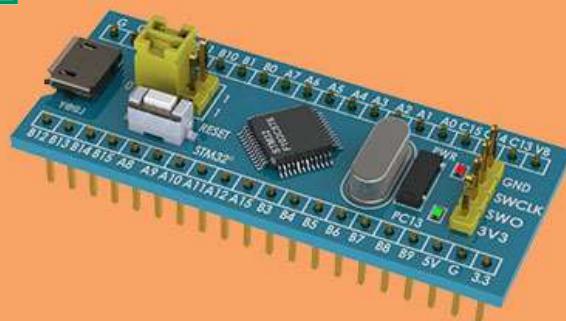


**LẬP TRÌNH STM32**

# Bài 12: Lập trình STM32 với giao thức SPI

POSTED ON 20/07/2020 BY KHUÊ NGUYỄN

20  
Th7**Khuê Nguyễn Creator**

## Bài 12: SPI là gì, hướng dẫn sử dụng giao thức SPI

Lập trình STM32 sử dụng giao thức SPI sẽ giúp chúng ta hiểu được.

- Nguyên lý truyền nhận của chuẩn SPI
- Cách thiết lập giao tiếp SPI trên STM32 Cube MX
- Lập trình với giao thức SPI

Bài số 12 trong serie **Học lập trình STM32 từ A tới Z**

## Mục Lục

- 1. Giao thức SPI là gì
  - 1.1. Cơ bản về giao thức SPI
  - 1.2. Nguyên lý hoạt động của giao thức SPI
  - 1.3. Chế độ hoạt động của giao thức SPI
  - 1.4. Các kiểu cấu hình giao thức SPI
- 2. Cấu hình STM32 SPI trong Cube MX
- 3. Lập trình giao thức SPI trong Keil C
- 4. Kết luận
  - 4.1. Related posts:



# Giao thức SPI là gì

## Cơ bản về giao thức SPI

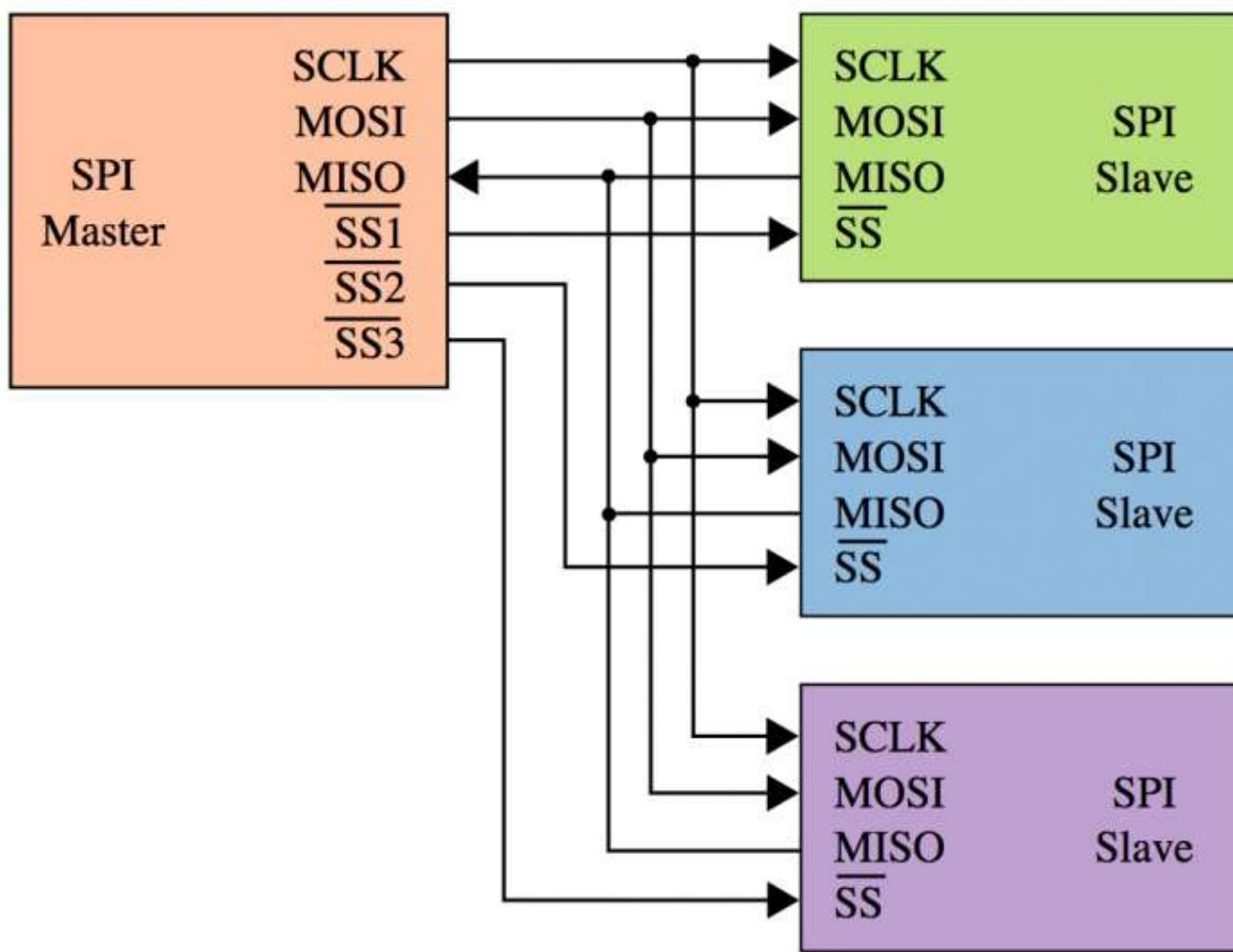
**SPI Serial Peripheral Interface** là một chuẩn truyền **thông nối tiếp đồng bộ** dùng để chuyển dữ liệu ở chế độ song công toàn phần (full duplex).

