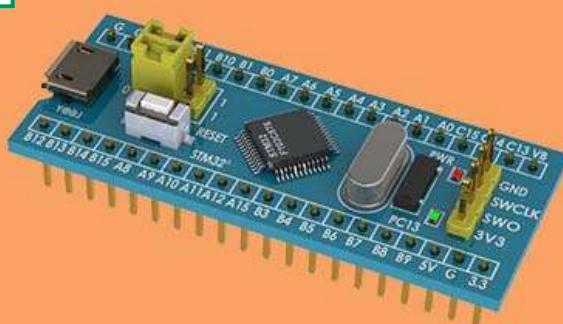


**LẬP TRÌNH STM32**

Bài 21: Lập trình STM32 Bit Band điều khiển GPIO

POSTED ON 15/08/2021 BY KHUÊ NGUYỄN

15
Th8**Khuê Nguyễn Creator**

Bài 21: Lập trình STM32 Bit Band điều khiển GPIO

STM32 Bit Band được sử dụng rất nhiều khi muốn điều khiển ngoại vi một cách nhanh chóng, nhưng có thể do mất thời gian định nghĩa ra các địa chỉ nên không được nhiều người viết hướng dẫn.

Trong bài này chúng ta sẽ học cách điều khiển GPIO bằng 4 cách khác nhau. Và cách sử dụng bit band trong STM32 nhé

Bài 20 trong Serie Học STM32 từ A tới Z

Mục Lục

- 1. Các cách điều khiển GPIO
- 2. Điều khiển bằng hàm trong thư viện
- 3. Điều khiển bằng thanh ghi
- 4. Điều khiển bằng bit field
- 5. Điều khiển bằng bit band
 - 5.1. Bit Band là gì?
 - 5.2. Bit Band Region trên STM32
 - 5.3. Cách truy cập vào vùng bit band trên STM32
 - 5.4. Lập trình Bit Band với STM32 HAL
- 6. Kết
 - 6.1. Related posts:



Các cách điều khiển GPIO

Thông thường có 3 cách điều khiển GPIO đó là:

- Sử dụng hàm điều khiển GPIO như: HAL_GPIO_Write, Read
- Sử dụng thanh ghi ODR: Khi thay đổi giá trị của thanh ghi ODR chúng sẽ ánh xạ tới đầu ra (các PORT).
- Sử dụng bit field: Tạo ra struct trong đó có các phần tử, đọc ghi giá trị từ struct đó.
- Sử dụng bit band: Sử dụng một khoảng bộ nhớ dành riêng cho việc truy cập theo từng bit.

Điều khiển bằng hàm trong thư viện

Đây là cách làm đơn giản nhất, nhưng tương tự cũng sẽ mất thời gian xử lý nhất.

Ví dụ trong HAL chúng ta sử dụng các hàm:

- HAL_GPIO_Write

- HAL_GPIO_Read
- HAL_GPIO_Toggle

Điều khiển bằng thanh ghi

Lập trình bằng thanh ghi khiến tốc độ của chương trình sẽ tăng lên, do không phải chạy qua các lệnh điều hướng (do thư viện sẽ phải tương thích với nhiều dòng vdk khác nhau).

Thao tác với thanh ghi chúng ta có 3 tác vụ chính đó là đọc/ghi/xóa.

Ví dụ: mình sẽ ghi và xóa vào bit 13 thanh ghi ODR của GPIO C. Lệnh này tương ứng với việc bạn bật và tắt led trên chân PC13

- Để ghi bit: GPIOC->ODR |= 1<<13; Giải thích: chúng ta sẽ dịch bit 1 sang trái 13 đơn vị, tương ứng với bit 13 trên ODR GPIOC, sau đó sử dụng phép tử OR để thay đổi giá trị của bit đó.
- Để xóa bit: GPIOC->ODR &= ~(1<<13); Tương tự dịch bit như trên nhưng chúng ta sẽ sử dụng toán tử AND và đổi bit 1 thành 0. Lúc này sẽ là xóa bit 13
- Để đọc bit: PC13_IN = (GPIOC->IDR) & (0x0001<<13). Bạn phải tạo 1 số 16bit có giá trị là 1, sau đó dịch trái 13 để tới bit 13 trên thanh ghi. Khi sử dụng phép toán & với giá trị thanh ghi đó. Tất cả những giá trị khác sẽ về 0, chỉ còn giá trị bit 13 nếu là 0 thì kết quả trả về là 0, nếu là 1 thì trả về 1

