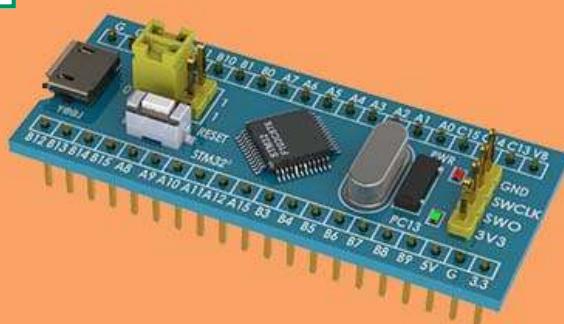
**LẬP TRÌNH STM32**

HAL_Delay với Systick và Delay us với Timer trên STM32

POSTED ON 11/08/2021 BY KHUÊ NGUYỄN

11
Th8**Khuê Nguyễn Creator**

Cách tạo Delay_us với Timer STM32 và cách hoạt động của Systick

HAL_Delay là một hàm có sẵn trong thư viện HAL của STM32. **Cách** hoạt động của HAL_Delay như thế nào và làm như thế nào để tạo một hàm Delay, cách sử dụng chúng như thế nào trong STM32. Tất cả sẽ được đề cập trong bài hôm nay.

Bài này nằm trong **Học STM32 từ A tới Z**

Mục Lục



1. Delay là gì?
2. Cách sử dụng Systick trong STM32
 - 2.1. Systick là gì?
 - 2.2. Sử dụng Systick trong HAL_Delay(ms)
 - 2.3. Sử dụng HAL_GetTick() code delay chế độ non-Blocking
3. Giải thích vì sao không nên dùng HAL_Delay trong ngắt
4. Delay us với Timer trong STM32
 - 4.1. Khởi tạo Timer trong CubeMX
 - 4.2. Lập trình Delay_us với Keil C
 - 4.3. Sử dụng thư viện Delay_Timer
5. Kết
 - 5.1. Related posts:

Delay là gì?

Delay trong tiếng anh là sự chậm trễ, sự trì hoãn. Trong lập trình cũng tương tự như vậy, khi **vi điều khiển** làm xong 1 việc chúng sẽ nhảy ngay tới việc tiếp theo chỉ trong một vài micro giây (tùy vào tần số hoạt động của MCU).

Để duy trì trạng thái đó 1 khoảng thời gian, chúng ta dùng delay. Khi đó MCU sẽ không làm gì cả (thực ra là vẫn làm nhưng là chạy lòng vòng trong hàm delay) khi thực hiện xong hàm delay mới thực hiện tiếp lệnh tiếp theo.

Cách sử dụng Systick trong STM32

Systick là gì?

STM32 có một bộ định thời hệ thống (System Timer) 24bit đếm xuống, tự nạp lại. Nghĩa là mỗi khi đếm đến 0 sẽ tự động quay về đếm từ số lớn nhất. Giá trị nạp lại sẽ được cấu hình trong thanh ghi SysTick Reload Value Register (SYST_RVR) của STM32.

Systick là hàm xử lý mỗi khi xảy ra sự kiện ngắt (thanh ghi counter của System Timer đếm về 0). Trong hàm này giá trị của biến đếm uwTick sẽ được tăng lên 1

(hoặc 1 số bất kì) bởi hàm incTick.

Mặc định khi khởi tạo code với CubeMx. Ngắt Systick xảy ra mỗi 1 mili giây.

```

184 void SysTick_Handler(void)
185 {
186     /* USER CODE BEGIN SysTick_IRQn 0 */
187
188     /* USER CODE END SysTick_IRQn 0 */
189     HAL_IncTick();
190     /* USER CODE BEGIN SysTick_IRQn 1 */
191
192     /* USER CODE END SysTick_IRQn 1 */
193 }
```

Hàm Xử lý ngắt Systick

```

284 /**
285  * @brief This function is called to increment a global variable "uwTick"
286  *        used as application time base.
287  * @note In the default implementation, this variable is incremented each 1ms
288  *        in SysTick ISR.
289  * @note This function is declared as __weak to be overwritten in case of other
290  *        implementations in user file.
291  * @retval None
292  */
293 __weak void HAL_IncTick(void)
294 {
295     uwTick += uwTickFreq;
296 }
297
298 /**