Giao thức SPI thường được sử dụng On Board hoặc các đường tín hiệu ngắn. Tốc độ của SPI khá cao thường phụ thuộc vào tốc độ xung truyền vào bộ SPI

Giao thức SPI là dao thức dạng Master –Slave trong đó Master giữ quyền điều khiển xung Clock và chọn Slave nào giao tiếp với mình.

Bus SPI bao gồm 4 đường tín hiệu:

- MOSI (Master Out Slave In): Đường truyền tín hiệu từ Master đến Slave
- MISO (Master In Slave Out): Đường truyền tín hiệu từ Slave đến Master
- CLK (Serial Clock): Đường phát xung đồng hồ được điều khiển bởi Master
- CS (Chip Select): Đường chọn chip giao tiếp với Master, được kéo xuống 0 khi được chọn

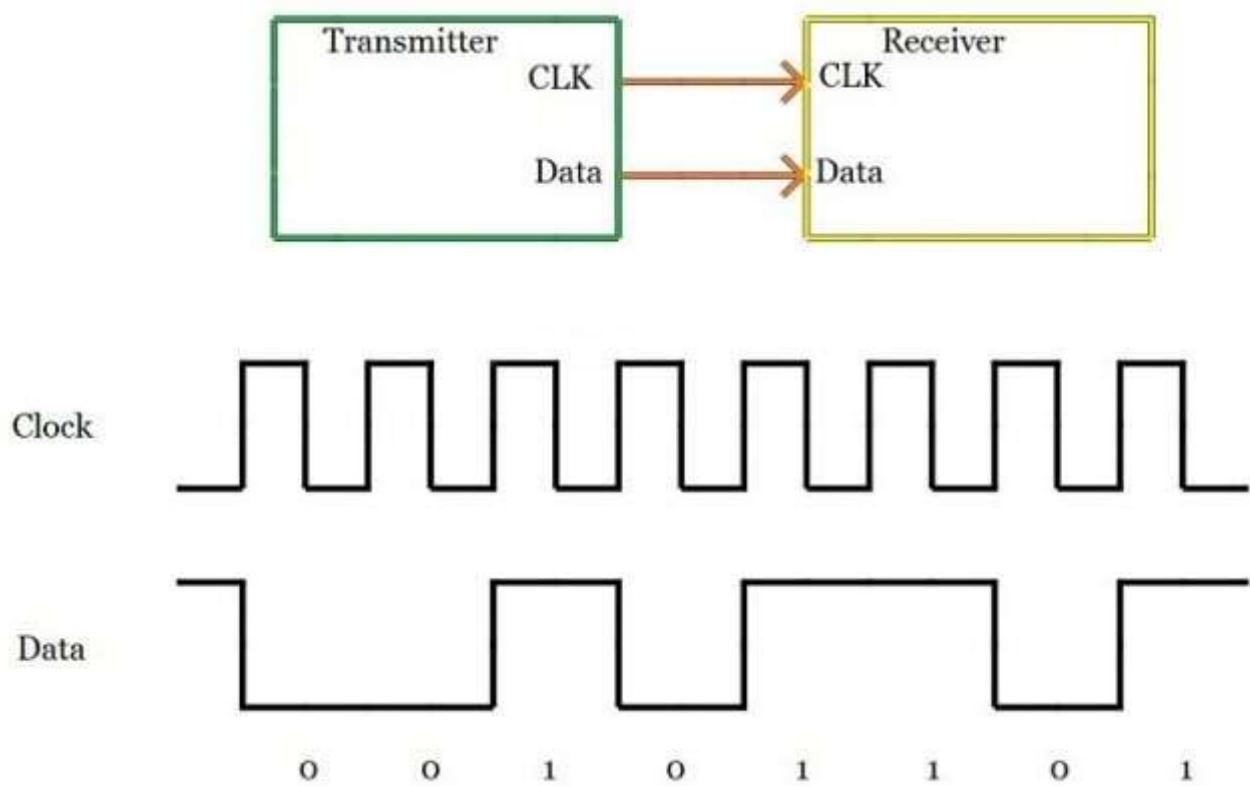


Bài 12: Lập trình STM32 với giao thức SPI 57

## Nguyên lý hoạt động của giao thức SPI

Nguyên lý hoạt động như sau:

1. Khi muốn truyền nhận dữ liệu tới các Slave, đầu tiên Master kéo đường CS kết nối từ Master tới Slave đó xuống 0.
2. Gửi xung Clock, tương ứng với mỗi Clock sẽ gửi DATA trên chân MOSI tại thời điểm Clock ở mức cao (hoặc thấp tùy người lập trình)
3. Slave cũng có thể gửi ngược lại DATA tại chân MISO tới Master
4. Sự truyền nhận dữ liệu là liên tục nên SPI thường có tốc độ rất cao



## Bài 12: Lập trình STM32 với giao thức SPI 58

### Chế độ hoạt động của giao thức SPI

Có 4 chế độ hoạt động của SPI dựa trên hai Bit **CPOL** và **CPHA**

**CPOL:** Clock Polarity xác định mức tín hiệu tại Clock lúc nhàn rỗi (Idle). Nếu **CPOL = 0**, lúc không gửi dữ liệu (nhàn rỗi) Clock sẽ ở mức thấp **CPOL = 1** sẽ ngược lại

