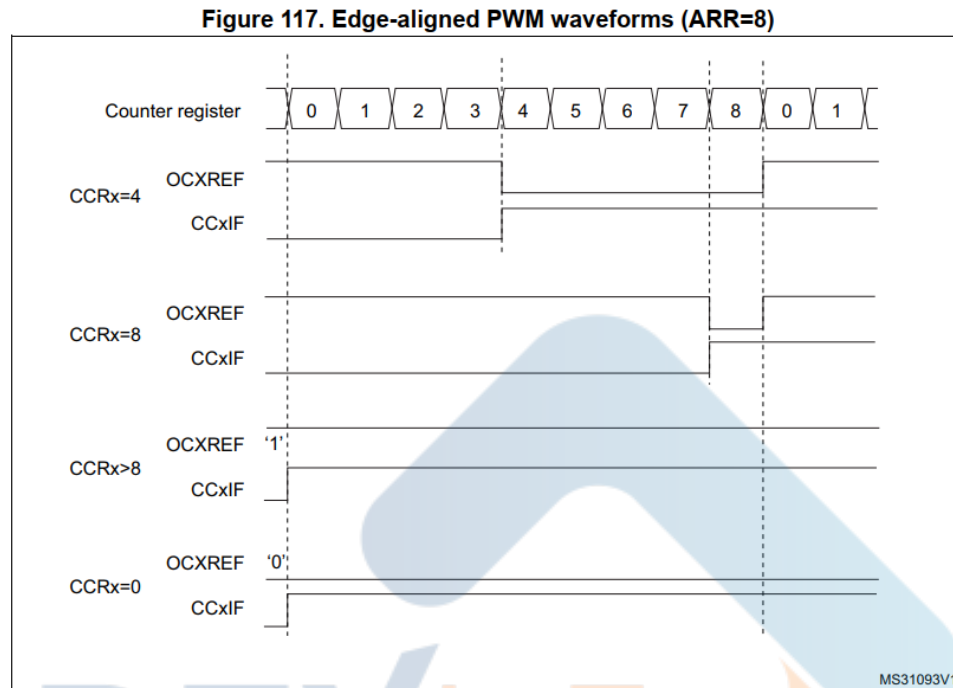


Tiếp theo mình cấu hình 4 Chanel PWM trên TIM4



Ở đây mình cài đặt Timer ở chế độ CountingUp có Period đếm từ 0 → 99.

Các bạn theo dõi hình sau:

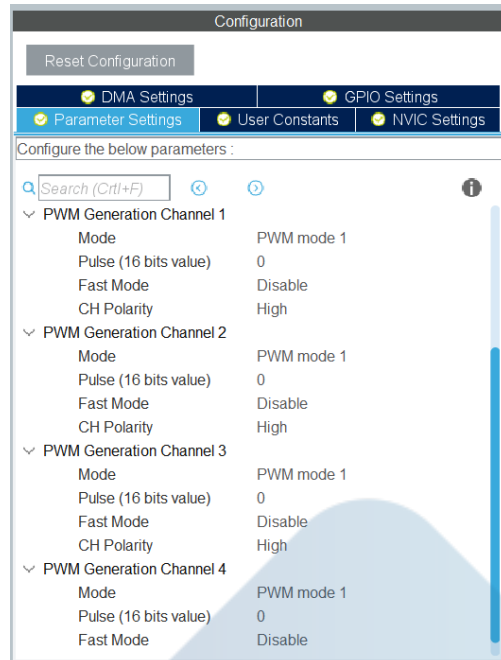


Như vậy

- duty cycle = 0 → CCRx = 0
- duty cycle = 1 → CCRx = 1
- duty cycle = 99 → CCRx = 99
- duty cycle = 100 → CCRx = 100

