
payitforward.edu.vn

Pay It Forward

TIMER

C21

Nội dung:

1. **Các khái niệm cơ bản**
2. **Các chế độ của timer**
3. Ứng dụng timer

1. Các khái niệm cơ bản

1.1 Timer là gì?

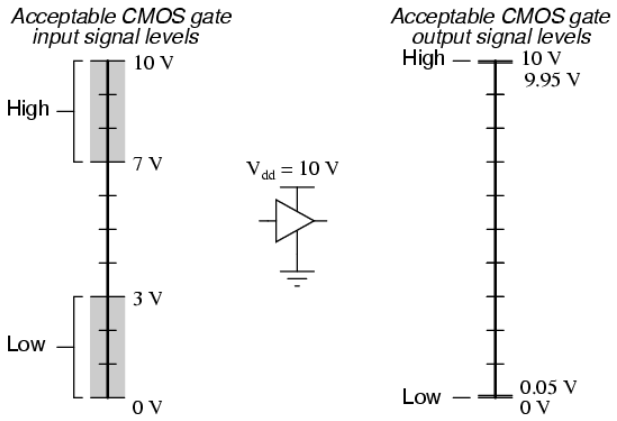
+ Timer là đồng hồ bấm giờ

+ Đồng hồ bấm giờ thì có gì hay

1. Các khái niệm cơ bản

1.2 Timer trong vi điều khiển để làm gì?

+ Như chúng ta đã được học trong các bài trước, vi điều khiển có chức năng là điều khiển điện áp theo các "mức logic", ngoài chức năng này ra ra còn có các ngoại vi thêm vào

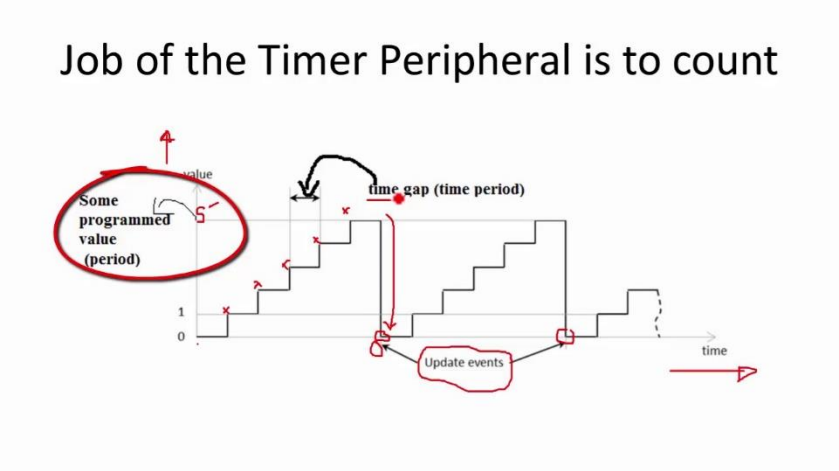


1. Các khái niệm cơ bản

1.2 Timer trong vi điều khiển để làm gì?

+ Timer là một ngoại vi, chức năng của nó là đếm

+ Timer nhận clock từ nguồn xung clock, sau đó sẽ đếm xem có bao nhiêu xung clock được đưa vào ngoại vi này.



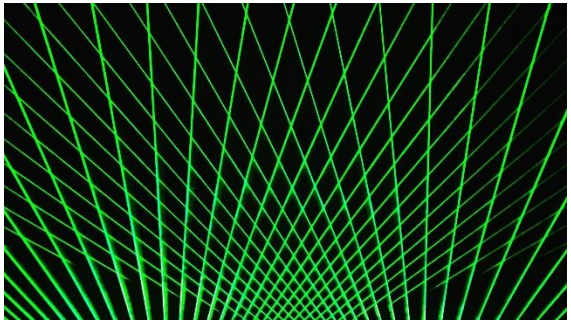
Mô tả hoạt động của timer, giá trị được đếm lên.

1. Các khái niệm cơ bản

1.2 Timer trong vi điều khiển để làm gì?

+ Cơ mà đếm thế thì được gì 

+ Chúng ta hãy tưởng tượng như thế này, giả sử ta cần đếm người ra vào tòa nhà, ta sẽ dùng 1 cái laser chiếu qua cửa để tên nào đi qua thì sẽ có 1 cái cảm biến đọc được tia laser bị khuất, sẽ tạo 1 thay đổi điện áp từ mức 0 -> 1. Nhiệm vụ của chúng ta sẽ là đếm số người bằng vđk



1. Các khái niệm cơ bản

1.2 Timer trong vi điều khiển để làm gì?

+Như các bài trước chúng ta có thể nối chân GPIO với sensor để đọc, phương pháp đọc có thể là dùng ngắt hoặc là đọc liên tục bằng cách dùng while(1)

+ Tuy nhiên những cách này đều phải sử dụng CPU để xử lý, trong khi thời gian CPU xử lý những thứ này có thể dùng làm việc khác.

=> Sử dụng timer bằng cách đưa tín hiệu vào làm xung clock đếm cho timer, ta có thể tiết kiệm thời gian cho CPU

1. Các khái niệm cơ bản

1.2 Timer trong vi điều khiển để làm gì?

+ Đó là ứng dụng của timer, tất nhiên dùng chân GPIO cũng được, tuy nhiên lợi thế của một ngoại vi là không tốn tài nguyên của CPU quá nhiều

+ Ngoài ra timer có thể sử dụng nguồn clock nội và ngoại để thực hiện các ứng dụng khác như phát xung PWM, ngắt,….

1. Các khái niệm cơ bản

1.2 Timer trong vi điều khiển để làm gì?

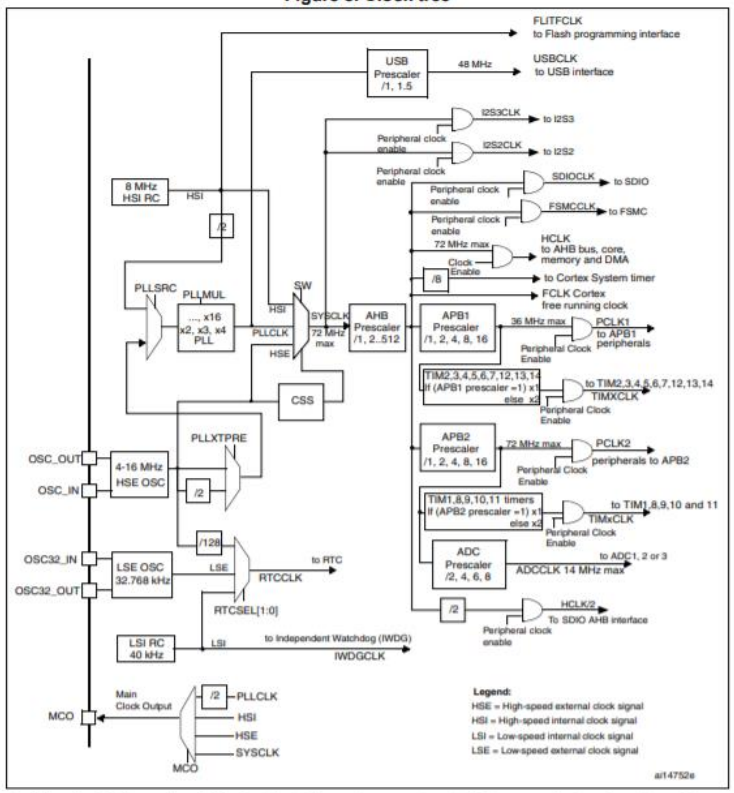
+ Đó là ứng dụng của timer, tất nhiên dùng chân GPIO cũng được, tuy nhiên lợi thế của một ngoại vi là không tốn tài nguyên của CPU quá nhiều

+ Ngoài ra timer có thể sử dụng nguồn clock nội và ngoại để thực hiện các ứng dụng khác như phát xung PWM, ngắt,….

1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

Figure 8. Clock tree



1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.

Phân phối clock của STM32

1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

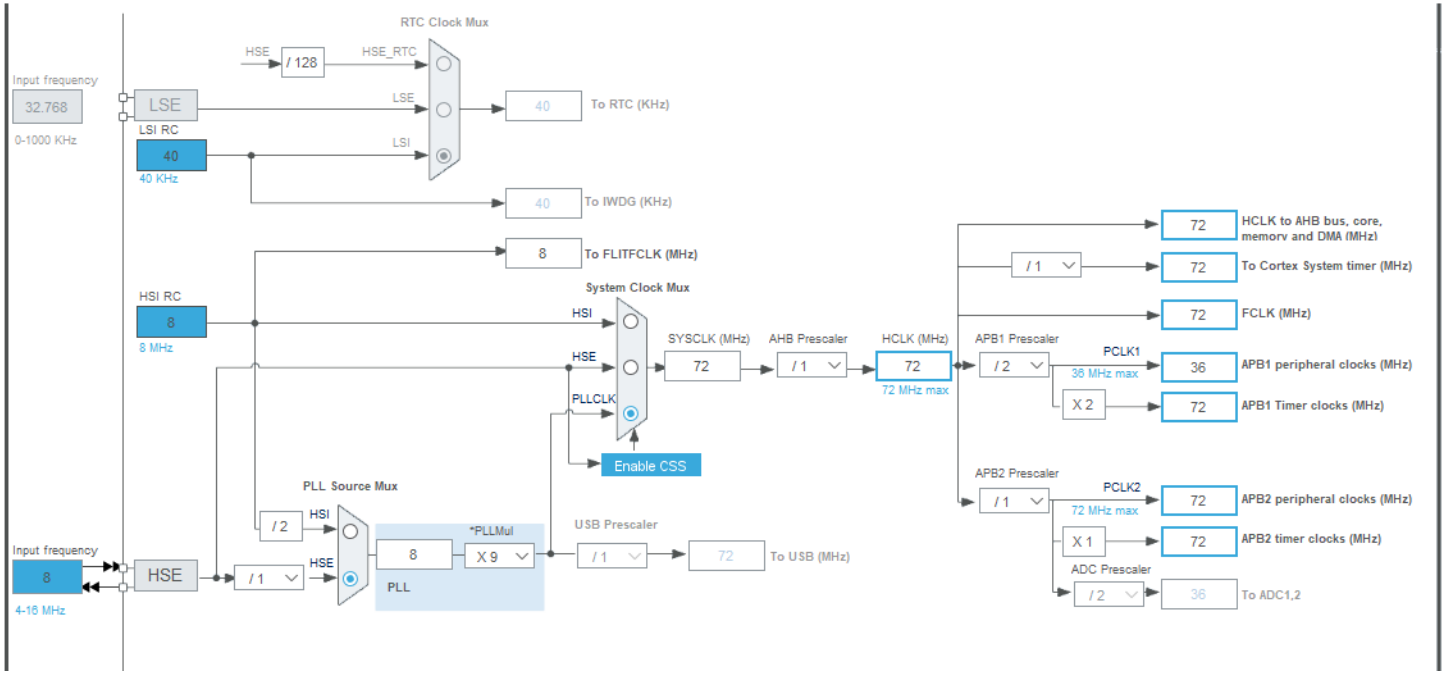
+ Tham khảo clock tree kĩ hơn ở đây:

https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

Trong STM32CubeIDE:



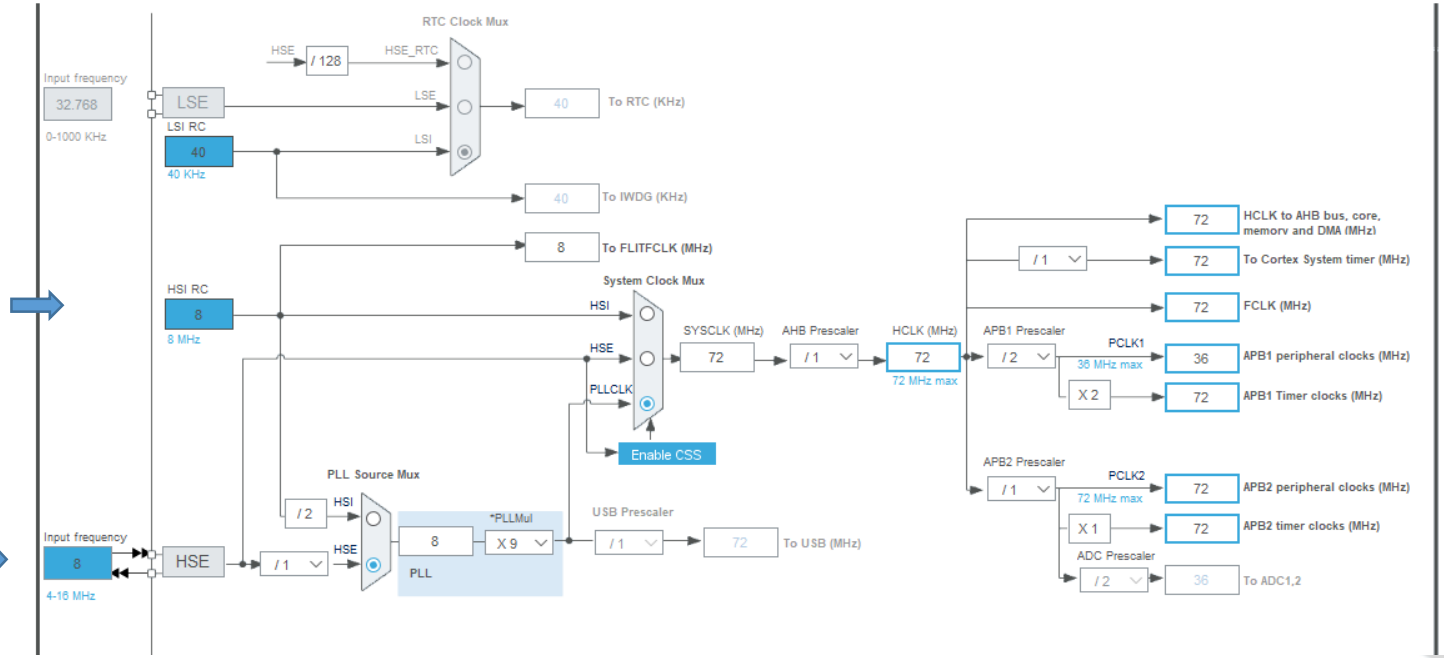
1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

Trong STM32, có thể sử dụng 2 loại xung clock chính cho chip, đó là HSI, tức mạch RC nội, hoặc HSE, clock bên ngoài, có thể từ thạch anh,….