Điều khiển bằng bit field

Đầu tiên, bạn phải tạo một cấu trúc trường bit theo yêu cầu của bạn.

```
/* define structure of Port Pin*/
typedef struct {
    volatile unsigned int Bit0:1;
    volatile unsigned int Bit1:1;
    volatile unsigned int Bit2:1;
```

```

volatile unsigned int Bit3:1;
.
.
.
volatile unsigned int Bit31:1;
}SPortPin;

```

Tạo một con trỏ đến trường bit mô tả ở trên và gán địa chỉ của thanh ghi cho con trỏ mà bạn muốn truy cập.

```

volatile SPortPin *psGpioPort = (volatile SPortPin *) (GPIO_BASE + GPIO_OFFSET

```

Với GPIO_BASE là địa chỉ bắt đầu của GPIO, GPIO_OFFSET là địa chỉ cộng thêm. (VD: ODR là 0x0C)

Bây giờ cấu trúc trường bit của bạn đã được ánh xạ với thanh ghi phần cứng mà bạn muốn truy cập. Với GPIO là thanh ghi ODR và IDR.

Để đọc giá trị từ thanh ghi chúng ta sử dụng lệnh.

```

Value = psGpioPort-> Bit1;

```

Để ghi giá trị vào thanh ghi ta sử dụng

```

psGpioPort-> Bit1 = 1;
or
psGpioPort-> Bit1 = 0;

```

Điều khiển bằng bit band

Bit Band là gì?

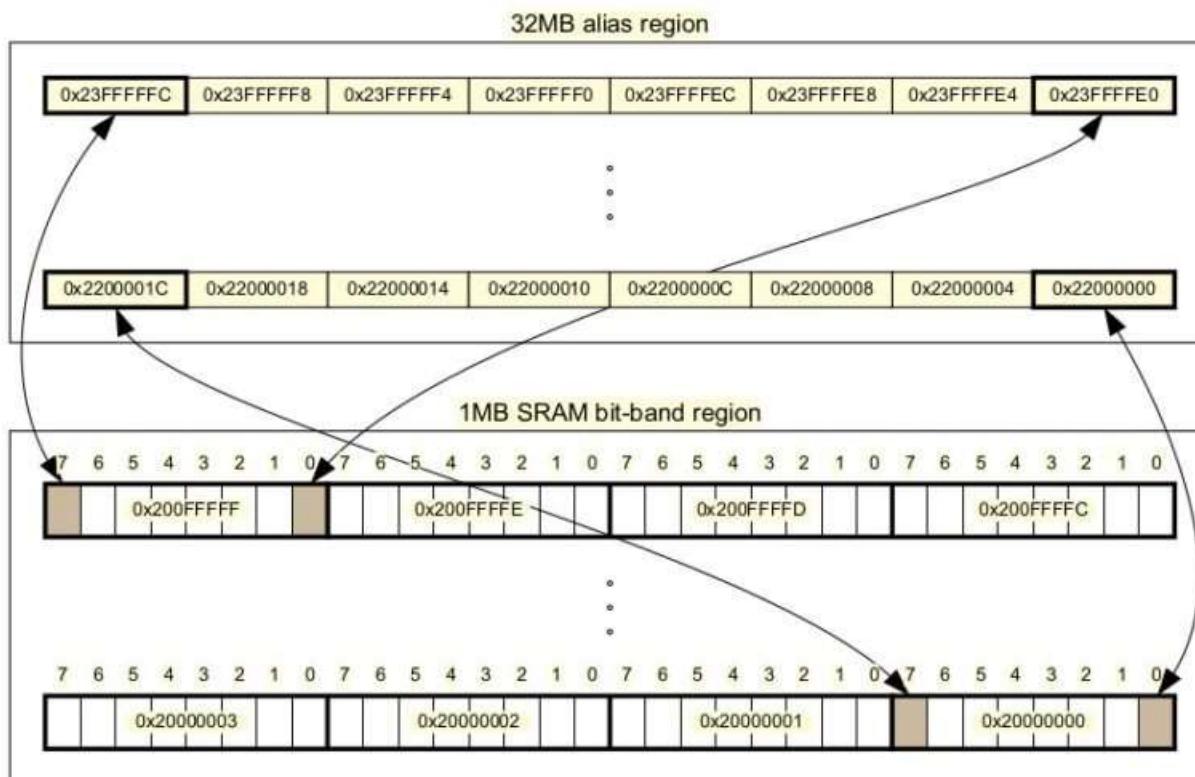
Bit-banding là khả năng điều khiển theo từng bit của một vùng nhớ tới một word (một số 32 bit) trong vùng bí danh (aliased region). Vùng nhớ bí danh là vùng