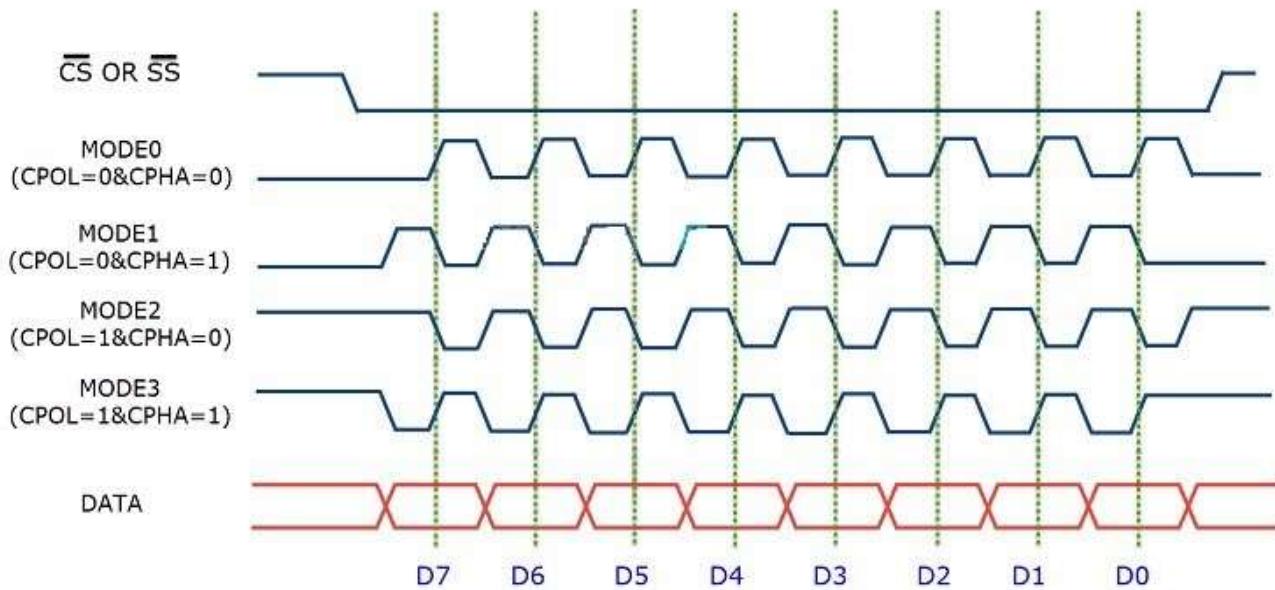
**CPHA:** Phase Clock xác định quá trình truyền dữ liệu tại cạnh lên (**CPHA = 0**) hay cạnh xuống (**CPHA = 1**)

Mode 0:

1. Mode 0: xảy ra khi Clock Polarity và Clock Phase là 0 (**CPOL = 0** và **CPHA = 0**). Trong Mode 0, truyền dữ liệu xảy ra trong khi cạnh lên của xung đồng hồ.
2. Mode 1: xảy ra khi Clock Polarity là 0 và Clock Phase là 1 (**CPOL = 0** và **CPHA = 1**). Trong mode 1, truyền dữ liệu xảy ra trong khi cạnh xuống của xung đồng hồ.