Clock nội

Clock ngoại



1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

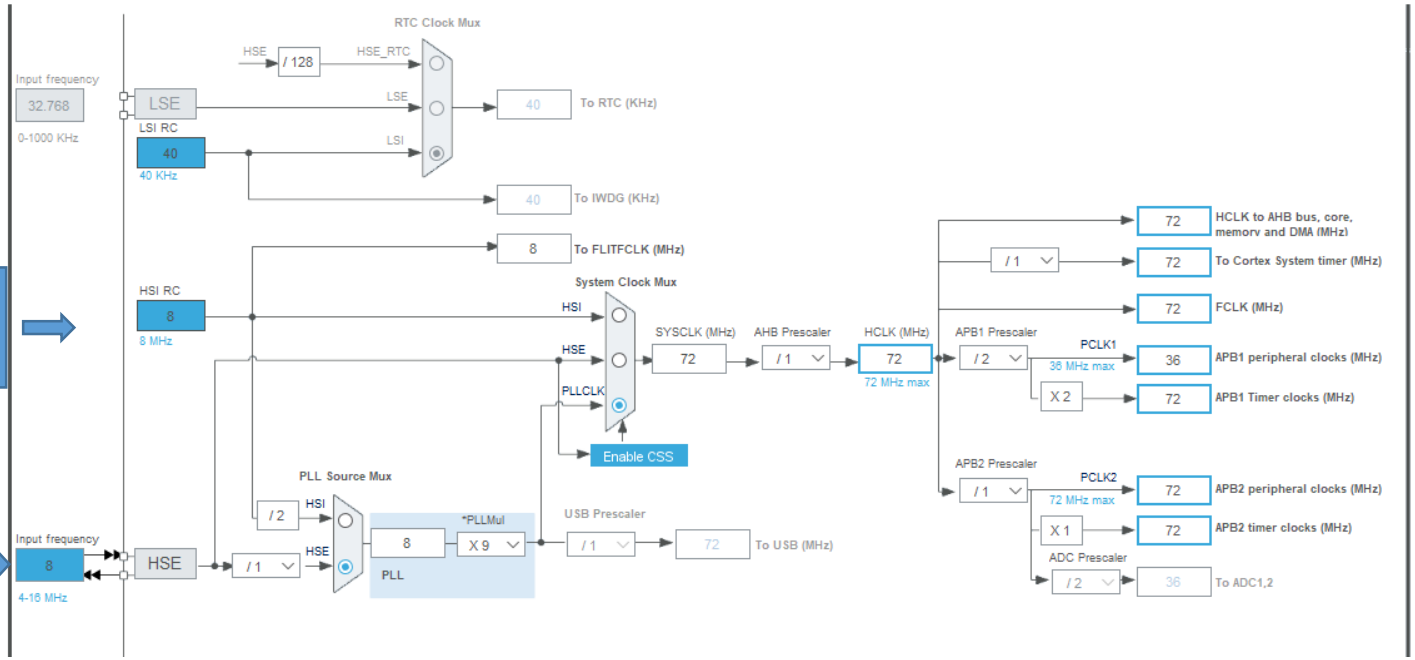
Điểm khác biệt của 2 loại clock này là:

+ HSI có độ chính xác +- 5%, là mạch RC nội trong con chip

+ HSE có thể chính xác đến +- 0.2%, vì vậy với ứng dụng như USB sẽ dùng thạch anh ngoại

Nguồn clock nội

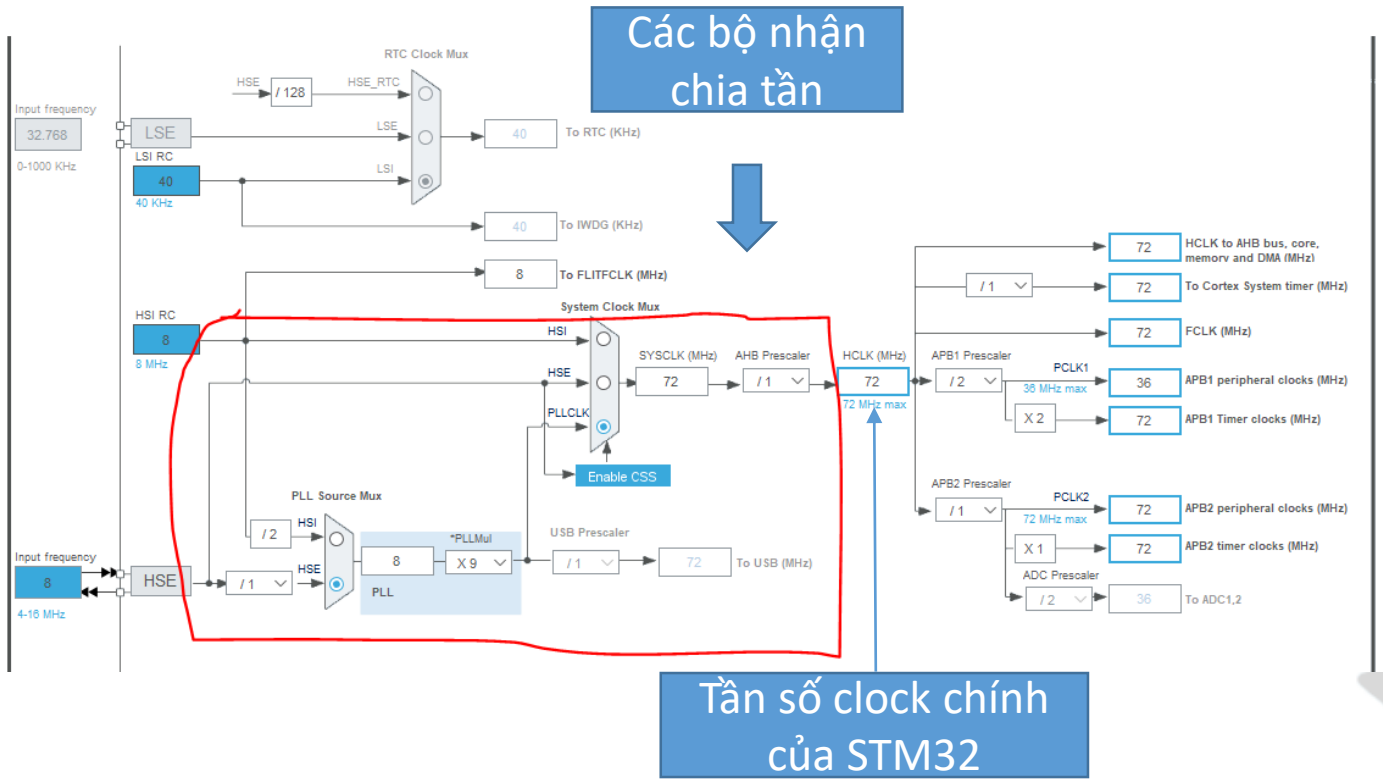
Nguồn clock ngoại



1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

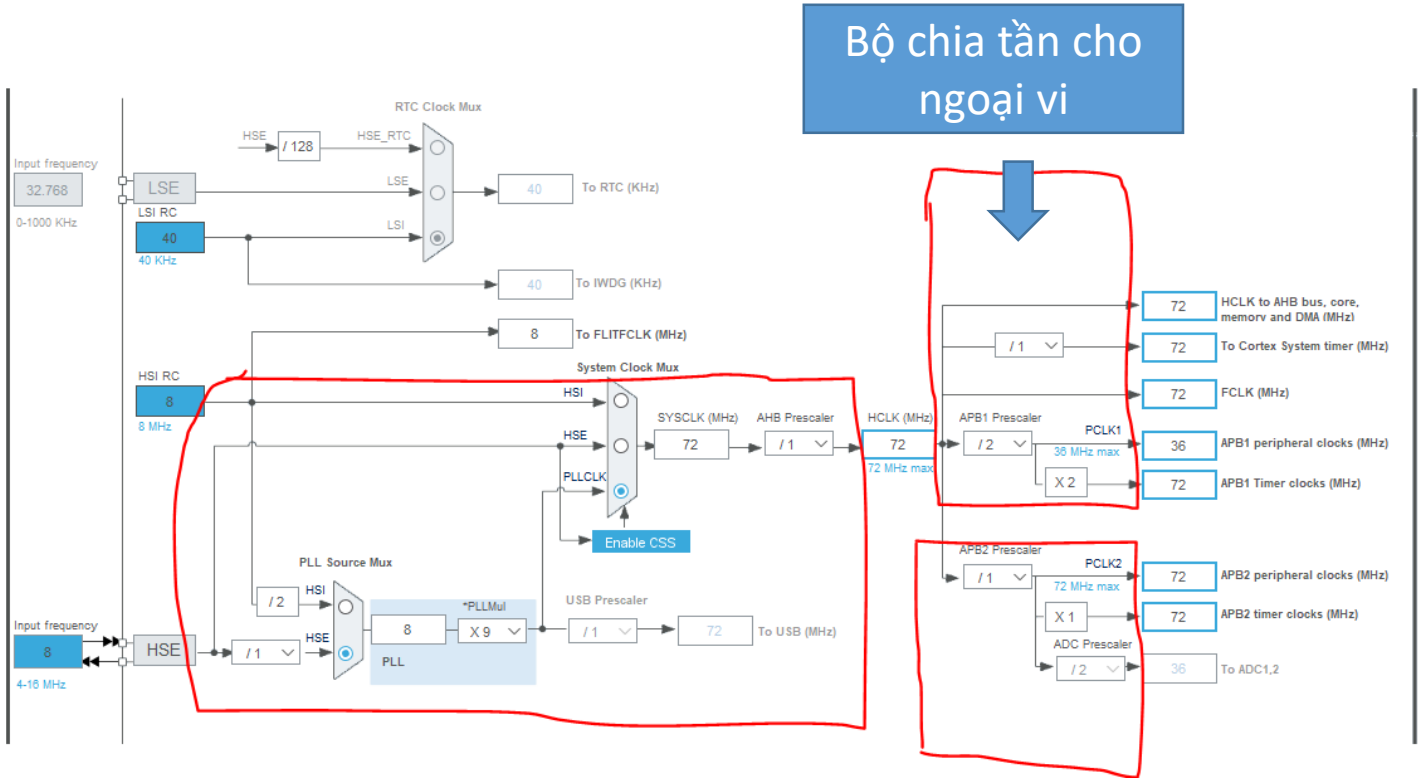
Clock này sẽ được đưa qua một bộ nhân tần và bộ chia tần, mục đích là để cấu hình xung nhịp phù hợp cho từng loại ứng dụng:



1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

Clock từ clock nội của chip, sẽ qua các bộ nhân tần và chia tần để đến với clock cấp cho ngoại vi:

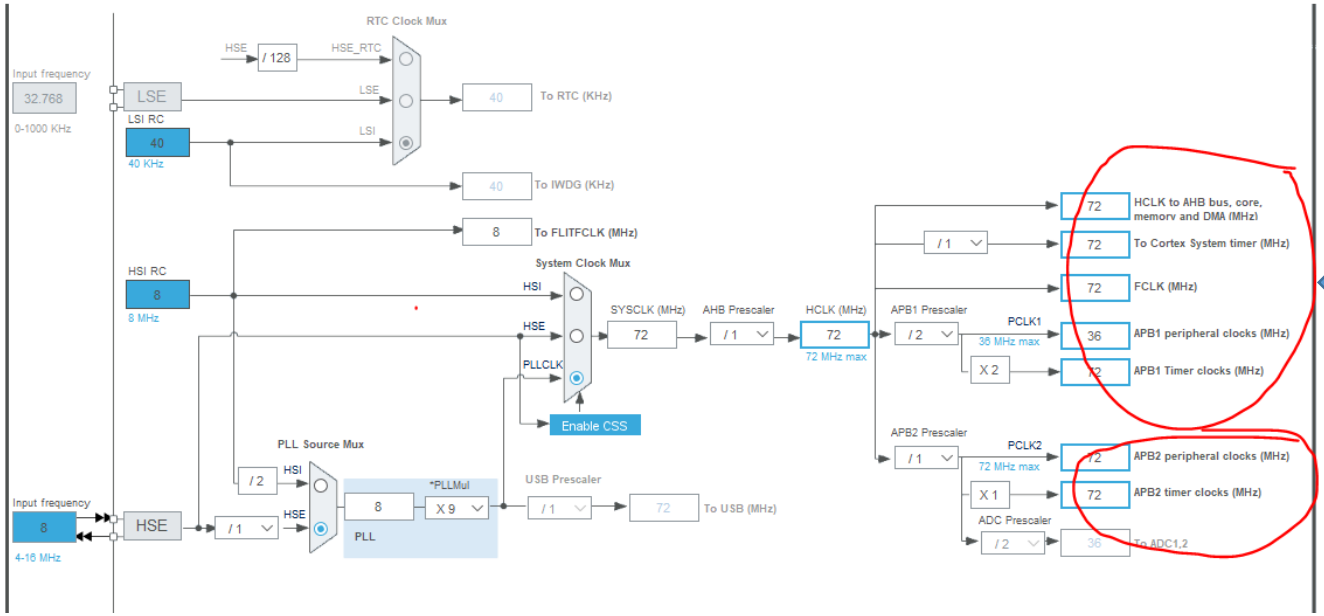


1. Các khái niệm cơ bản

1.3 Clock cho timer lấy ở đâu?

Clock ngoại vi này sẽ được cấp cho các ngoại vi như timer, USB, ADC,…..