nhớ chỉ để dùng trong việc lưu trữ dữ liệu cho các vùng nhớ.

Nghĩa là khi bạn ghi dữ liệu vào cùng **Bit Band Alias region** một thao tác đọc/sửa/ghi sẽ được tạo ra để thay đổi giá trị bit trong vùng **Bit Band Region**. Tất cả các thao tác này được thực hiện bởi hệ thống vậy nên tốc độ xử lý rất nhanh.

Bit band thường được sử dụng để truy cập nhanh các thanh ghi, đặc biệt các ứng dụng cần điều khiển GPIO một cách nhanh chóng.

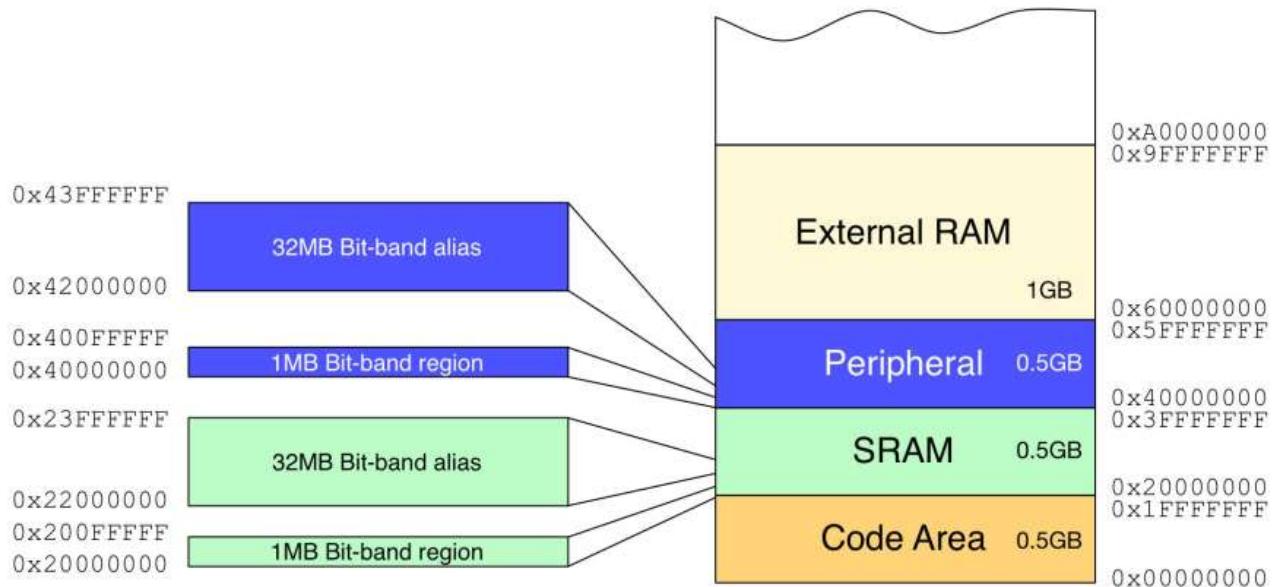
Cách này sẽ giúp việc lập trình GPIO trên STM32 đơn giản hơn rất nhiều, không cần nhiều câu lệnh lăng nhăng.



Cách dữ liệu map với nhau trong SRAM bit band

Bit Band Region trên STM32

ARM định nghĩa 2 vùng nhớ bit-band cho Cortex-M, mỗi vùng là 1MB và được map tới vùng bí danh 32Mbit. Mỗi một số 32bit trong vùng bí danh sẽ tương đương với một bit trong vùng bit-band.