3. Mode 2: xảy ra khi Clock Polarity là 1 và Clock Phase là 0 (CPOL = 1 và CPHA = 0). Trong mode 2, truyền dữ liệu xảy ra trong khi cạnh lên của xung đồng hồ.

4. Mode 3: xảy ra khi Clock Polarity là 1 và Clock Phase là 1 (CPOL = 1 và CPHA = 1). Trong mode 3, truyền dữ liệu xảy ra trong khi cạnh lên của xung đồng hồ.



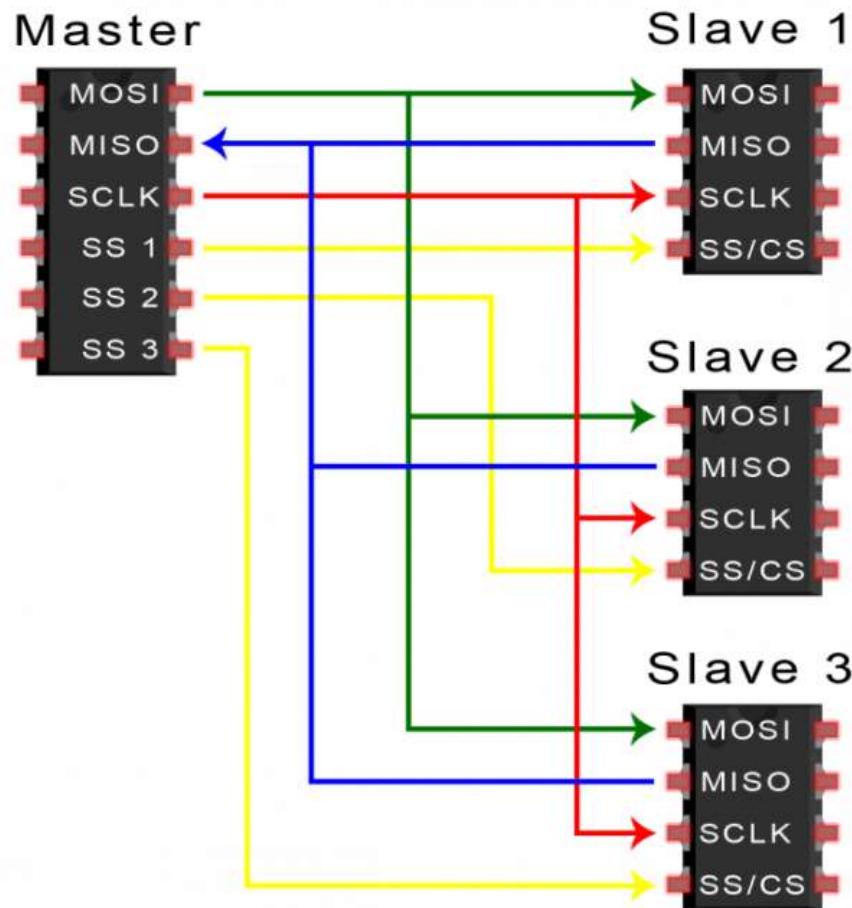
Bài 12: Lập trình STM32 với giao thức SPI 59