Ví dụ: Timer 1 sử dụng clock từ APB1, có tần số tối đa là 36Mhz



Clock cấp cho các ngoại vi

1. Các khái niệm cơ bản

1.4 Các chế độ hoạt động của timer STM32

Dùng clock từ APB:

+ Counter
+ Input Capture
+ PWM Generate
+ Encoder mode
+….

Dùng clock ngoại:

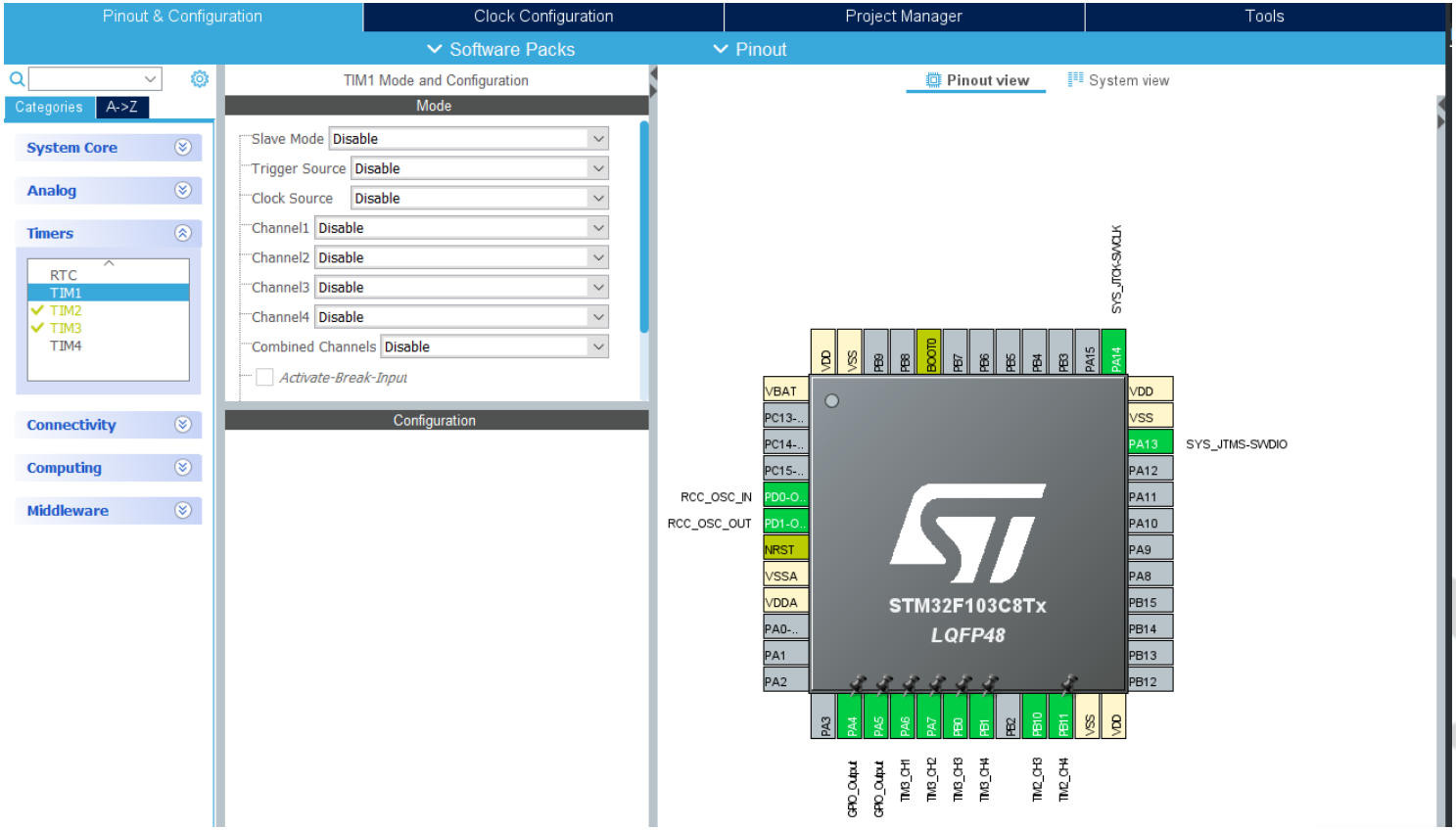
+ Input Capture
+ PWM Generate
+….

Ta sẽ tìm hiểu Mode Counter và PWM trong bài này

2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

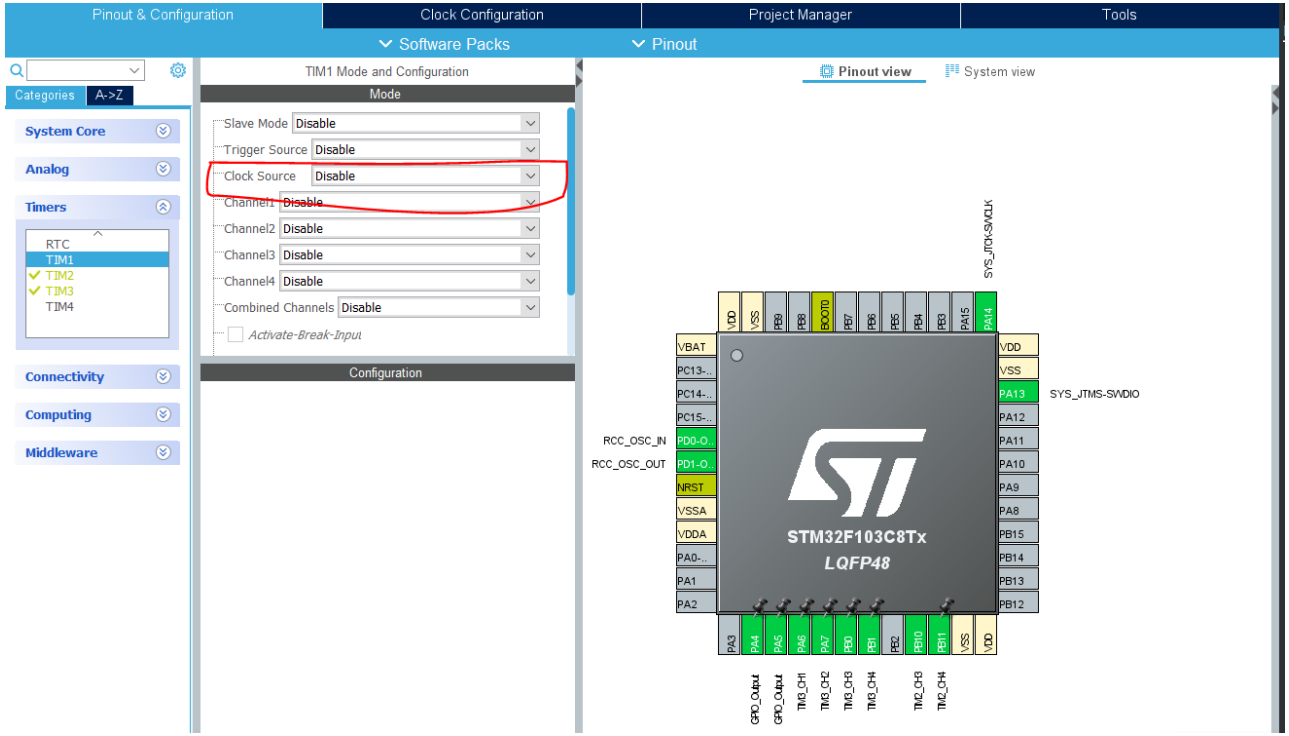
Setup clock cho Timer:



2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

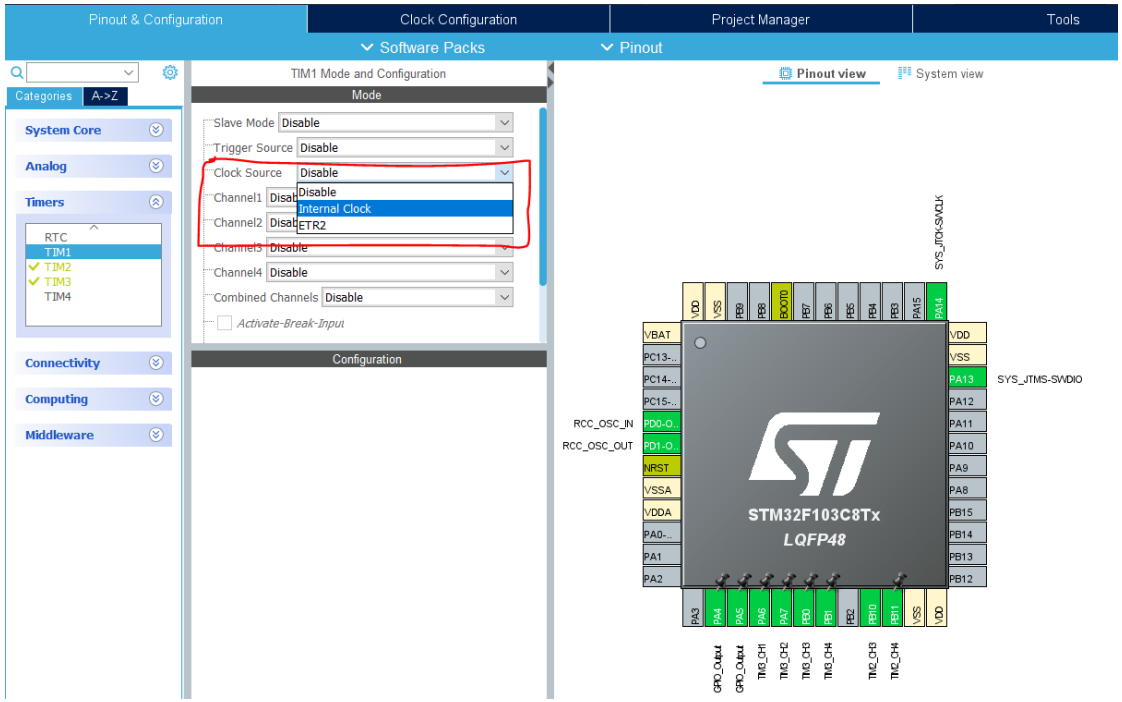
Setup clock cho Timer: Bật clock ở đây (Chú ý chưa cần set chân)