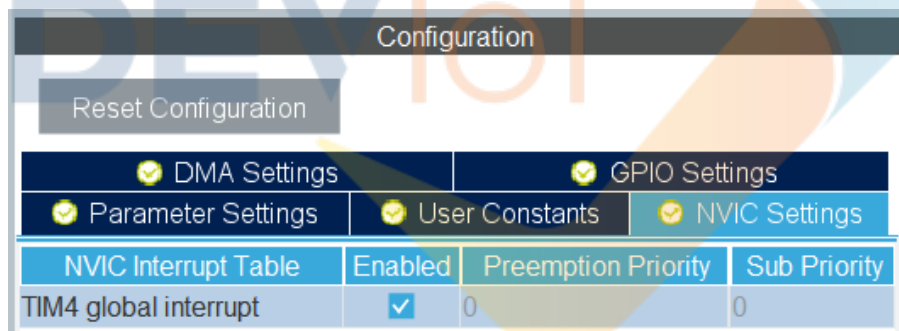
Tiếp theo cài đặt thông số 4 Channel PWM

deviot.vn

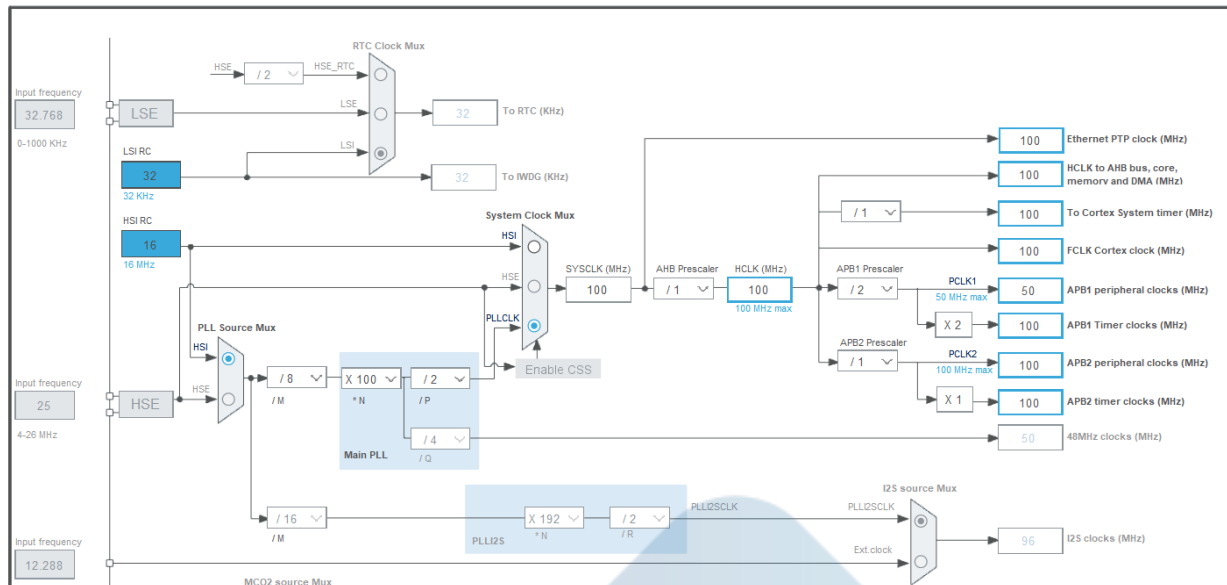


Bạn chọn CH Polarity là High, như vậy khi $TIMxCNT < TIMxCCRx$ thì xung xuất ra ở mức “1”.

Enable ngắt cho TIM4.