*bit band map*

Vùng bit-band đầu tiên là từ 0x2000 0000 đến 0x200F FFFF của SRAM và **vùng bí danh** tương ứng của nó là 0x2200 0000 tới 0x23FF FFFF.

Vùng bit-band còn lại là từ 0x4000 0000 và kết thúc ở 0x400F FFFF của các ngoại vi và vùng bí danh tương ứng là từ 0x4200 0000 đến 0x43FF FFFF.

Chi tiết các bạn đọc lại bài [Bản đồ bộ nhớ trên STM32](#)

Cách truy cập vào vùng bit band trên STM32

Để truy cập được vào vùng bit-band bí danh, ta dùng công thức sau:

```
bit_band_address = alias_region_base + (region_base_offset * 32) + (bit_number * 4)
```

Ví dụ: Chúng ta sẽ điều khiển led trên chân PC13

- Vùng **alias_region_base** = 0x4200 0000 là địa chỉ bắt đầu vùng bí danh của ngoại vi
- Vùng **region_base_offset** chính là địa chỉ thanh ghi muốn truy cập của Port C là 0x4001100C

- **bit number** = 13 chính là bit số bao nhiêu trên thanh ghi đó

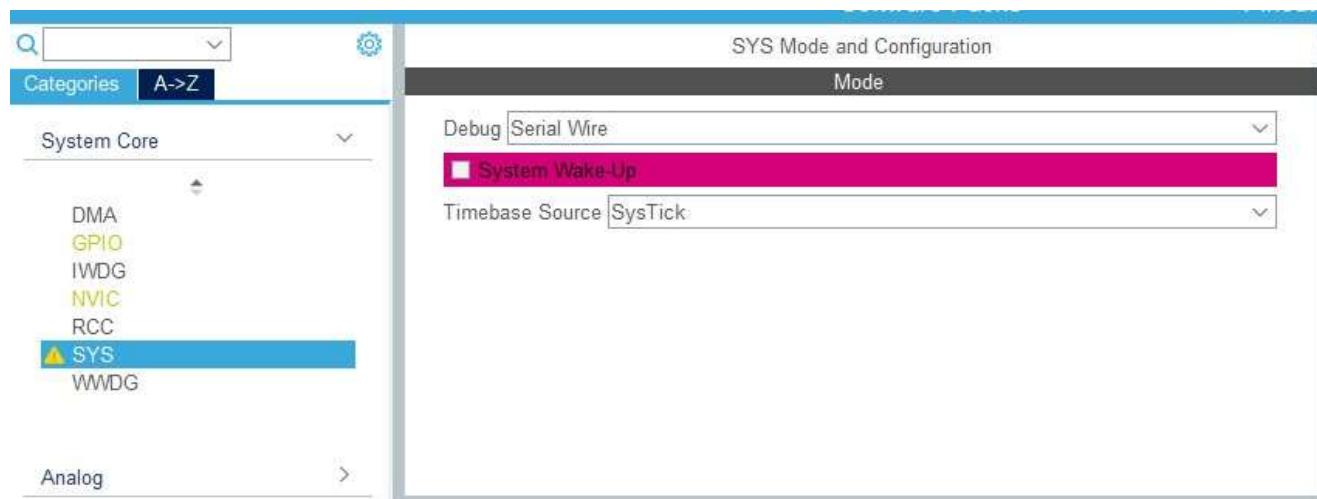
Vậy theo công thức chúng ta tính được địa chỉ bitband của **PC13** là:
0x422201B4

Nếu cứ tính bằng tay thì cũng rất mất thời gian ngồi tra cứu phải không, ai mà nhớ hết được. Vậy nên để dễ dàng trong tính toán mình sẽ sử dụng thư viện HAL và một vài dòng code để tính toán nhanh hơn.