## Các kiểu cấu hình giao thức SPI

Có 2 loại cấu hình SPI

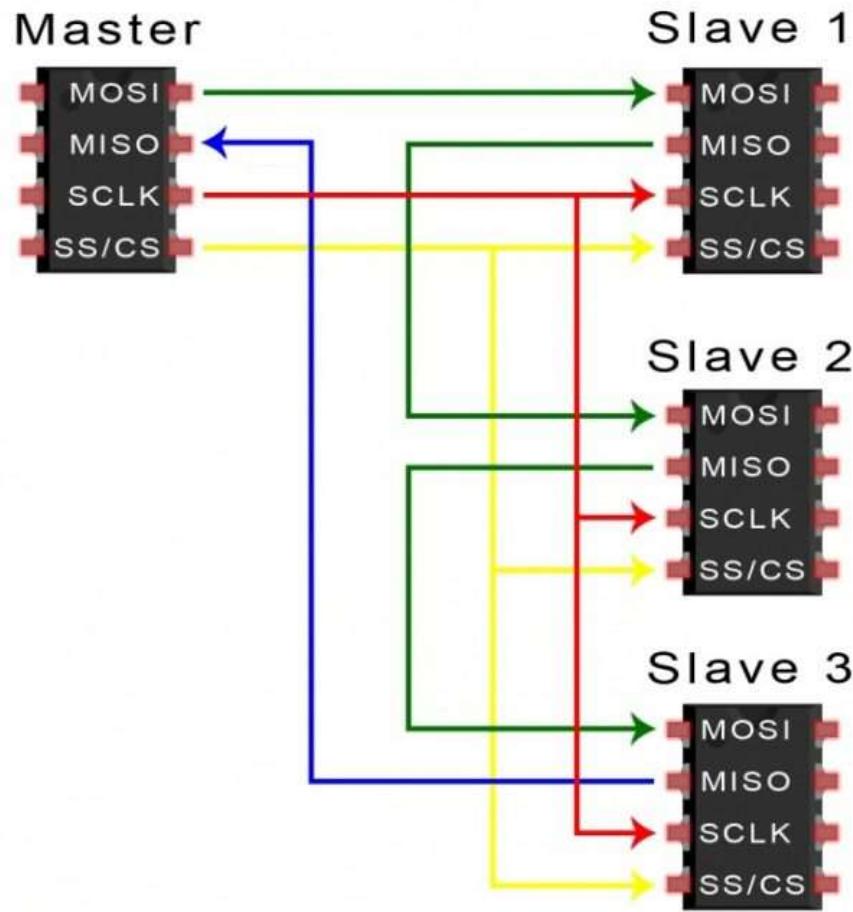
- Cấu hình Master và các Slave độc lập (Independent Slave Configuration)
- Cấu hình Daisy Chain (Daisy Chain Configuration).

Trong cấu hình Master và các Slave độc lập, Master đã dành riêng các đường Slave Select cho tất cả các Slave và mỗi Slave có thể được chọn riêng lẻ. Tất cả tín hiệu đồng hồ của các Slave được kết nối với chung với SCK của Master.