2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

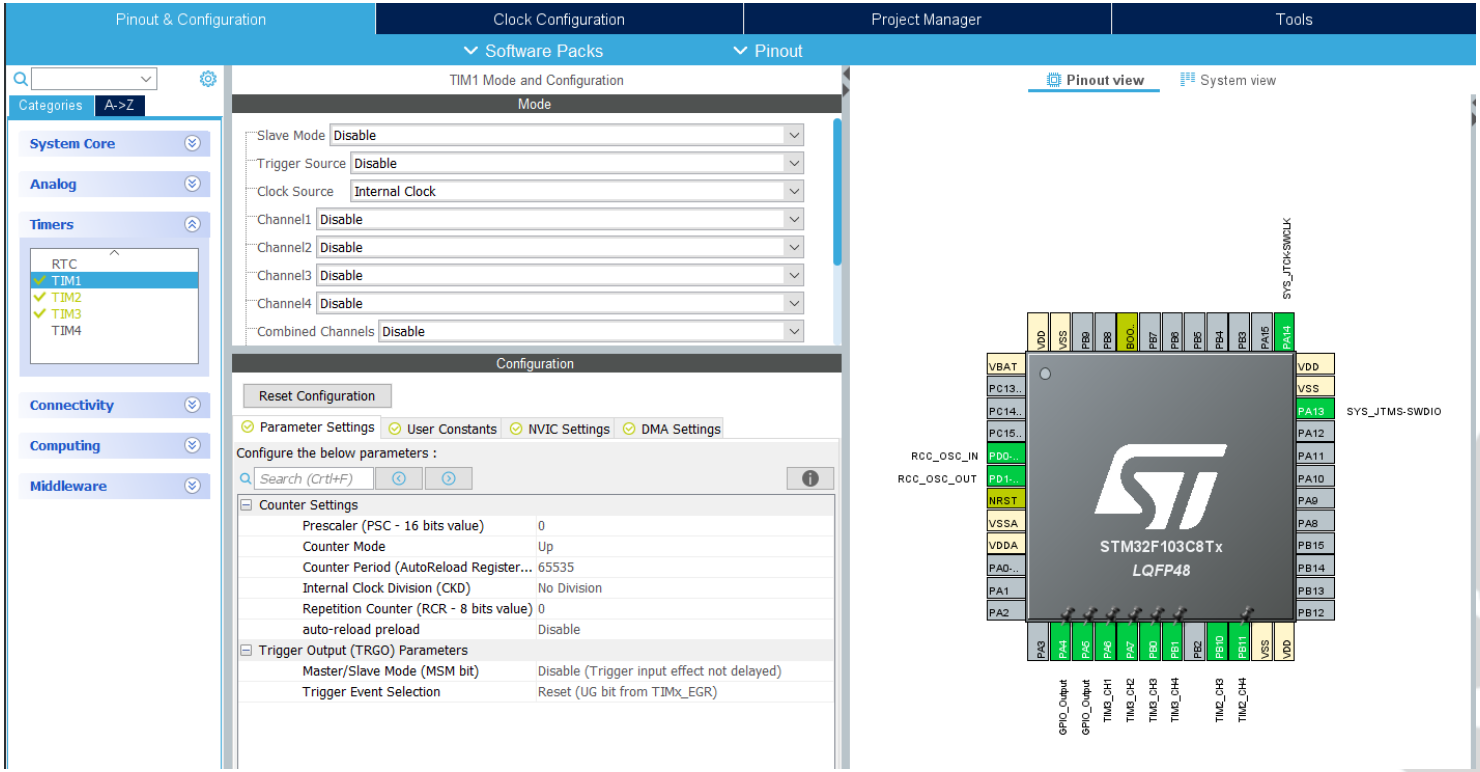
Setup clock cho Timer: Có nhiều chế độ, chúng ta sẽ dùng clock nội, tùy timer sẽ có nguồn clock khác nhau, thường là lấy từ APB1 và APB2, các bạn cần tham khảo reference manual để biết thêm



2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

Sau khi setup clock cho ngoại vi timer, chúng ta sẽ setup các thông số của timer:



2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

Sau khi setup clock cho ngoại vi timer, chúng ta sẽ setup các thông số của timer:

+ Prescaler: Bộ chia tần số clock timer.

+ Counter Mode: Chế độ đếm lên hay xuống của timer

+ Counter Period: Chu kì đếm của timer.

+ Internal Clock Division: Dành cho mode clock ngoại, chưa xét đến

+ Auto Reload Preload: Set up giá trị preload của thanh ghi ARR, điều khiển việc nạp lại giá trị ban đầu của timer counter.

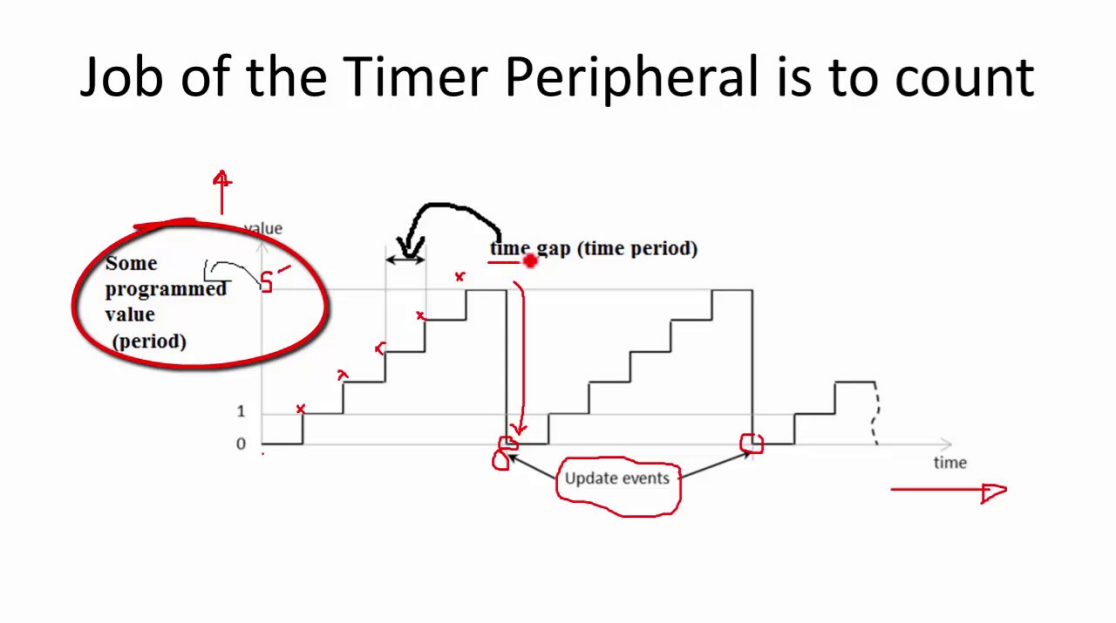
2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

+ Ở chế độ counter up, mỗi chu kì đếm lên sẽ kéo dài một khoảng thời gian là:

time_gap = clock / (pre_scaler + 1)

+ Sau khi đếm lên hết chu kì thì timer sẽ thông báo (set flag) để thông báo, hoặc người dùng có thể setup ngắt ngoài để biết timer đã chạy hết,

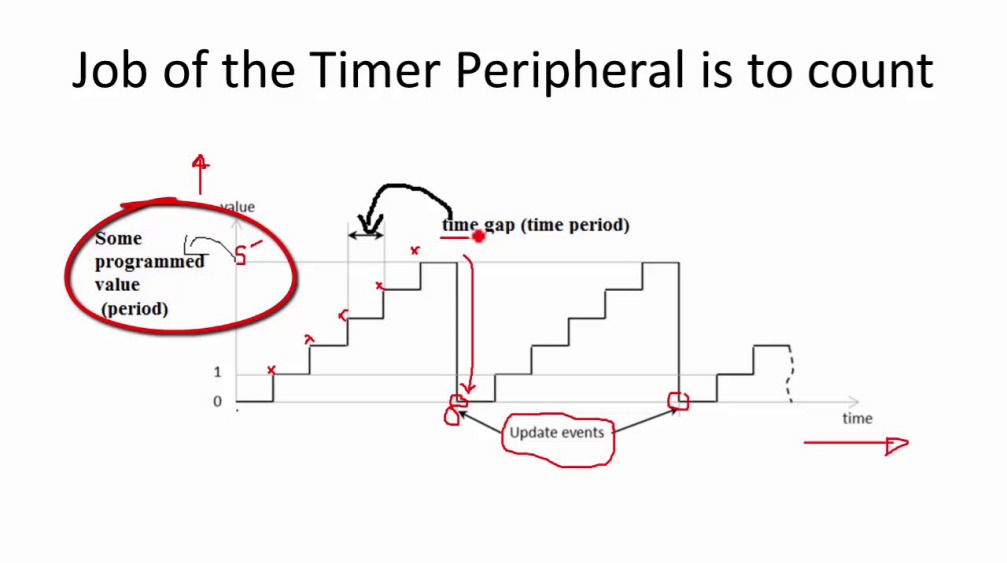


2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

+ Timer sẽ có một thanh ghi lưu giá trị counter, là thanh ghi CNT, giá trị của thanh ghi sẽ tang, hay giảm tùy theo chế độ hoạt động của timer.

+ Khi CNT tăng (giảm) đến giá trị Period lưu trong thanh ghi ARR (Auto reload) thì giá trị CNT được reset về giá trị ban đầu = 0 (đếm lên) hoặc ARR (đếm xuống), đồng thời bit ngắt Update Interrupt Timer được set lên cao để báo hiệu timer đếm xong 1 chu kì.



2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

+ Ví dụ: Để có timer count up với chu kì đếm là 1Hz, chu kì clock của timer là APB1, 36Mhz, yêu cầu period là 999, thì chọn pre_scaler như thế nào?

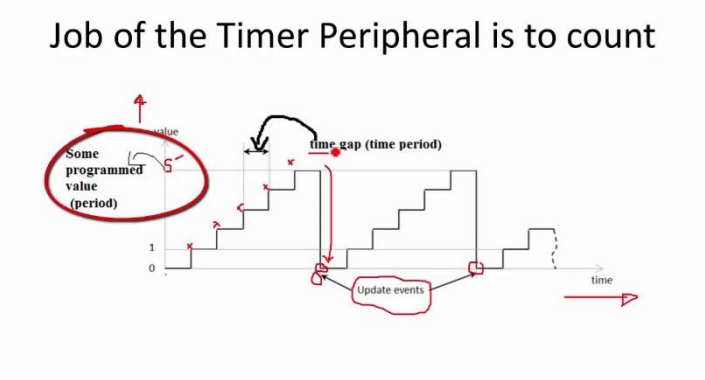
Giải:

+ Để có tần số 1Hz đếm lên thì thời gian thực hiện 1 chu kì là 1s, period là 999 + 1 (đếm từ 0)

=> time_gap = 1/1000 = 1(ms)

+ Tần số đầu vào của timer sẽ là f = 36*10^6 / (pre_scaler + 1)

=> time_gap = 1/f = (pre_scaler +1)/ (36*10^6) = 1 /1000 => pre_scaler = 3599



2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

+ Ví dụ: Để có timer count up với chu kì đếm là 1Hz, chu kì clock của timer là APB1, 36Mhz, yêu cầu pre_scaler là 35, thì chọn period như thế nào?

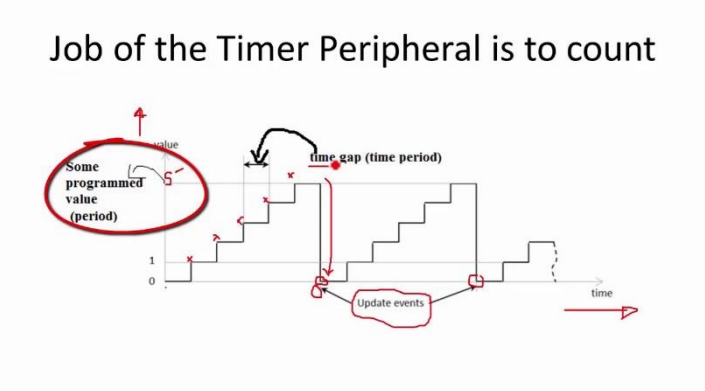
Giải:

+ Để có tần số 1Hz đếm lên thì thời gian thực hiện 1 chu kì là 1s, period là p

=> time_gap = 1/(p + 1) (s)

+ Tần số đầu vào của timer sẽ là f = 36*10^6 / (pre_scaler + 1)

=> time_gap = 1/f = (35 +1)/ (36*10^6) = 10^-6 =>1/(p+1) = 10^-6 => p = 10^6 - 1

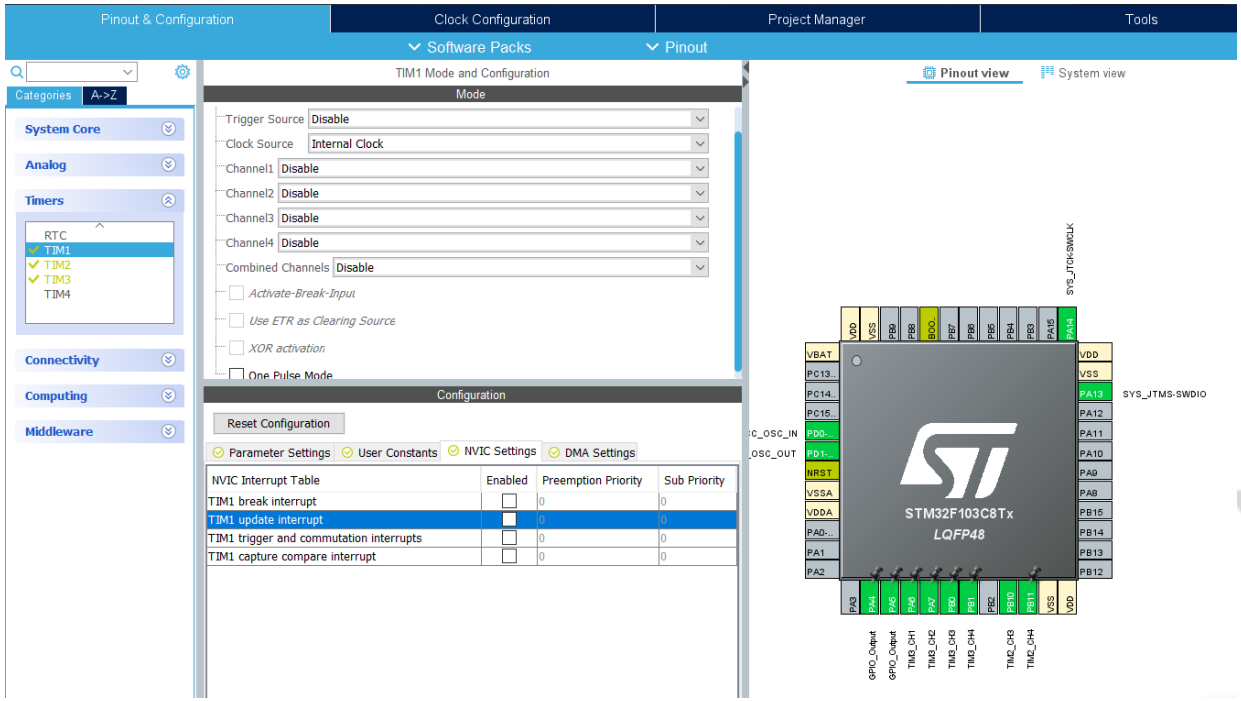


2. Các chế độ của timer

1.4 Các chế độ hoạt động của timer STM32

Ngắt timer và cách setup ngắt timer

+ Vì đặc điểm đơn giản của timer, chúng ta có thể setup ngắt timer để thực hiện xử lý task theo thời gian:



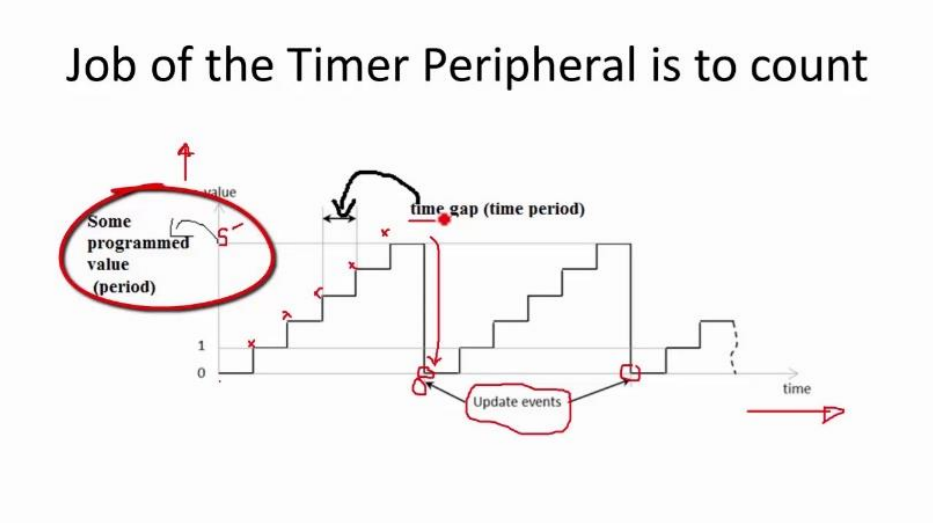
2. Các chế độ của timer

2.1 Chế độ Counter up

+ Ứng dụng:

+ Điều khiển một nhiệm vụ thực hiện theo thời gian một cách chính xác, ta có thể dùng delay để làm việc này, tuy nhiên thời gian delay CPU bị tạm ngưng, đây không phải là cách hay để thực hiện công việc

+ Tạo PWM, xung clock.

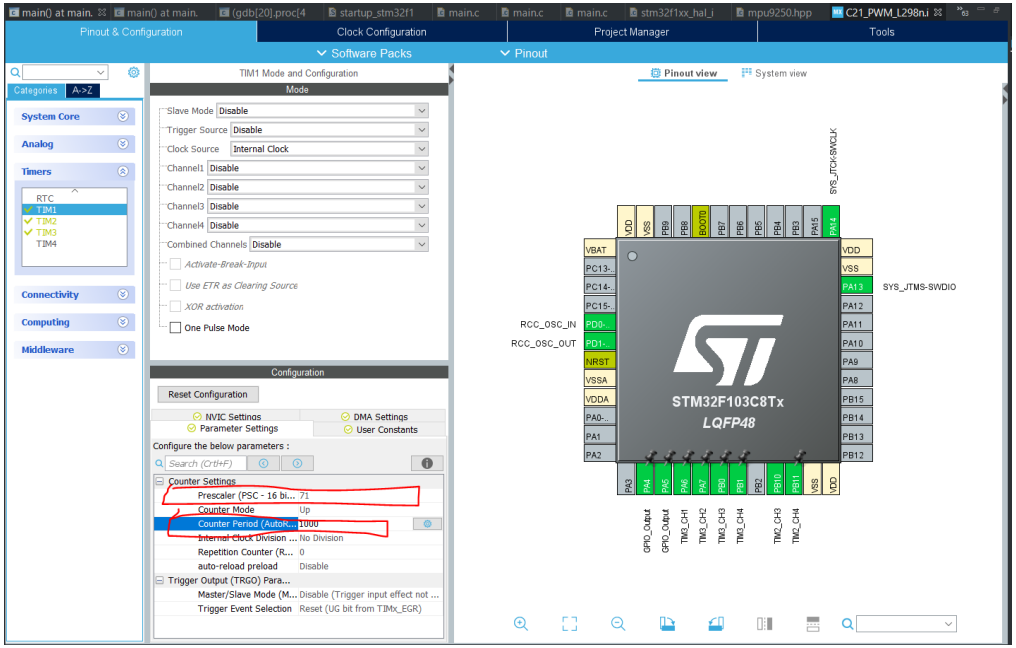


2. Các chế độ của timer

2.1 Chế độ Counter up

Cách set up:

+ Chọn các thông số cần thiết (Prescaler, Period, autoreload,…)

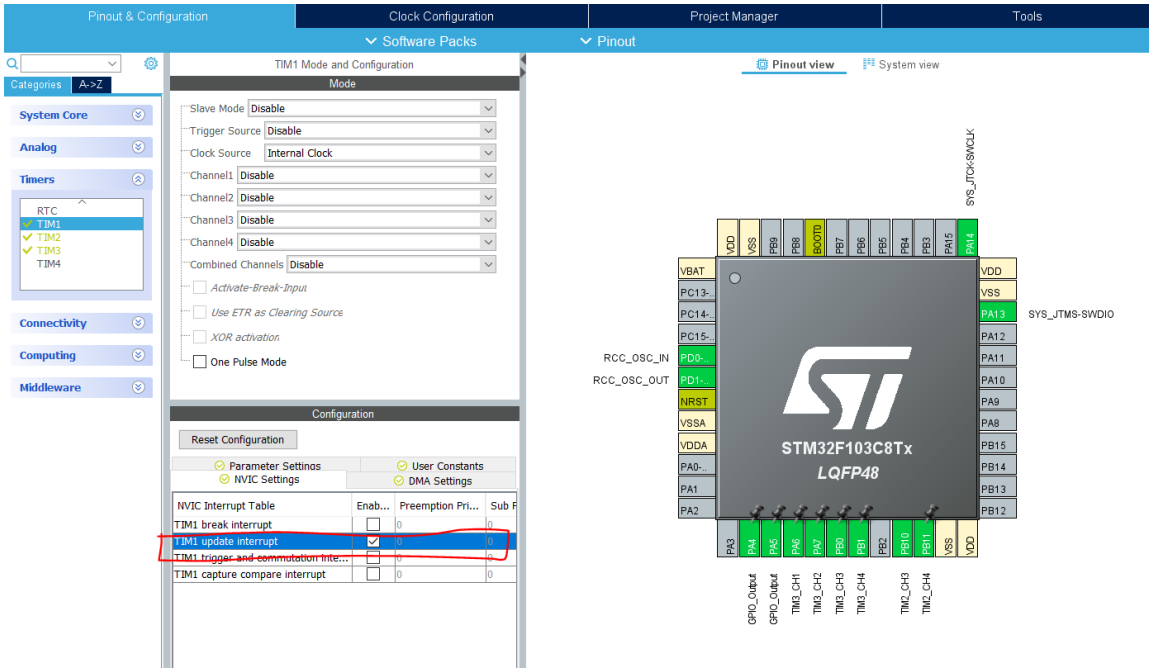


2. Các chế độ của timer

2.1 Chế độ Counter up

Cách set up:

+ Để chọn ngắt, có một số loại cờ ngắt như update, break,… trong đó cờ ngắt update xảy ra khi kết thúc một period



2. Các chế độ của timer

2.1 Chế độ Counter up

Sau khi đã setup, chúng ta sẽ bật timer trong code:

Code sinh ra như sau:

+ Biến quản lý timer:

```
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim1;
```

+ Hàm Setup timer:

```
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 71;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 1000;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */

}
```

2. Các chế độ của timer

2.1 Chế độ Counter up

Sau khi đã setup, chúng ta sẽ bật timer trong code:

Code sinh ra như sau:

+ Biến quản lý timer:

```
42 /* Private variables ---------------------------------------------------------*/
43 TIM_HandleTypeDef htim1;
```

+ Hàm Setup timer: Hàm này sẽ được gọi trong hàm main.

```
181 static void MX_TIM1_Init(void)
182 {
183
184   /* USER CODE BEGIN TIM1_Init 0 */
185
186   /* USER CODE END TIM1_Init 0 */
187
188   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
189   TIM_MasterConfigTypeDef sMasterConfig = {0};
190
191   /* USER CODE BEGIN TIM1_Init 1 */
192
193   /* USER CODE END TIM1_Init 1 */
194   htim1.Instance = TIM1;
195   htim1.Init.Prescaler = 71;
196   htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
197   htim1.Init.Period = 1000;
198   htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
199   htim1.Init.RepetitionCounter = 0;
200   htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
201   if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
202   {
203     Error_Handler();
204   }
205   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
206   if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
207   {
208     Error_Handler();
209   }
210   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
211   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
212   if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
213   {
214     Error_Handler();
215   }
216   /* USER CODE BEGIN TIM1_Init 2 */
217
218   /* USER CODE END TIM1_Init 2 */
219
220 }
```

2. Các chế độ của timer

2.1 Chế độ Counter up

Sau khi đã setup, chúng ta sẽ bật timer trong code:

Để bật timer, chúng ta sử dụng hàm sau:

HAL_StatusTypeDef HAL_TIM_Base_Start(TIM_HandleTypeDef *htim)

+Trong đó TIM_HandleTypeDef *htim là con trỏ chỉ đến địa chỉ của biến timer đã được sinh ở trước.

Để bật timer có ngắt:

HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)

2. Các chế độ của timer

2.1 Chế độ Counter up

Ví dụ: Nháy led theo chu kì 1s dùng timer:

Timer đếm lên, trong while ta kiểm tra CNT của timer, khi CNT có giá trị bằng ARR thì ta nháy led

```
 99   HAL_TIM_Base_Start(&htim1);
100
101
102   /* USER CODE END 2 */
103
104   /* Infinite loop */
105   /* USER CODE BEGIN WHILE */
106   while (1)
107   {
108     /* USER CODE END WHILE */
109
110     /* USER CODE BEGIN 3 */
111     if(htim1.Instance->CNT == htim1.Instance->ARR){
112         HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
113     }
114
115   }
116   /* USER CODE END 3 */
117 }
```

2. Các chế độ của timer

2.1 Chế độ Counter up

Nhận xét: Cách làm này tuy thực hiện được, nhưng không hay vì CPU phải liên tục đọc giá trị của CNT

```
 99   HAL_TIM_Base_Start(&htim1);
100
101
102   /* USER CODE END 2 */
103
104   /* Infinite loop */
105   /* USER CODE BEGIN WHILE */
106   while (1)
107   {
108     /* USER CODE END WHILE */
109
110     /* USER CODE BEGIN 3 */
111     if(htim1.Instance->CNT == htim1.Instance->ARR){
112         HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
113     }
114
115   }
116   /* USER CODE END 3 */
117 }
```

=> Ta cần ngắt để thực hiện các tác vụ này.

2. Các chế độ của timer

2.1 Chế độ Counter up

Nhận xét: Cách làm này tuy thực hiện được, nhưng không hay vì CPU phải liên tục đọc giá trị của CNT

```
 99   HAL_TIM_Base_Start(&htim1);
100
101
102   /* USER CODE END 2 */
103
104   /* Infinite loop */
105   /* USER CODE BEGIN WHILE */
106   while (1)
107   {
108     /* USER CODE END WHILE */
109
110     /* USER CODE BEGIN 3 */
111     if(htim1.Instance->CNT == htim1.Instance->ARR){
112         HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
113     }
114
115   }
116   /* USER CODE END 3 */
117 }
```

=> Ta cần ngắt để thực hiện các tác vụ này.