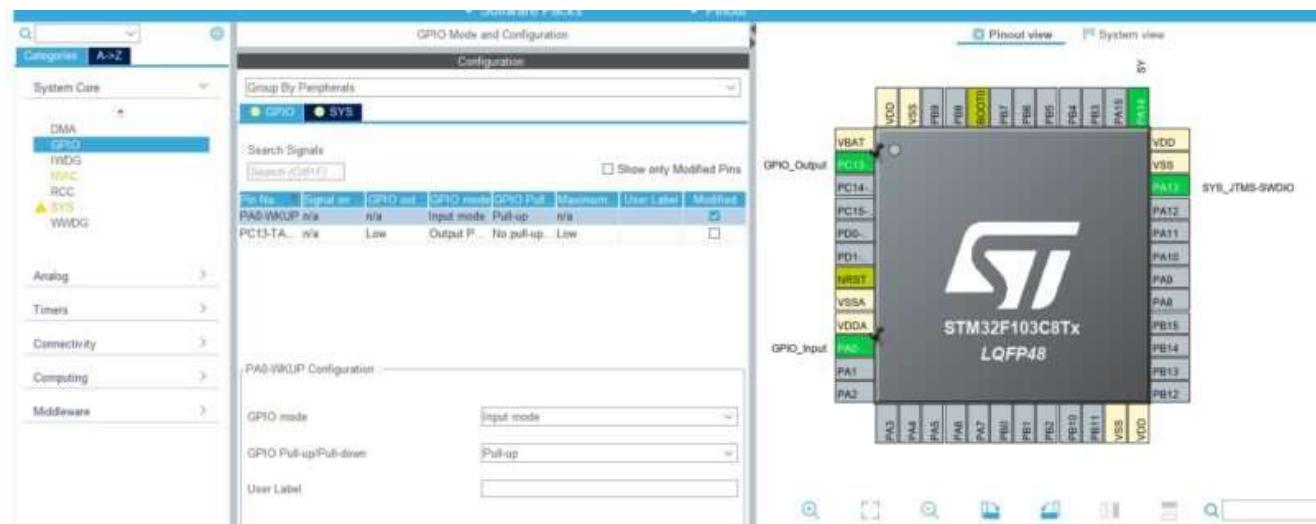
Lập trình Bit Band với STM32 HAL

Trong bài này mình sẽ sử dụng PA0 làm nút nhấn và PC13 điều khiển LED. Giống [bài 3 STM32 GPIO](#), và xem code đã được thay đổi như thế nào nhé!

Tạo một project trên cube MX. SYS – Debug – Serial Write, nhớ làm bước này để chúng ta debug nhé



GPIO cho PC13 là output, PA0 là input pull up



Gen code và mở trên Keil C

Keil C mình sẽ khai báo các biến chứa địa chỉ bit band của PC13 và PA0

```

44
45  /* USER CODE BEGIN PV */
46  uint32_t PC13_Bit_Band_Add;
47  uint32_t PC_ODR_Offset;
48  uint8_t pin_num1 = 13;
49
50  uint32_t PA0_Bit_Band_Add;
51  uint32_t PA_IDR_Offset;
52  uint8_t pin_num2 = 0;
53  uint8_t button_val;
54  /* USER CODE END PV */
55
56  /* Private function prototypes ----- */
57  void SystemClock_Config(void);
58  static void MX_GPIO_Init(void);
59  /* USER CODE BEGIN PFP */
60
61  /* USER CODE END PFP */
62
63  /* Private user code ----- */
64  /* USER CODE BEGIN O */
65
66  /* USER CODE END O */
67
68  /**
69   * @brief The application entry point.
70   * @retval int
71   */
72  int main(void)

```

Sử dụng công thức bên trên để tính toán, với:

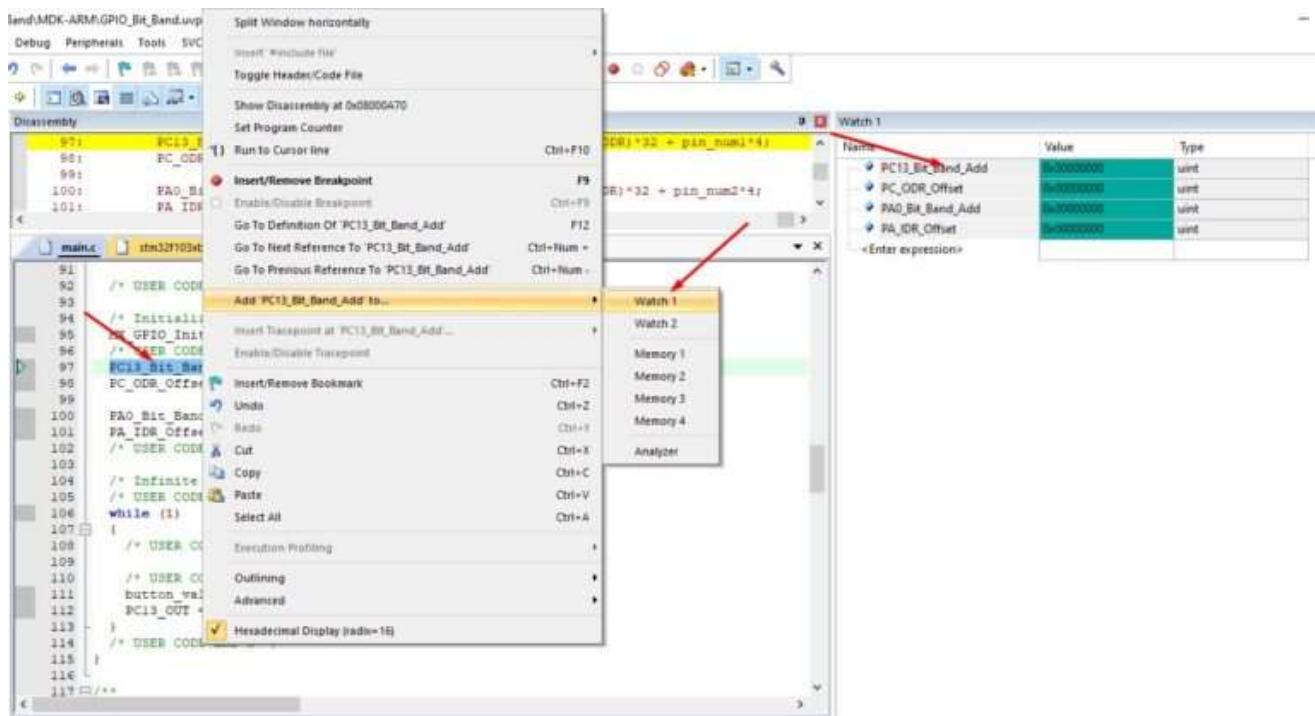
- PERIPH_BB_BASE: địa chỉ bit band
- (uint32_t)(&GPIOC->ODR): địa chỉ của thanh ghi ODR thuộc port C
- pin_num: chân hay bit cần can thiệp

```

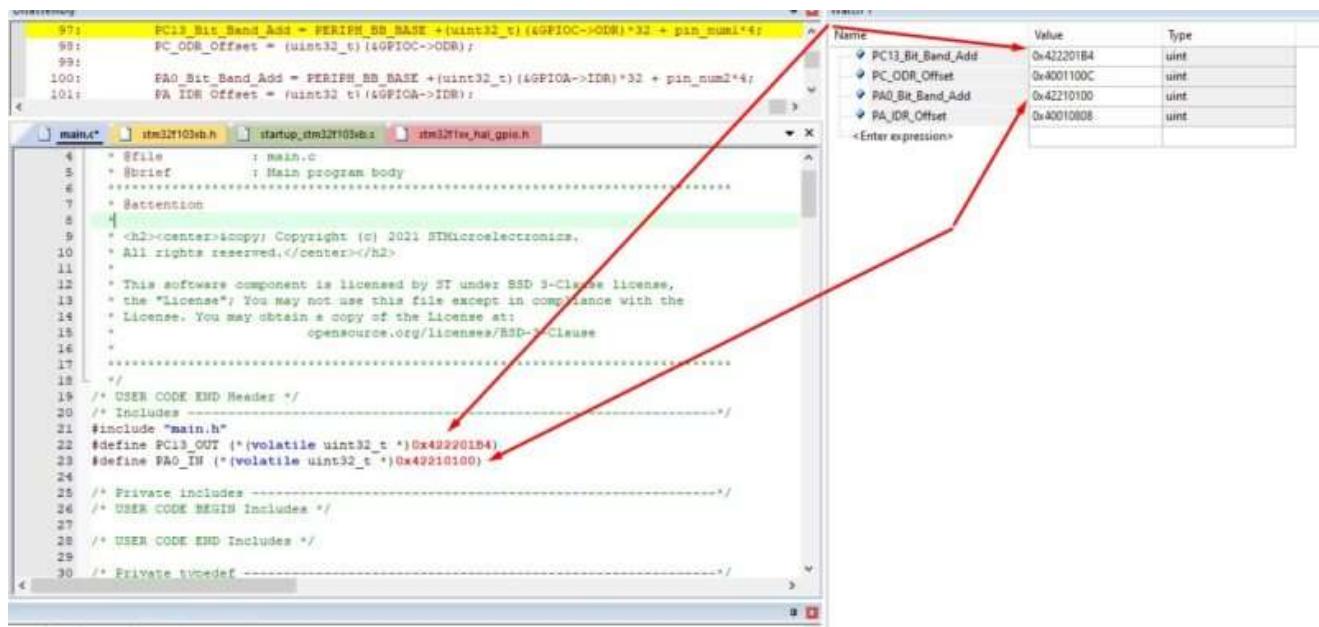
96  /* USER CODE BEGIN 2 */
97  PC13_Bit_Band_Add = PERIPH_BB_BASE +(uint32_t)(&GPIOC->IDR)*32 + pin_num1*4;
98  PC_ODR_Offset = (uint32_t)(&GPIOC->ODR);
99
100 PA0_Bit_Band_Add = PERIPH_BB_BASE +(uint32_t)(&GPIOA->IDR)*32 + pin_num2*4;
101 PA_IDR_Offset = (uint32_t)(&GPIOA->IDR);
102 /* USER CODE END 2 */
103
104 /* Infinite loop */
105 /* USER CODE BEGIN WHILE */
106 while (1)
107 {
    /* USER CODE END WHILE */