```

Hàm tăng Tick

Sử dụng Systick trong HAL_Delay(ms)

Trong hàm HAL_Delay, khi bắt đầu vào hàm giá trị ban đầu của Systick counter sẽ được đọc bằng hàm HAL_GetTick(); Trong while sẽ so sánh giá trị của System Timer tại thời điểm đó với thời điểm ban đầu, nếu lớn hơn hoặc bằng giá trị delay sẽ thoát vòng lặp và kết thúc hàm delay

```

349 /*/
350 * @brief This function provides minimum delay (in milliseconds) based
351 *       on variable incremented.
352 * @note In the default implementation , SysTick timer is the source of time base.
353 *       It is used to generate interrupts at regular time intervals where uwTick
354 *       is incremented.
355 * @note This function is declared as __weak to be overwritten in case of other
356 *       implementations in user file.
357 * @param Delay specifies the delay time length, in milliseconds.
358 * @retval None
359 */
360 __weak void HAL_Delay(uint32_t Delay)
361 {
362     uint32_t tickstart = HAL_GetTick();
363     uint32_t wait = Delay;
364
365     /* Add a freq to guarantee minimum wait */
366     if (wait < HAL_MAX_DELAY)
367     {
368         wait += (uint32_t)(uwTickFreq);
369     }
370
371     while ((HAL_GetTick() - tickstart) < wait)
372     {
373     }
374 }
```

Khi sử dụng hàm HAL_Delay, code của chúng ta sẽ chạy ở chế độ blocking mode, nghĩa là chúng không làm gì cả trong suốt thời gian delay(Thực tế là nó đi kiểm tra cái điều kiện while đó). Khi làm xong HAL_Delay thì mới tiếp tục làm những việc khác.

Ưu điểm của phương pháp này: Thời gian delay chính xác

Nhược điểm: Vì code ở chế độ Blocking thế nên có thể làm chậm các tác vụ khác. Ví dụ quét màn hình, hiển thị led 7

Sử dụng HAL_GetTick() code delay chế độ non-Blocking

Phương pháp này cũng gần giống như HAL_Delay chỉ khác là chúng ta sẽ sử dụng **if** thay cho **while**. Cấu trúc như thế này:

```

while(1)
{
    static int last_time = HAL_GetTick();
    int current_time = HAL_GetTick();
    if(current_time - last_time >= delay)
    {
        HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_0);
        last_time = current_time;
    }
}
```

```
//do something
}
```

Giải thích:

Chúng ta tạo ra 2 biến lưu giá trị counter cuối cùng (hay giá trị bắt đầu delay) và giá trị hiện tại của counter.

Mỗi khi code đi qua sẽ cập nhật giá trị của current time. Nếu giá trị hiện tại – giá trị bắt đầu = thời gian delay, chương trình sẽ thực hiện đảo trạng thái pin PA0. Sau đó lưu giá trị bắt đầu = giá trị hiện tại.

Phương pháp này thường sử dụng trong các việc cần lặp lại liên tục và tuần tự.

Ưu điểm: Chương trình sẽ làm được những việc khác mà không cần phải chờ đợi.

Nhược điểm: Thời gian delay có thể bị sai vì code thực hiện các câu lệnh dài, không kịp quay về kiểm tra biến current time.

Cả 2 phương pháp sử dụng Systick đều chỉ code được các hàm delay ở Mili giây, vậy để code hàm delay ở micro giây chúng ta cần làm như thế nào. Hãy tiếp tục đọc nhé

Giải thích vì sao không nên dùng HAL_Delay trong ngắt

Giả sử ta có một ngắt A có độ ưu tiên ngang bằng hoặc hơn so với ngắt Systick, khi ngắt A được gọi và có hàm HAL_Delay() trong phần xử lý ngắt của A.

Khi đó chương trình sẽ chạy vào HAL_Delay, sau khi biến tickstart lấy được giá trị trả về của hàm GetTick, chương trình chạy vào while, tuy nhiên ngắt Systick lúc này đang ở chế độ chờ (do ngắt A chưa chạy xong) như vậy, hàm GetTick không thể tăng được giá trị trả về, dẫn tới treo tại while.