## Bài 12: Lập trình STM32 với giao thức SPI 60

Trong cấu hình Daisy Chain, chỉ có một đường Slave Select được kết nối với tất cả các Slave. MOSI của Master được kết nối với MOSI của Slave 1. MISO của Slave 1 được kết nối với MOSI của Slave 2 và v.v.. MISO của Slave cuối cùng được kết nối với MISO của Master.



## Bài 12: Lập trình STM32 với giao thức SPI 61

Tuy nhiên, cầu hình Daisy Chain không phải lúc nào cũng áp dụng được cho tất cả các thiết bị Slave. Do đó, ta cần phải tham khảo datasheet trước khi tiến hành kết nối.

## Cấu hình STM32 SPI trong Cube MX

Trong bài này chúng ta sẽ sử dụng Bộ SPI1 ở chế độ Master và SPI2 ở chế độ Slave. Truyền nhận ở chế độ ngắn và DMA

Trong CubeMX cấu hình như sau. SYS debug: Serial Wire

SPI1 Chế độ Full Duplex, Hard ware NSS: Disable, chân chọn chip ta sẽ sử dụng chân GPIO khác để dễ dàng điều khiển nếu sử dụng nhiều Slave

Parameter cấu hình như trong hình

The screenshot shows the STM32CubeMX software interface. On the left, there is a sidebar with categories like System Core, Analog, Timers, Connectivity (with CAN, I2C1, I2C2, SPI1 checked, SPI2, USART1, USART2, USART3, USB), Computing, and Middleware. The main area is titled "SPI1 Mode and Configuration". It has two tabs: "Mode" and "Configuration". Under "Mode", the "Mode" dropdown is set to "Full-Duplex Master" and "Hardware NSS Signal" is set to "Disable". Under "Configuration", there are tabs for NVIC Settings, DMA Settings, GPIO Settings, Parameter Settings (which is selected), and User Constants. A search bar at the top says "Search (Ctrl+F)". Below it, there are sections for Basic Parameters (Frame Format: Motorola, Data Size: 8 Bits, First Bit: MSB First), Clock Parameters (Prescaler: 8, Baud Rate: 1000.0 KBits/s, Clock Polarity (CPOL): Low, Clock Phase (CPHA): 1 Edge), and Advanced Parameters (CRC Calculation: Disabled, NSS Signal Type: Software).

## Bài 12: Lập trình STM32 với giao thức SPI 62

SPI2 chế độ Full-Duplex Slave, Hard ware NSS: Enable Bật chân chọn chip cho SPI2

Parameter cấu hình giống SPI1

## SPI2 Mode and Configuration

## Mode

Mode Full-Duplex Slave

Hardware NSS Signal Hardware NSS Input Signal

## Configuration

Reset Configuration

 NVIC Settings DMA Settings GPIO Settings Parameter Settings User Constants

Configure the below parameters :



## Basic Parameters

Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

## Clock Parameters

Prescaler (for Baud Rate)	8
Baud Rate	1000.0 KBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge

## Advanced Parameters

CRC Calculation	Disabled
NSS Signal Type	Input Hardware

## Bài 12: Lập trình STM32 với giao thức SPI 63

Chọn thêm chân PA4 làm chân Chip Select cho SPI1

GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO    SPI    SYS

Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin Na...	Signal on ...	GPIO out...	GPIO mode	GPIO Pull...	Maximum...	User Label	Modified
PA4	n/a	High	Output P...	No pull-up...	Low	CS	<input checked="" type="checkbox"/>

PA4 Configuration :

GPIO output level	High
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	Low
User Label	CS

## Bài 12: Lập trình STM32 với giao thức SPI 64

Bật ngắt cho SPI1 và SPI2

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
SPI1 global interrupt	<input checked="" type="checkbox"/>	0	0
SPI2 global interrupt	<input checked="" type="checkbox"/>	0	0

## Bài 12: Lập trình STM32 với giao thức SPI 65

Chọn toolchain đặt tên và Gen code

**Project Settings**

Project Name  
Bai12\_SPI\_Echo

Project Location  
D:\STM32 Cube Youtube

Application Structure  
Basic  Do not generate the main()