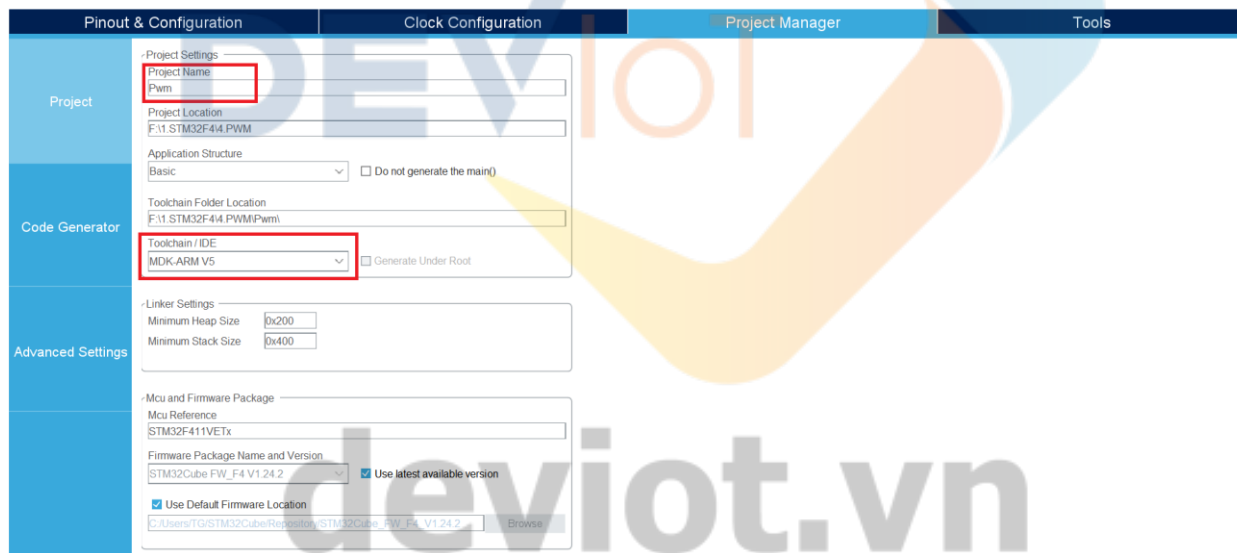


Tiếp theo chúng ta sang cài đặt Clock cho hệ thống, ở đây mình sẽ sử dụng nguồn Clock nội có sẵn trong Chip.



Nhìn vào đây có thể thấy bộ Timer được cung cấp nguồn Clock 100Mhz.

Chúng ta cấu hình tên dự án và để Cube MX sinh source code.



Nào giờ hãy cùng nhau bắt tay vào sửa code nhé.

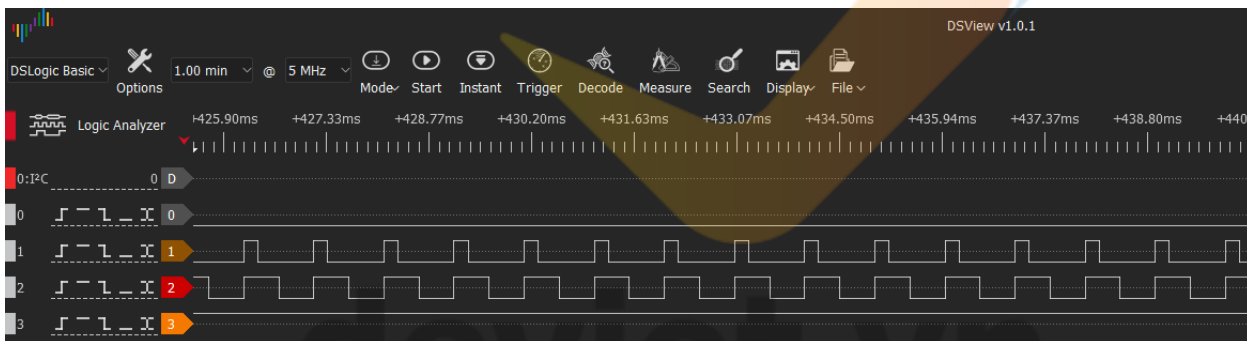
```

68 int main(void)
69 {
70     HAL_Init();
71     SystemClock_Config();
72     MX_GPIO_Init();
73     MX_TIM4_Init();
74
75     /*Start exporting 4 PWM pulses without generating interrupts*/
76     HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
77     HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
78     HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
79     HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);
80
81     __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_1,0);    // set duty cycle = 0
82     __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_2,20);   // set duty cycle = 20
83     __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_3,50);   // set duty cycle = 50
84     __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_4,100);  // set duty cycle = 100
85
86     while (1)
87     {
88         /* USER CODE END WHILE */
89
90         /* USER CODE BEGIN 3 */
91     }
92     /* USER CODE END 3 */
93 }
94


```


Sau đó build code và nạp lên KIT.

Dưới đây là xung xuất ra trên 4 kênh PWM.



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