Để khắc phục hiện tượng này chúng ta có thể giảm ưu tiên của ngắt A hoặc tăng ưu tiên của ngắt Systick, lúc này khi vào HAL_Delay, ngắt Systick sẽ đè lên

ngắt A, và chương trình vẫn hoạt động bình thường

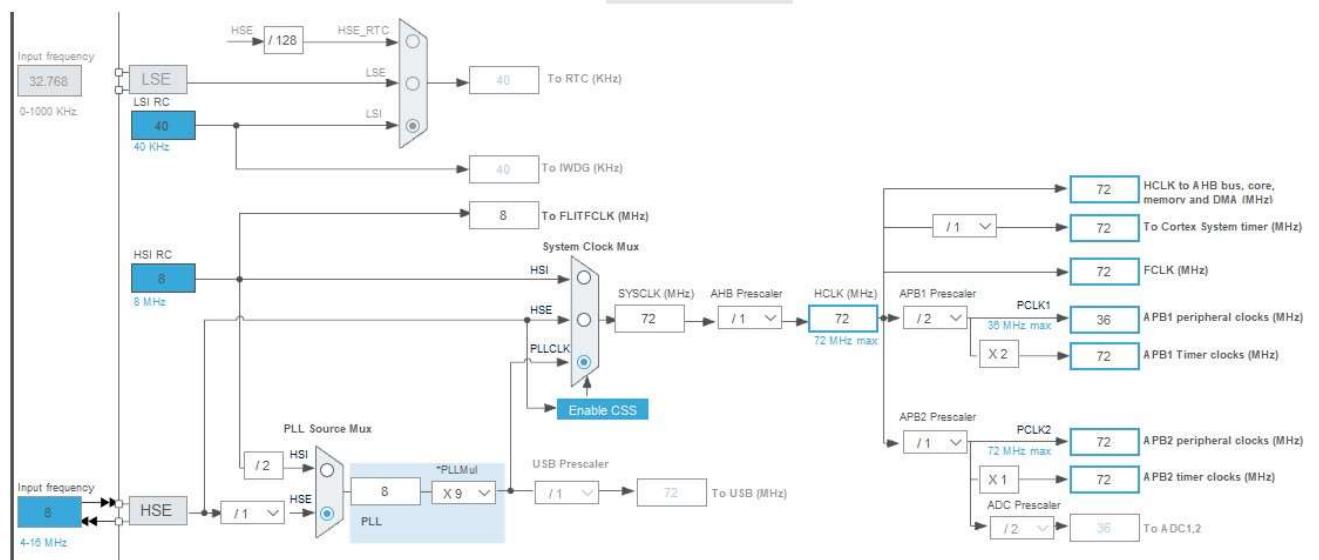
Delay us với Timer trong STM32

Nếu các bạn chưa biết Timer là gì và cách khởi tạo Timer thì vui lòng đọc lại bài: [Lập trình STM32 Timer](#) nhé!

Ý tưởng: Chúng ta sẽ tạo ra một Timer với mỗi lần đếm (Count Step) là 1 us. Đếm từ 0 tới 0xFFFF và lặp lại liên tục. Sau đó viết một hàm giống HAL_Delay để tạo delay chế độ blocking. Với non-Blocking thì không nên sử dụng vì chắc chắn sẽ sai thời gian (do mỗi lệnh của vdk xử lý đã mất vài us rồi).

Khởi tạo Timer trong CubeMX

fmaster Clock chúng ta sử dụng trong bài này là 72Mhz, nguồn thạch anh ngoài. Nên sẽ setup RCC là Crystal