```

Nhấn F7 để build và Ctr_F5 để vào Debug. Trong debug add các biến vào Watch 1.



Nhấn Run để lấy giá trị các biến, sau đó define PC13_OUT và PA0_IN sử dụng các giá trị đọc được.



Tiếp tới trong while chúng ta đọc giá trị của PA0 sau đó ghi vào PC13. (Bạn có thể xóa các dòng tính toán bên trên đi dc rồi nhé).

```

106     while (1)
107     {
108         /* USER CODE END WHILE */
109
110         /* USER CODE BEGIN 3 */
111         button_val = PA0_IN;
112         PC13_OUT = button_val;
113     }
114     /* USER CODE END 3 */
115 }
116

```

Chúng ta sẽ được kết quả tương tự bài GPIO nhưng code được thu gọn hơn rất nhiều đúng không nào.

Để tiếp tục với các chân GPIO khác, các bạn cũng làm tương tự nhé

Kết

Chúng ta đã học qua các cách điều khiển GPIO trong **STM32 HAL**, các cách này có thể ứng dụng với nhiều dòng vi điều khiển khác nhau chứ không riêng gì STM32. Vì bản chất của **vi điều khiển** là giống nhau mà.

Để sử dụng bit band, quan trọng là bạn cần nắm rõ bản đồ bộ nhớ (memory map) của dòng đấy, nếu không hãy sử dụng các thư viện đã có khai báo sẵn như HAL, SPL, LL bạn chỉ cần tìm tới nơi define chúng là sẽ rõ ràng mình đang truy cập tới địa chỉ nào nhé.

Nếu cảm thấy bài viết này có ích hãy chia sẻ với bạn bè nhé. Đừng quên ra nhập hội những anh em **Nghiện lập trình** nhé!!

5/5 - (2 bình chọn)

Related Posts:

1. [Lập trình STM32 DFPlayer phát nhạc từ thẻ nhớ](#)
2. [Lập trình STM32 đọc nhiệt độ với DS18b20 giao tiếp I2C](#)
3. [Hướng dẫn download và sử dụng tài liệu Lập trình STM32](#)
4. [Bài 14: Sử dụng STM32 IWDG Independent Watchdog Timer chống treo vi điều khiển](#)
5. [Bài 10: Giao thức I2C, lập trình STM32 với module RTC DS3231](#)
6. [Bài 2: Tổng quan về KIT STM32F103C8T6 Blue Pill](#)