Toolchain Folder Location  
D:\STM32 Cube Youtube\Bai12\_SPI\_Echo\

Toolchain / IDE  
MDK-ARM  Min Version V5  Generate Under Root **Generate**

Linker Settings

Minimum Heap Size 0x200

Minimum Stack Size 0x400

**Mcu and Firmware Package**

Mcu Reference  
STM32F103C8Tx

Firmware Package Name and Version  
STM32Cube\_FW\_F1\_V1.8.0

Use Default Firmware Location  C:/Users/admin/STM32Cube/Repository/STM32Cube\_FW\_F1\_V1.8.0 **Browse**

Bài 12: Lập trình STM32 với giao thức SPI 66

## Lập trình giao thức SPI trong Keil C

Trong Keil C thêm các biến u8\_SPI1\_TxBuff để làm bộ đệm truyền SPI1, u8\_SPI2\_RxBuff làm bộ đệm nhận từ SPI2, biến Count để xem sự thay đổi của mảng truyền đi

```

20
21 /* Includes -----
22 #include "main.h"
23
24 /* Private includes -----
25 /* USER CODE BEGIN Includes */
26 #include <stdio.h>
27 #define sizeofBuff 20
28 uint8_t u8_SPI1_TxBuff[sizeofBuff] = "Master send";
29 uint8_t u8_SPI2_RxBuff[sizeofBuff];
30 uint8_t Count;
31 /* USER CODE END Includes */
32

```

## Bài 12: Lập trình STM32 với giao thức SPI 67

Tìm hàm HAL\_SPI\_RxCpltCallback copy và dán vào trên hàm main(), bên trong chúng ta kiểm tra có phải ngắt SPI2 không, sau đó bật ngắt nhận RX để nhận thêm chuỗi

```

54 /* Private function prototypes -----
55 void SystemClock_Config(void);
56 static void MX_GPIO_Init(void);
57 static void MX_SPI1_Init(void);
58 static void MX_SPI2_Init(void);
59 /* USER CODE BEGIN PFP */
60 void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi)
61 {
62     /* Prevent unused argument(s) compilation warning */
63     UNUSED(hspi);
64     if(hspi->Instance == SPI2)
65     {
66         HAL_SPI_Receive_IT(&hspi2, u8_SPI2_RxBuff,sizeofBuff);
67     }
68     /* NOTE : This function should not be modified, when the callback is needed,
69      the HAL_SPI_RxCpltCallback should be implemented in the user file
70      */
71 }
72 /* USER CODE END PFP */
73 /* Private user code -----
74 /* USER CODE BEGIN 0 */
75 /* USER CODE END 0 */
76 /* USER CODE BEGIN 0 */
77 /* USER CODE END 0 */
78 /* USER CODE END 0 */
79

```

## Bài 12: Lập trình STM32 với giao thức SPI 68

Trước while(1) chúng ta truyền dữ liệu đầu tiên lên SPI1 bằng hàm:

HAL\_SPI\_Transmit(&hspi1,u8\_SPI1\_TxBuff,sizeofBuff,100);

, sau đó bận nhận SPI2 bằng ngắt bằng hàm:

HAL\_SPI\_Receive\_IT(&hspi2, u8\_SPI2\_RxBuff,sizeofBuff);

Trong While (1) copy dữ liệu vào SPI1\_TxBuff

Cho chân CS xuống 0 để chọn chip, gửi dữ liệu, cho chân CS về 1 để kết thúc quá trình gửi.