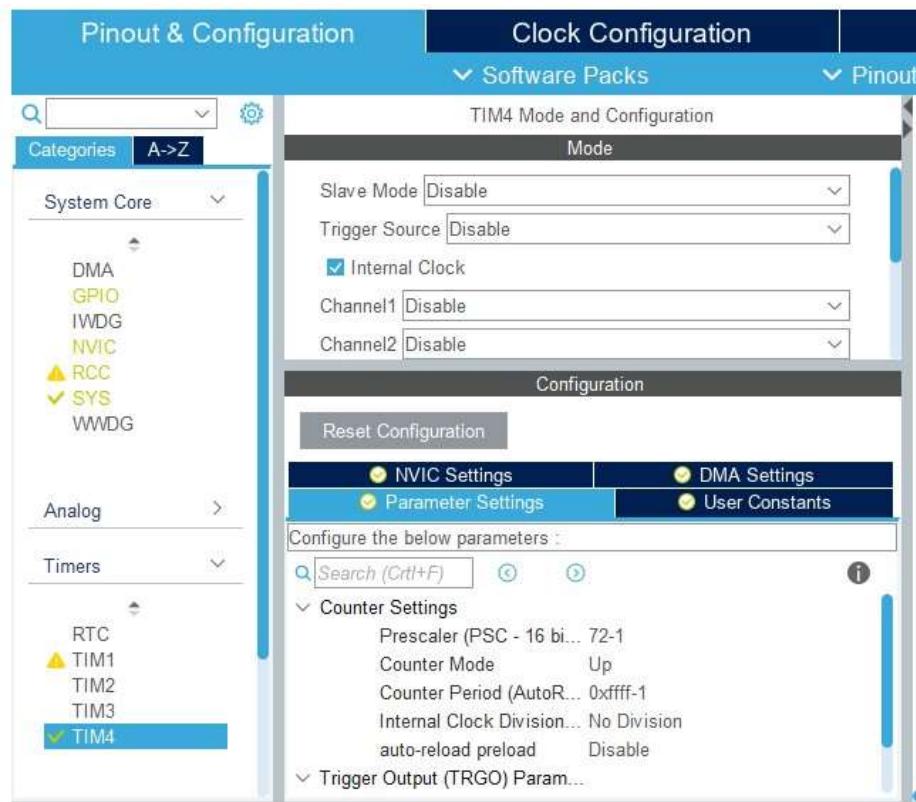


Tiếp đó chính Clock lên 72Mhz

Timer sử dụng Timer 2. Cấu hình như sau:

- Bộ chia 72-1: do prescale luôn được cộng thêm 1 với bất kì số nào được đưa vào. Với fmaster là 72Mhz, thì Count step sẽ là 1Mhz, tương ứng với 1us.
- Bộ nạp lại **ARR** sẽ để max là 0xffff - 1 : vì đếm từ 0 lên mà

- Chọn chế độ đếm là đếm lên



Gen code và mở bằng Keil C

Lập trình Delay_us với Keil C

Sử dụng `HAL_TIM_Base_Start(&htim2);` để khởi động Timer 2

Tạo hàm delay_us như sau:

```
void delay_us (uint16_t us)
{
    __HAL_TIM_SET_COUNTER(&htim2,0); // set the counter value a 0
    while (__HAL_TIM_GET_COUNTER(&htim2) < us); // wait for the counter to reach
}
```

Giải thích: Mỗi lần gọi delay, chúng ta sẽ xóa counter về 0. Sau đó chờ trong while tới khi counter bằng giá trị delay thì mới thoát ra.

Thực hành tạo xung với delay_us.

Khởi tạo PC13 là Output. Sau đó trong while như sau:

```
while (1)
{
    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
    delay_us(10);
}
```

Sử dụng Oscillo đo xung nhận được chúng ta sẽ được kết quả.

Sử dụng thư viện Delay_Timer

Các bạn download thư viện delay theo hướng dẫn: [Download tài liệu lập trình STM32](#)

Add file .c vào project và chỉnh đường dẫn thư viện. Cái này làm nhiều quá rồi nên mình không nhắc lại nữa nhé!

Trong main() chúng ta khởi tạo Timer cho delay bằng hàm `DELAY_TIM_Init(&htim2)`

Sau đó dùng hàm `DELAY_TIM_Us(&htim2, time_delay);` để tạo trễ nhé

Kết

Hal_Delay và Delay_us được sử dụng rất phổ biến trong [lập trình](#) nhúng trên STM32. Hãy hiểu rõ và sử dụng chúng một cách linh hoạt để tránh việc bị treo hoặc chậm chương trình nhé.