KHUÊ NGUYỄN

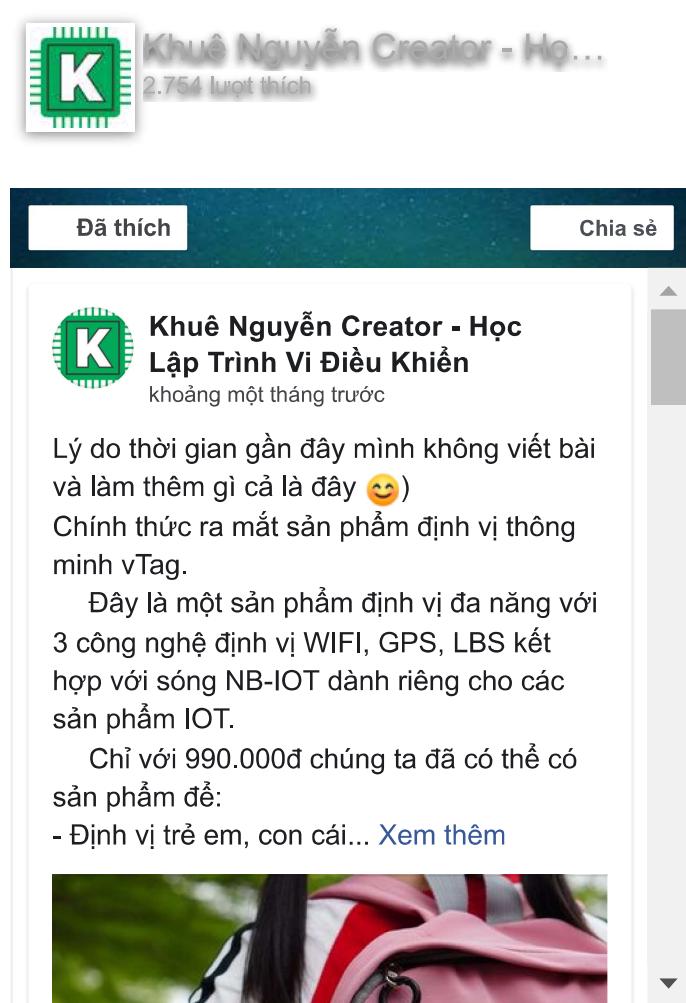
Chỉ là người đam mê điện tử và lập trình. Làm được gì thì viết cho anh em xem thôi. :D

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 - Học Lập Trình Vi Điều Khiển
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



Khuê Nguyễn Creator



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



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 giải quyết với chuột và bàn phím

[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

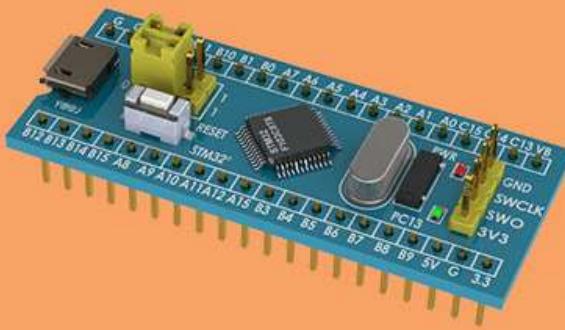
Trong bài này chúng ta sẽ cùng nhau tìm hiểu STM32 DAC với KIT STM32F407VE...

[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



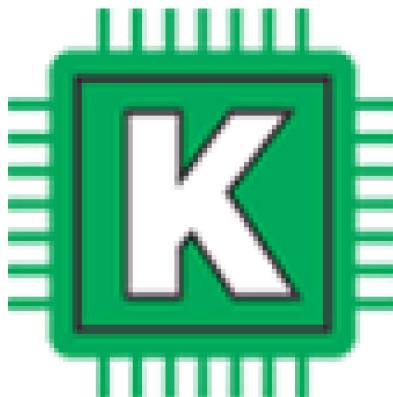
Cài đặt SDC Complier và Code:Blocks IDE

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.

DMCA PROTECTED

Liên Kết

Nhóm: Nghiên Lập Trình

Fanpage: Khuê Nguyên Creator

My Shop

Thông Tin

Tác Giả

Chính Sách Bảo Mật



Copyright 2022 © Khuê Nguyễn