2. Các chế độ của timer

2.1 Chế độ Counter up

Ví dụ: Nháy led sử dụng ngắt

Ta bật timer ở chế độ ngắt:

```
99  HAL_TIM_Base_Start_IT(&htim1);
100
101
102  /* USER CODE END 2 */
103
104  /* Infinite loop */
105  /* USER CODE BEGIN WHILE */
106  while (1)
107  {
108    /* USER CODE END WHILE */
109
110    /* USER CODE BEGIN 3 */
111  }
112  /* USER CODE END 3 */
113 }
```

Hàm ngắt của timer, hàm này sẽ tự động được gọi khi có ngắt timer, do đó ta cần câu lệnh if để check có đúng timer này ngắt

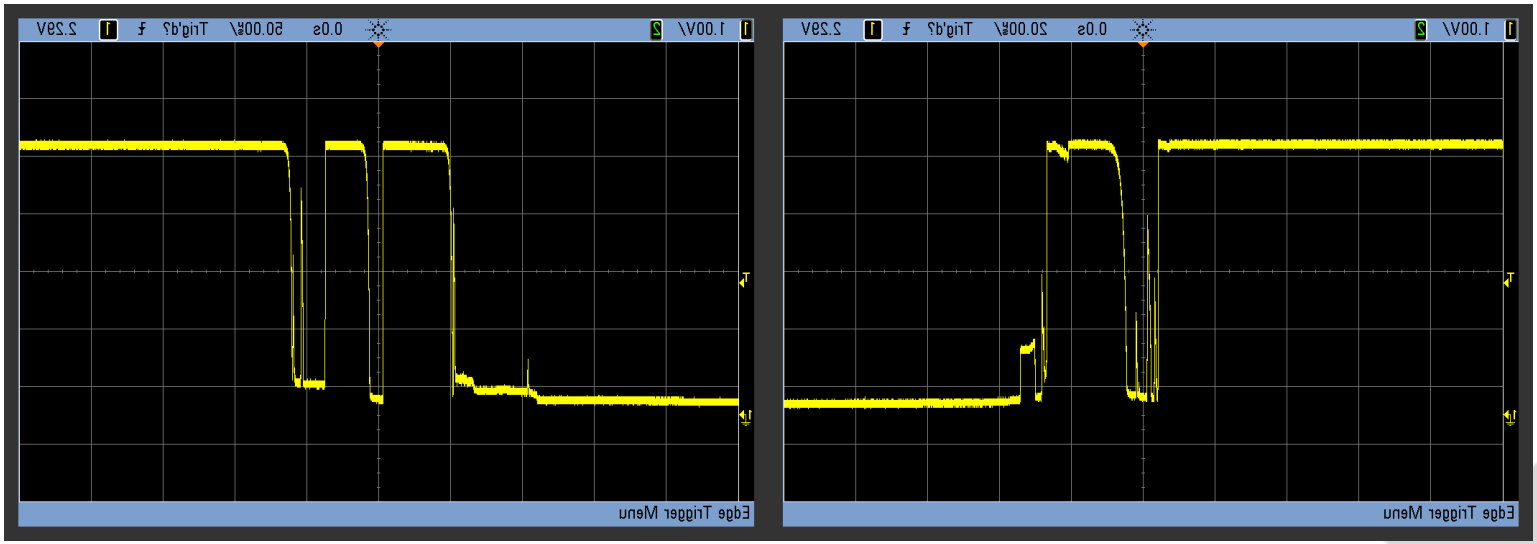
```
361 /* USER CODE BEGIN 4 */
362 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
363 {
364     if(htim->Instance == TIM1)
365     {
366         HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
367     }
368 }
```

2. Các chế độ của timer

2.1 Chế độ Counter up

Ví dụ: Chống rung button sử dụng timer:

Nhắc lại về rung button: Khi nhấn button, có khả năng sẽ có những "gai" điện áp trên button:



Để xử lý thì ta có thể đọc điện áp của button sau 1 thời gian khi điện áp đã ổn định, thực tế là khoảng 50ms (tất nhiên không ai bấm button nhanh hơn 50ms được ☺))

2. Các chế độ của timer

2.1 Chế độ Counter up

Ví dụ: Chống rung button sử dụng timer:

=> Trong ngắt ngoài button, việc sử dụng HAL_Delay là không nên, đồng thời việc thêm Delay này tốn thời gian CPU không cần thiết

⇒Giải pháp sử dụng timer:

Trong ngắt ngoài, khi có ngắt: Khi ngắt timer xảy ra (50ms sau), ta check lại trạng thái nút nhấn, nếu như trạng thái đã đổi, ta thực hiện nháy led

```
375 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
376 {
377       if ( GPIO_Pin == GPIO_PIN_13)
378           {
379               // Write your code here
380             HAL_TIM_Base_Start_IT(&htim4);
381
382             pin_selected = 3;
383
384           }
```

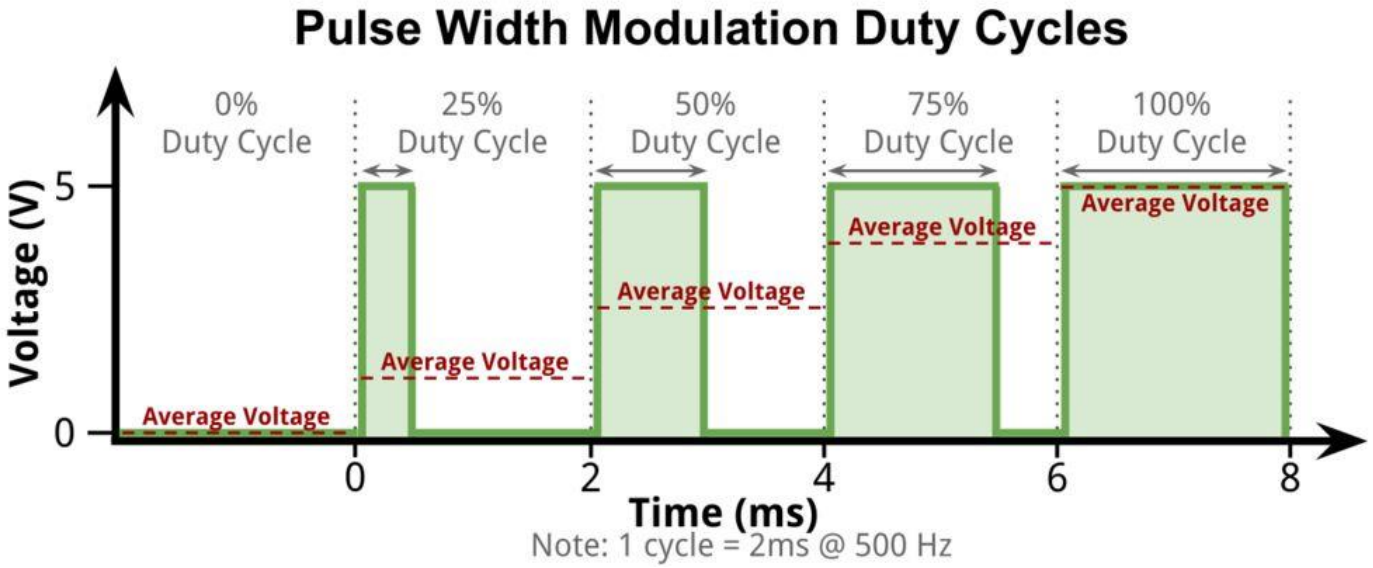
```
897 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
898 {
899   /* Prevent unused argument(s) compilation warning */
900   UNUSED(htim);
901
902   /* NOTE : This function should not be modified, when the cal
903             the HAL_TIM_PeriodElapsedCallback could be impleme
904    */
905   if(pin_selected == 3 ){
906     if(HAL_GPIO_ReadPin(GPIOC, 8192) == GPIO_PIN_SET){
907         // Do something
908         HAL_TIM_Base_Stop_IT(&htim4);
909     }
```

***Đây chỉ làm hàm tham khảo, chép lại không chạy thì tự hiểu nha :v**

2. Các chế độ của timer

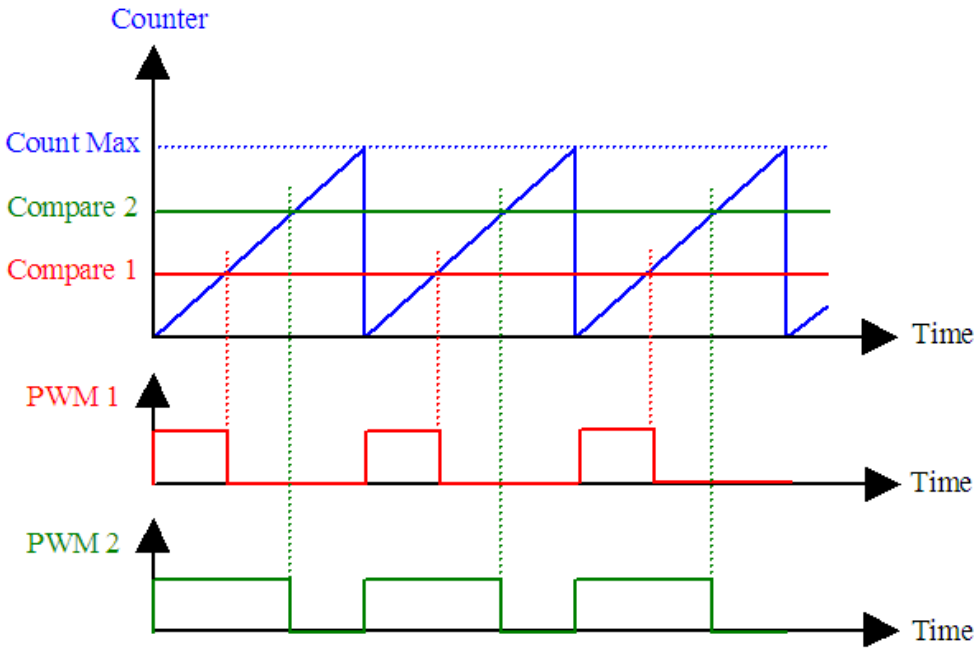
2. Chế độ PWM

- PWM (Pulse Width Modulation) là một cách điều khiển xung rất phổ biến trong nhiều thiết bị
- Nguyên tắc PWM là điều khiển độ rộng xung, điều này làm thay đổi điện áp trung bình => thay đổi công suất, từ đó ta có thể điều khiển độ sáng led, điều khiển động cơ,…..



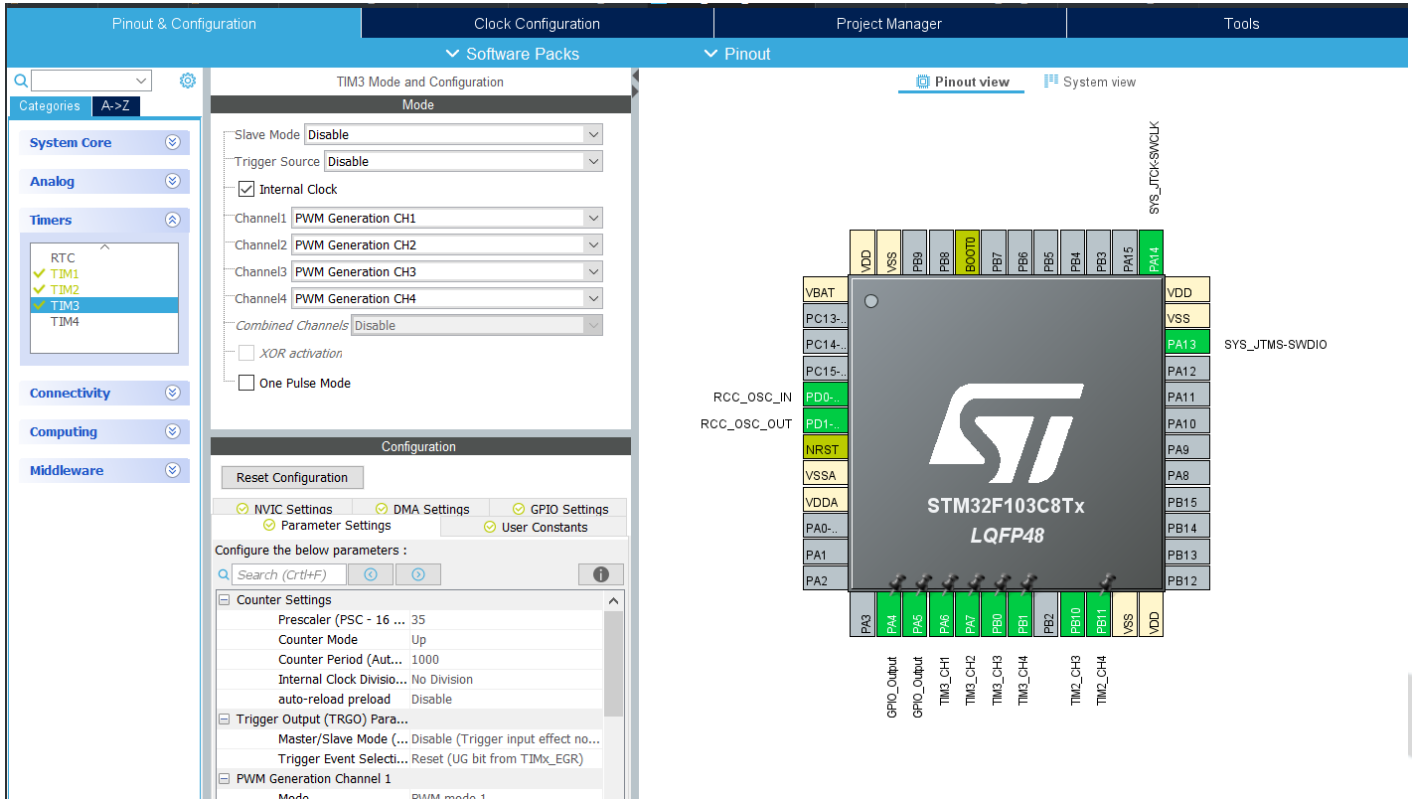
2. Các chế độ của timer

- PWM generator (Up mode)
- Mỗi timer sẽ có 4 thanh ghi so sánh (Compare Counter Register), khi timer đếm lên, thì khi CNT = CCRx thì chân PWM sẽ đổi trạng thái như hình:



2. Các chế độ của timer

- Cấu hình ngoại vi TIM3 xuất PWM ở 4 channels, trong đó trên board C21 có chân PA6 và PA7 nối đến led, các bạn có thể cắm jumper đẻ xem hiệu quả