Nếu cảm thấy bài viết này có ích hãy chia sẻ với bạn bè nhé. Đừng quên ra nhập hội những anh em [Nghiên cứu lập trình](#) nhé!!

Rate this post

Related Posts:

1. [Hướng dẫn download và sử dụng tài liệu Lập trình STM32](#)
2. [Bài 14: Sử dụng STM32 IWDG Independent Watchdog Timer chống treo vi điều khiển](#)
3. [Bài 10: Giao thức I2C, lập trình STM32 với module RTC DS3231](#)
4. [Bài 7: STM32 Timer chế độ PWM](#)
5. [Bài 6: STM32 Timer chế độ Input Capture và Output Compare](#)
6. [Bài 2: Tổng quan về KIT STM32F103C8T6 Blue Pill](#)



KHUÊ NGUYỄN

Chỉ là người đam mê điện tử và lập trình. Làm được gì thì viết cho anh em xem thôi. :D

Trả lời

Email của bạn sẽ không được hiển thị công khai. Các trường bắt buộc được đánh dấu *

Bình luận *

Tên *

Email *

Trang web

PHẢN HỒI

Fanpage

 Khuê Nguyễn Creator - Họ...
2.754 lượt thích

Đã thích **Chia sẻ**

Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển
khoảng một tháng trước

Lý do thời gian gần đây mình không viết bài và làm thêm gì cả là đây 😊) Chính thức ra mắt sản phẩm định vị thông minh vTag.

Đây là một sản phẩm định vị đa năng với 3 công nghệ định vị WIFI, GPS, LBS kết hợp với sóng NB-IOT dành riêng cho các sản phẩm IOT.

Chỉ với 990.000đ chúng ta đã có thể có sản phẩm để:

- Định vị trẻ em, con cái... [Xem thêm](#)



Bài viết khác

Lập trình 8051 - AT89S52



Bài 1: Tổng quan về 8051 và chip AT89S51 - 52



Khuê Nguyễn Creator



Tổng quan về 8051

8051 là một dòng chip nhập môn cho lập trình viên nhúng, chúng được sử...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator

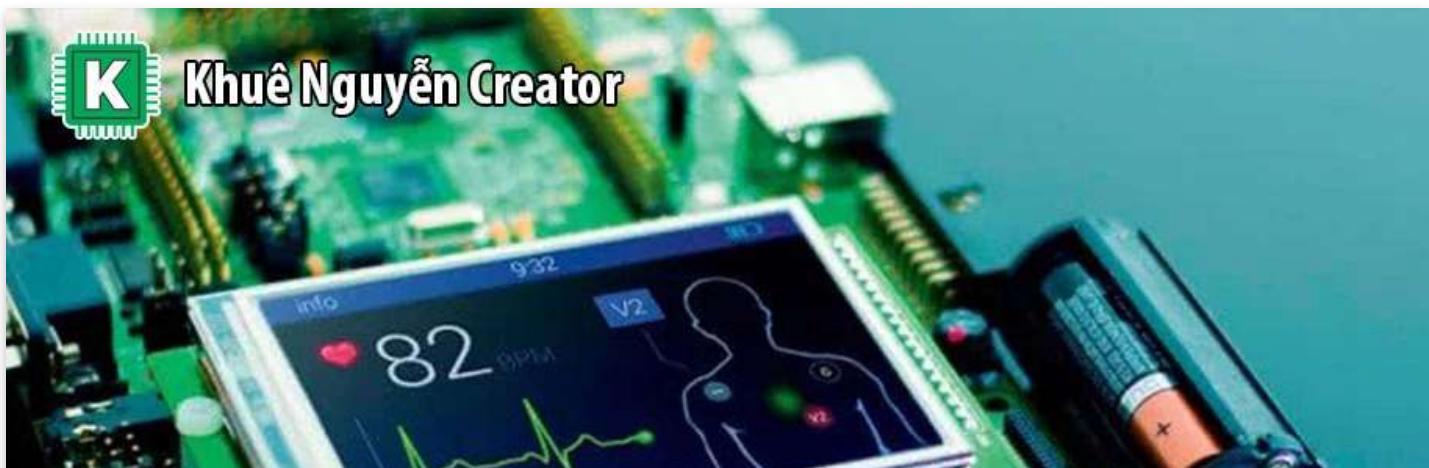


Lập trình STM32 HID Host giao tiếp với chuột và bàn phím

Lập trình STM32 USB HID Host giao tiếp với chuột và bàn phím máy tính

Trong bài này chúng ta sẽ cùng học STM32 HID Host, biến STM32 giống như...

[ĐỌC THÊM](#)





Lộ trình học lập trình nhúng từ A tới Z

Lập trình nhúng là một ngành có cơ hội nhưng cũng đòi hỏi nhiều kiến...

3 COMMENTS

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX





Khuê Nguyễn Creator




Lập trình STM32F407 SDIO đọc dữ liệu thẻ nhớ

Lập trình STM32 SDIO đọc ghi dữ liệu vào thẻ nhớ SD card

Trong bài này chúng ta cùng học cách lập trình STM32 SDIO, một chuẩn giao...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Lập trình STM32F407 DAC chuyển đổi số sang tương tự

Lập trình STM32 DAC tạo sóng hình Sin trên KIT STM32F407 Discovery

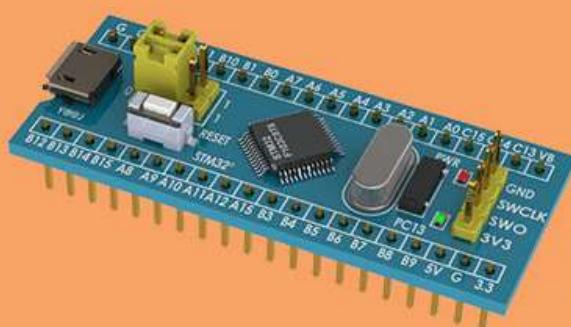
Trong bài này chúng ta sẽ cùng nhau tìm hiểu STM32 DAC với KIT STM32F407VE...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Sử dụng hàm printf để in Log khi Debug trên STM32

Hướng dẫn sử dụng printf với STM32 Uart để in Log trên Keil C

Trong bài này chúng ta sẽ học cách retarget hàm printf của thư viện stdio...

3 COMMENTS

[ĐỌC THÊM](#)

ESP32 và Platform IO



Khuê Nguyễn Creator



Bài 9 WIFI: Lập trình ESP32 OTA nạp firmware trên Internet

Lập trình ESP32 FOTA nạp firmware qua mạng Internet với OTA Drive

Trong bài này chúng ta sẽ học cách sử dụng ESP32 FOTA (Firmware Over The...

4 COMMENTS

[ĐỌC THÊM](#)

Lập trình Nuvoton



Khuê Nguyễn Creator

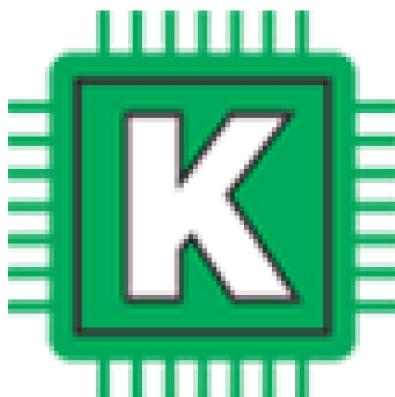


Cài đặt SDC Complier và Code:Blocks IDE

Hướng dẫn cài đặt SDCC và Code::Blocks lập trình Nuvoton

Ở bài này chúng ta sẽ cài đặt các công cụ cần thiết cho việc...

[ĐỌC THÊM](#)



KHUÊ NGUYỄN CREATOR
Chia sẻ đam mê

Blog này làm ra để lưu trữ tất cả những kiến thức, những câu chuyện của mình. Đôi khi là những ý tưởng nhất thời, đôi khi là các dự án tự mình làm. Chia sẻ cho người khác cũng là niềm vui của mình, kiến thức mỗi người là khác nhau, không hẳn quá cao siêu nhưng sẽ có lúc hữu dụng.

Liên Kết

Nhóm: Nghiên Lập Trình

Fanpage: Khuê Nguyên Creator

My Shop

Thông Tin

Tác Giả

Chính Sách Bảo Mật



Copyright 2022 © Khuê Nguyễn