2. Các chế độ của timer

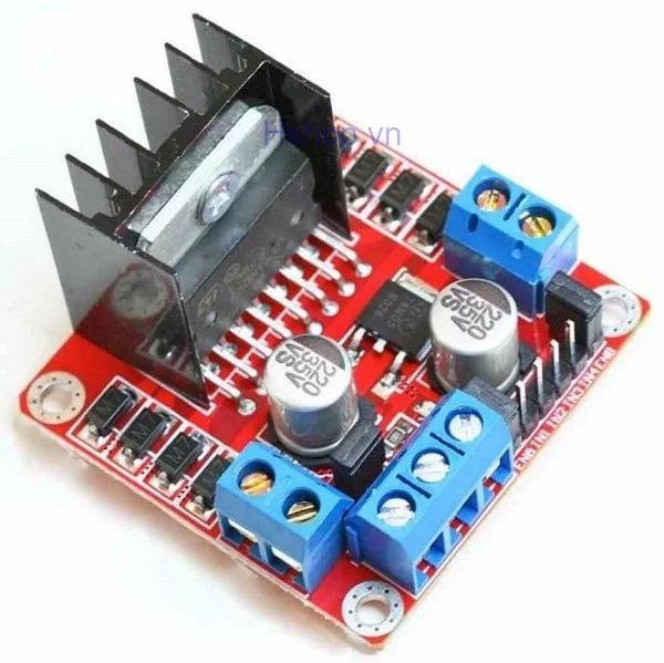
- Ta dùng Macro:

__HAL_TIM_SET_COMPARE(*htim, TIM_CHANNEL, duty) để thay đổi giá trị thanh ghi CCR, tức thay đổi chu kì PWM, chú ý duty < period

```
101   HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_1);
102   HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_2);
103   /* USER CODE END 2 */
104
105   /* Infinite loop */
106   /* USER CODE BEGIN WHILE */
107   while (1)
108   {
109     /* USER CODE END WHILE */
110
111     /* USER CODE BEGIN 3 */
112     for(int i = 0; i < 1000; i++){
113         __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1,i);
114         __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2,i);
115     }
116     for(int i = 1000; i > 0; i--){
117         __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1,i);
118         __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2,i);
119     }
120
121   }
122   /* USER CODE END 3 */
123 }
```

3. Ứng dụng timer

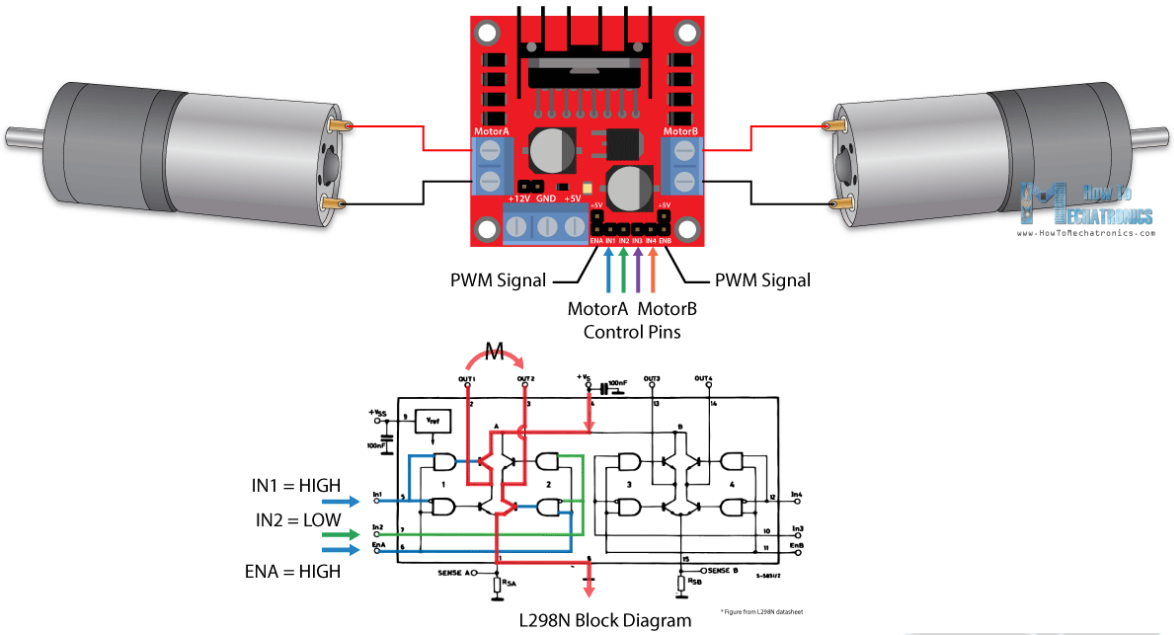
- **Ứng dụng PWM (Phần này đọc thêm)**
- Một số ứng dụng PWM:



L298 điều khiển động cơ DC

3. Ứng dụng timer

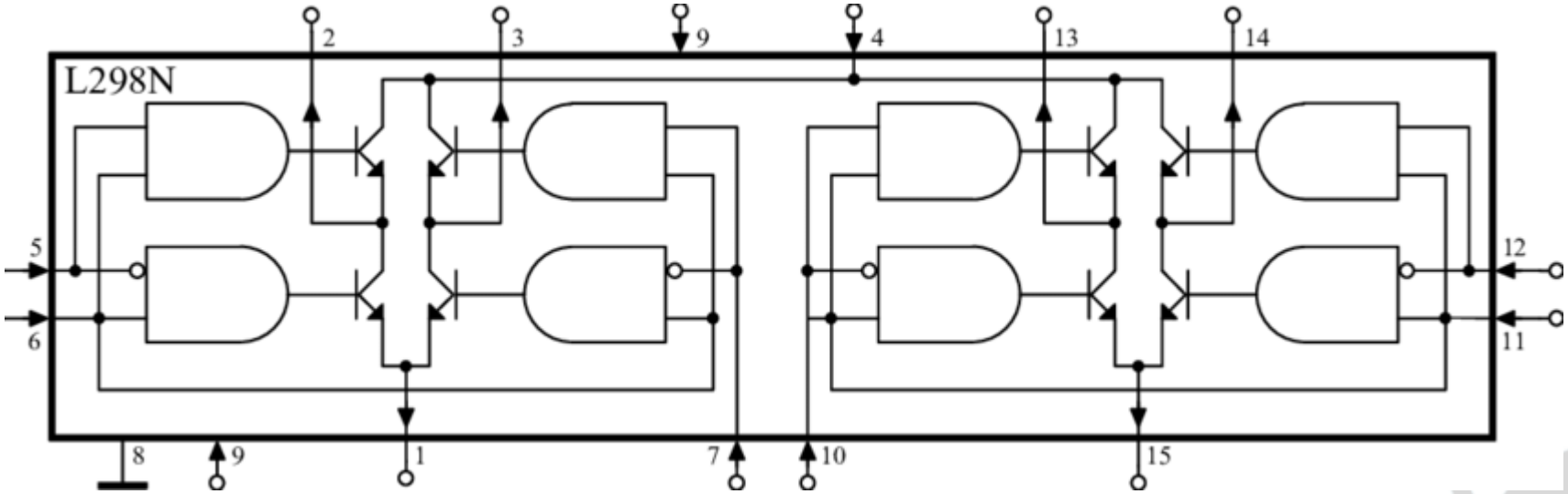
- **Ứng dụng PWM (Phần này đọc thêm)**
- **Để điều khiển PWM cho 2 động cơ, ta cần ít nhất là 4 chân điều khiển: Mỗi động cơ cần 1 chân hướng và 1 chân PWM**
- **Lưu ý: sơ đồ này thiếu nguồn cho L298N, cần cấp nguồn có điện áp 9-12v và có dòng xả đủ lớn, vì L298N gây sụt áp ~2v**



3. Ứng dụng timer

Sơ đồ nguyên lý L298N:

- L298N là module 2 cầu H, để điều khiển 1 động cơ ta vần 1 cầu H, bài này ta không đi sâu vào nguyên lý cầu H, tuy nhiên có thể tham khảo thêm ở đây:
- http://www.hocavr.com/2018/06/mach-cau-h.html

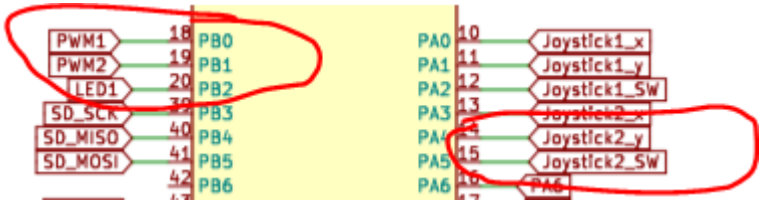


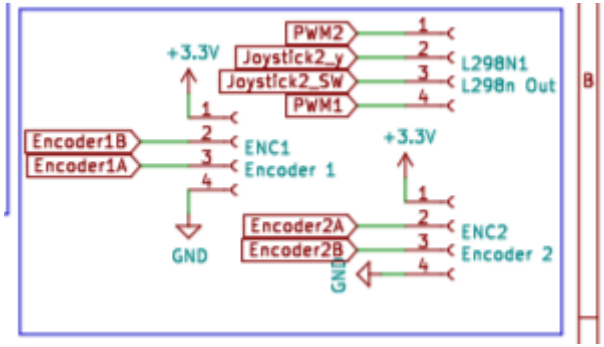
L298 điều khiển động cơ DC

3. Ứng dụng timer

Sử dụng L298N với board C21:

Board C21 có cổng ra cho L298N: trong đó chân PWM tương ứng với TIM2 channel 2 và 3, chân còn lại là chân chọn hướng cho cầu H





Code mẫu tham khảo:
https://github.com/DangLamTung/C21/tree/master/C21_Software/C21_Car

3. Ứng dụng timer

Sử dụng L298N với board C21:

Về cơ bản thì đây là code mẫu:

Hàm chạy theo 1 chiều:

```
15  void go_forward(float pulse_width)
16  {
17          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,GPIO_PIN_SET);
18          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4,GPIO_PIN_SET);
19          __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR0_CH, 999 - pulse_width);
20          __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR1_CH, 999 - pulse_width);
21  }
```

Hàm chạy theo chiều ngược lại

```
37  void go_backward(float pulse_width)
38  {
39          HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_RESET);
40          HAL_GPIO_WritePin(GPIOA,GPIO_PIN_4,GPIO_PIN_RESET);
41          __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR0_CH, pulse_width);
42          __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR1_CH, pulse_width);
43  }
```

3. Ứng dụng timer

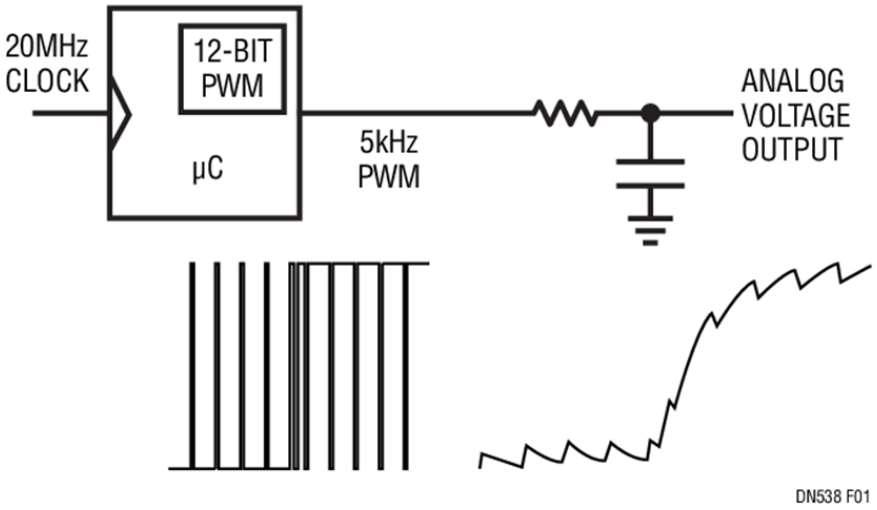
- **Ứng dụng chơi một bản nhạc :v**
- Vì sao người ta nghe được nhạc?
- Các nốt nhạc mà ta nghe được có các tần số của âm như sau:

	String Number (Note)					
Fret	1 (E)	2 (A)	3 (D)	4 (G)	5 (B)	6 ($\dot{E}$)
1	82.41	110.00	146.83	196.00	246.94	329.63
2	87.31	116.54	155.56	207.65	261.63	349.23
3	92.50	123.47	164.81	220.00	277.18	369.99
4	98.00	130.81	174.61	233.08	293.66	392.00
5	103.83	138.59	185.00	246.94	311.13	415.30
6	110.00	146.83	196.00	261.63	329.63	440.00
7	116.54	155.56	207.65	277.18	349.23	466.16
8	123.47	164.81	220.00	**293.66**	369.99	493.88
9	130.81	174.61	233.08	311.13	392.00	523.25
10	138.59	185.00	246.94	329.63	415.30	554.37
11	146.83	196.00	261.63	349.23	440.00	587.33
12	155.56	207.65	277.18	369.99	466.16	**622.25**
13	164.81	220.00	293.66	392.00	493.88	659.26
14	174.61	233.08	311.13	415.30	523.25	698.46
15	185.00	246.94	329.63	440.00	554.37	739.99
16	196.00	261.63	349.23	466.16	587.33	783.99
17	207.65	277.18	369.99	493.88	622.25	830.61
18	220.00	293.66	392.00	523.25	659.26	880.00
19	233.08	311.13	415.30	554.37	698.46	932.33
20	246.94	329.63	440.00	587.33	739.99	987.77

3. Ứng dụng timer

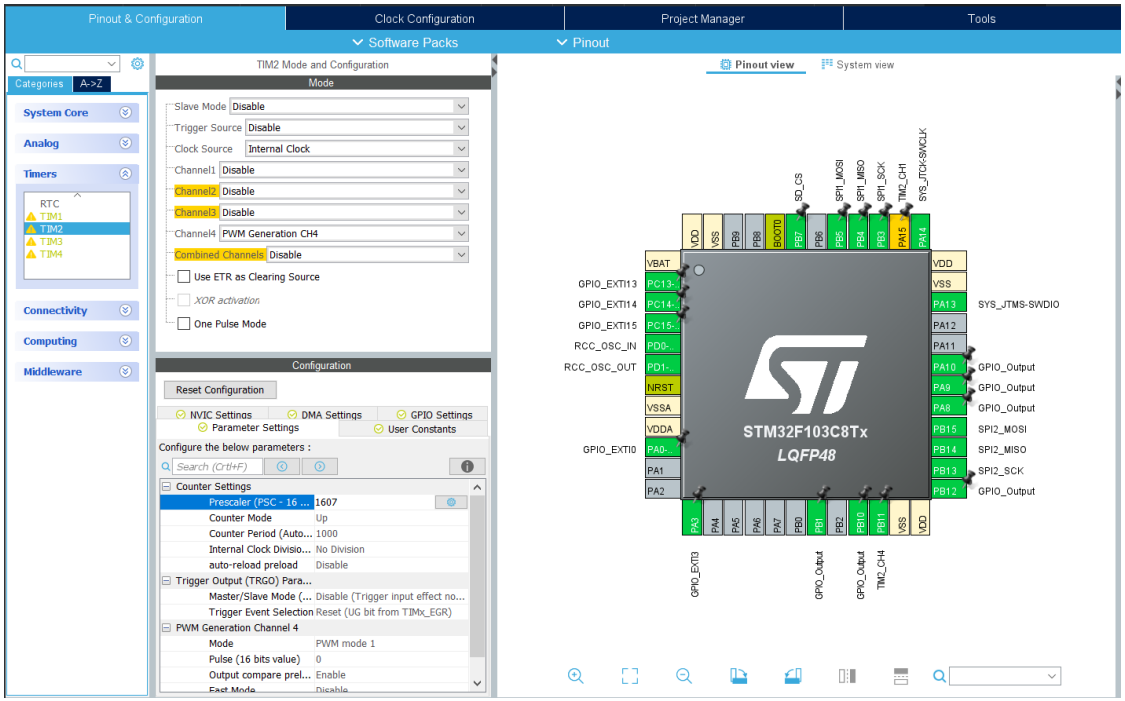
- Ứng dụng chơi một bản nhạc :v

- Tương tự, xung PWM với tần số phù hợp cũng sẽ gây ra hiệu ứng âm thanh tương tự, ta cần quan tâm đến tần số đóng ngắt của mạch, còn chu kì PWM ít có ảnh hưởng đến âm nghe được.
- Tất nhiên chất lượng âm thanh không thể tốt như analog, nhưng có thể sử dụng một mạch lọc thông thấp để cải thiện chất lượng âm:



3. Ứng dụng timer

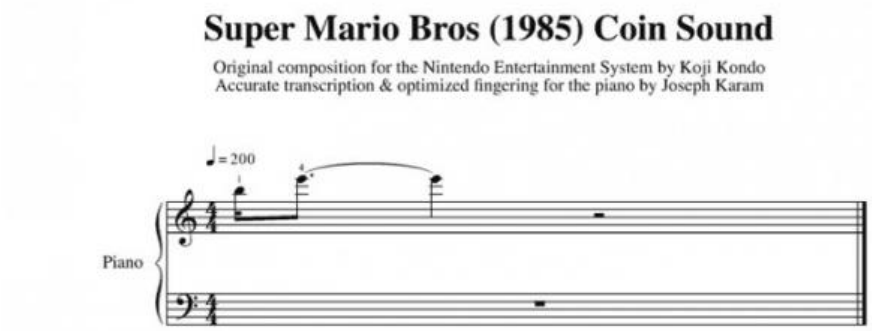
- **Ứng dụng chơi một bản nhạc :v**
- Ví dụ về việc chơi nốt nhạc: (Clock APB2 là 72Mhz)
- Tần số nốt = 10^6/(1607 + 1) = 622.22 Hz, tức nốt Rê thăng ở quãng tám thứ 5 của piano :v



3. Ứng dụng timer

- **Ứng dụng chơi một bản nhạc :v**
- Ví dụ về việc chơi nốt nhạc: (Clock APB2 là 72Mhz)
- Để thay đổi tần số ta thay đổi prescaler hoặc thay đổi counter period,
- Ví dụ thay đổi period: Đây là sound effect của tiếng rớt tiền trong game Mario:

```
42 uint16_t coin_effect[2] = { 506, 379};
```



Tần số là 10^6/509 =

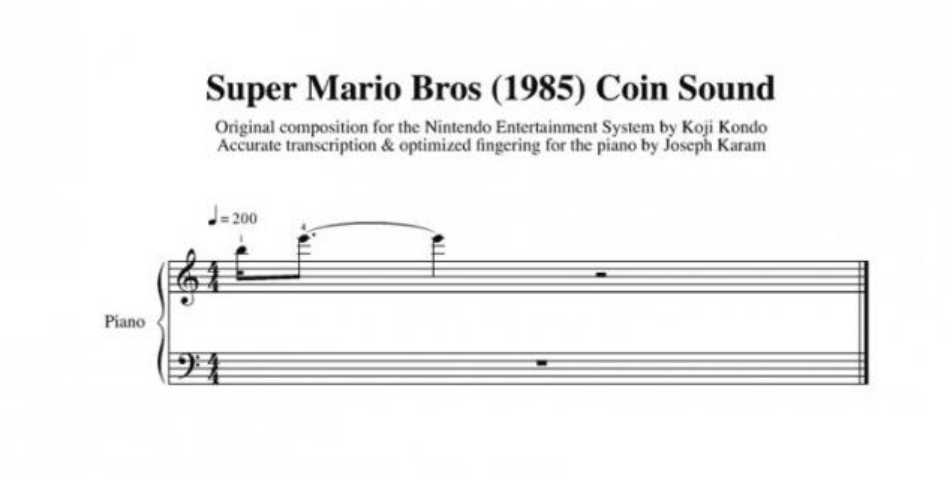
1964.636 (Hz),

Tức nốt Si ở quãng tám thứ 6, các bạn có thể

3. Ứng dụng timer

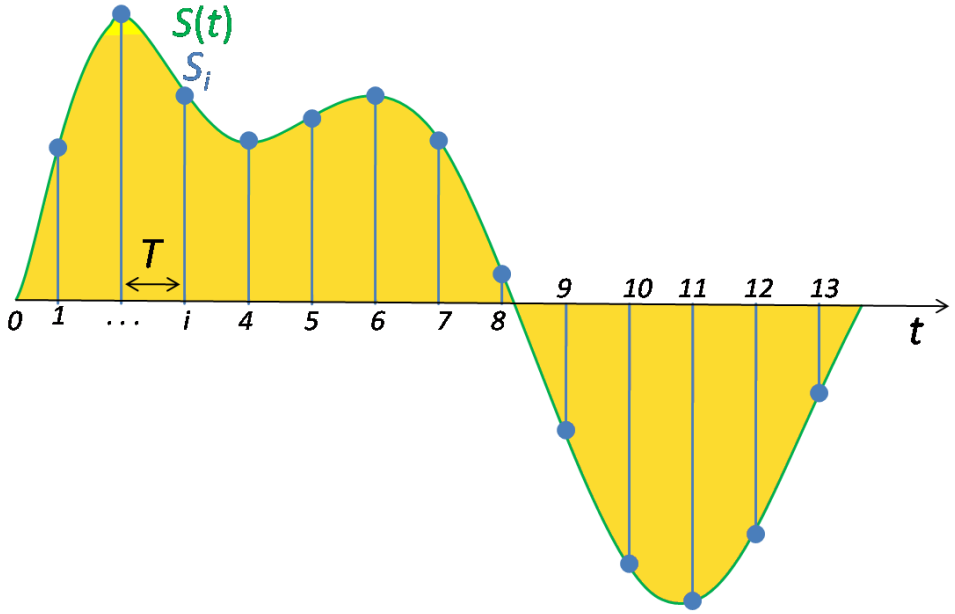
- **Ứng dụng chơi một bản nhạc :v**
- Hàm để thay đổi chu kì sẽ là:
-

```
__HAL_TIM_SET_AUTORELOAD(&htim2,coin_effect[0]);
```



3. Ứng dụng timer

- **Ứng dụng chơi một bản nhạc, nhưng full tiếng luôn nhé :v**
- Nguyên lý lấy mẫu âm thanh:
- Trong môn xử lý số tín hiệu, chắc các bạn cũng được học qua về phần lấy mẫu tín hiệu, đây là nền tảng cho các loại nhạc mà ta nghe được.



3. Ứng dụng timer

- Ứng dụng chơi một bản nhạc, nhưng full tiếng luôn nhé :v
- Micro sẽ sử dụng một bộ analog to digital (ADC) để lấy **cường độ** âm thanh tại một thời điểm n* (1/fs) với fs là tần số lấy mẫu âm thanh
- Như ta đã biết ở bài trên, PWM có tính chất đó là tạo ra giá trị **trung bình điện áp tại một thời điểm nào đó,** do đó ta hoàn toàn có thể tái tạo sóng âm với PWM

- **Theo nguyên lý Nyquist**

Tần số lấy mẫu tối thiểu

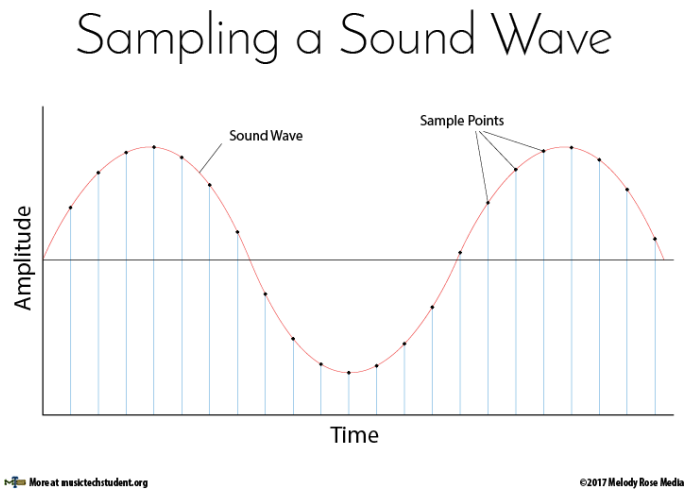
= 2 lần tần số âm, tức tần

Số lấy mẫu ít nhất là 40kHz

***Tuy nhiên nhiều khi lấy mẫu**

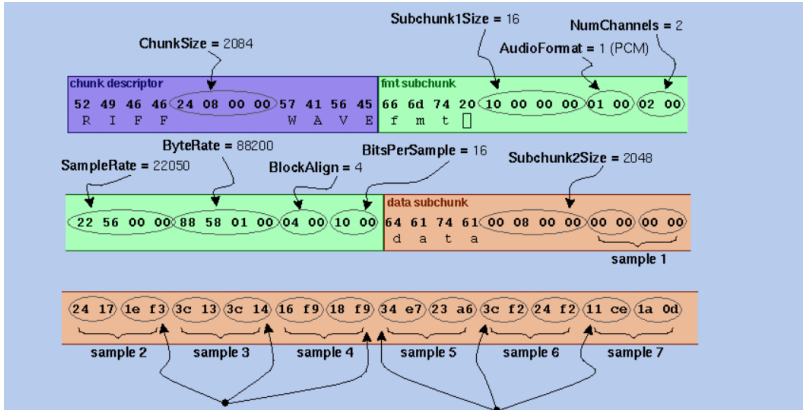
10kHz cũng được :v, vì tiếng nói

Tần số chỉ khoảng vài kHz



3. Ứng dụng timer

- File âm thanh WAV:
- Với các dòng nhạc thu âm lossless, chúng ta sẽ thu được các file WAV, tức file lấy mẫu cường độ âm thanh
- File WAV này thường rất lớn, và có chất lượng âm thanh tốt nhất



Ví dụ file WAV tần số lấy mẫu 22050 Hz, lấy mẫu ADC 16bit, 2 channel âm

3. Ứng dụng timer

- **File wav có kích thước khá lớn, bằng với:**

- **Số bit ADC * Tần số lấy mẫu * Thời gian**
- **Ví dụ: file 8bit * 12025 * 60 =** 721500 (bytes)
- **Tức 700kB !!!!, do đó ta cần các cách nén âm**

như MP3,…. Và bắt buộc dùng thẻ nhớ

Vì bài này khá dài nên code ở đây:

https://github.com/DangLamTung/C21/tree/master/C21_Software/C21_Music_Player

https://www.facebook.com/tung.danglam199/videos/2756594144621126


payitforward.edu.vn