Tăng biến Count và delay 1s

```

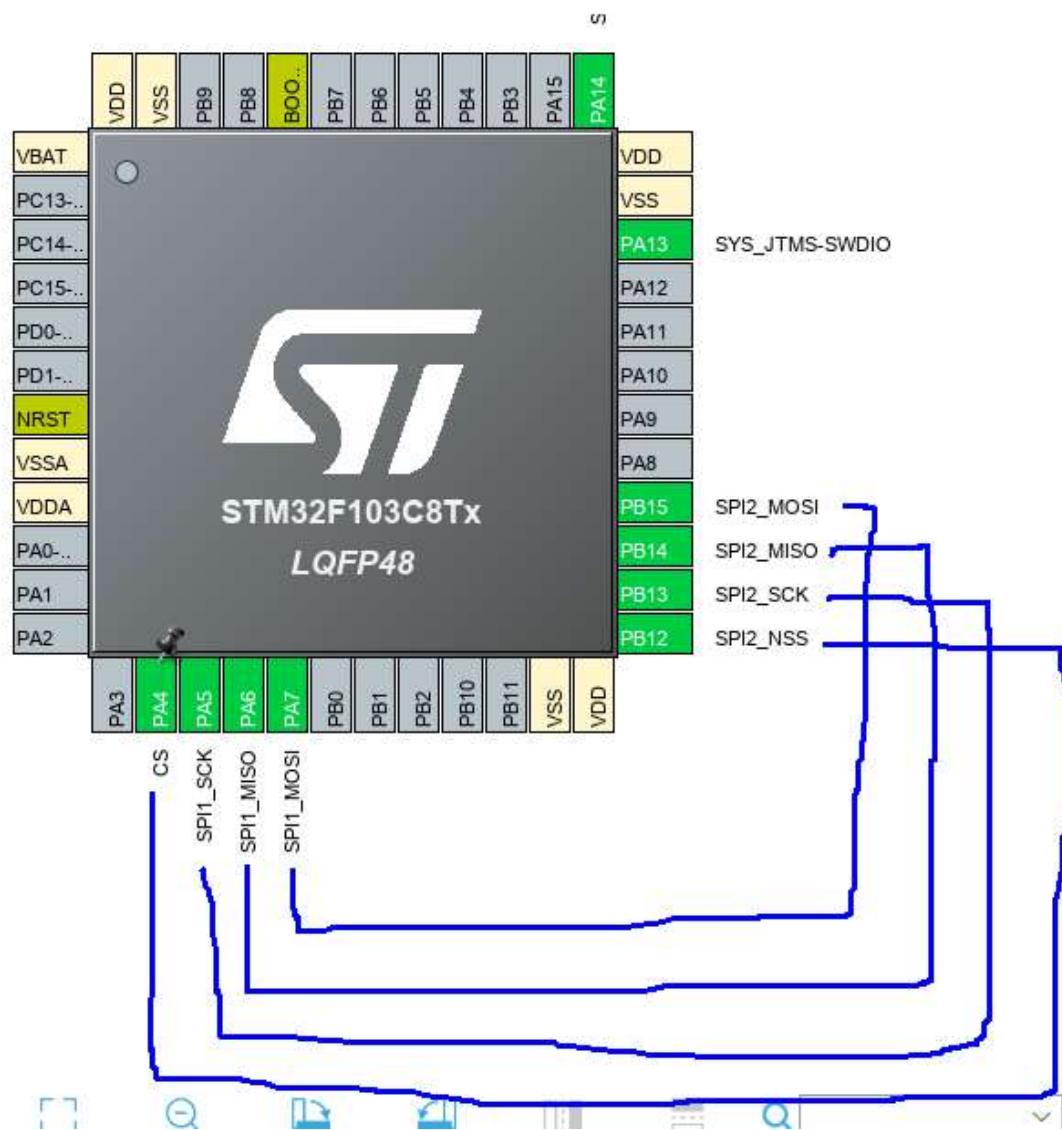
106     /* Initialize all configured peripherals */
107     MX_GPIO_Init();
108     MX_SPI1_Init();
109     MX_SPI2_Init();
110     /* USER CODE BEGIN 2 */
111     HAL_SPI_Receive_IT(&hspi2, u8_SPI2_RxBuff, sizeofBuff);
112     HAL_SPI_Transmit(&hspi1, u8_SPI1_TxBuff, sizeofBuff, 100);
113     /* USER CODE END 2 */
114
115     /* Infinite loop */
116     /* USER CODE BEGIN WHILE */
117     while (1)
118     {
119         /* USER CODE END WHILE */
120
121         /* USER CODE BEGIN 3 */
122         sprintf((char*)u8_SPI1_TxBuff, "Gửi lần thu: %d \n", Count);
123         HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, GPIO_PIN_RESET);
124         HAL_SPI_Transmit(&hspi1, u8_SPI1_TxBuff, sizeofBuff, 100);
125         HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, GPIO_PIN_SET);
126         Count++;
127         HAL_Delay(1000);
128     }
129     /* USER CODE END 3 */

```

Bài 12: Lập trình STM32 với giao thức SPI 69

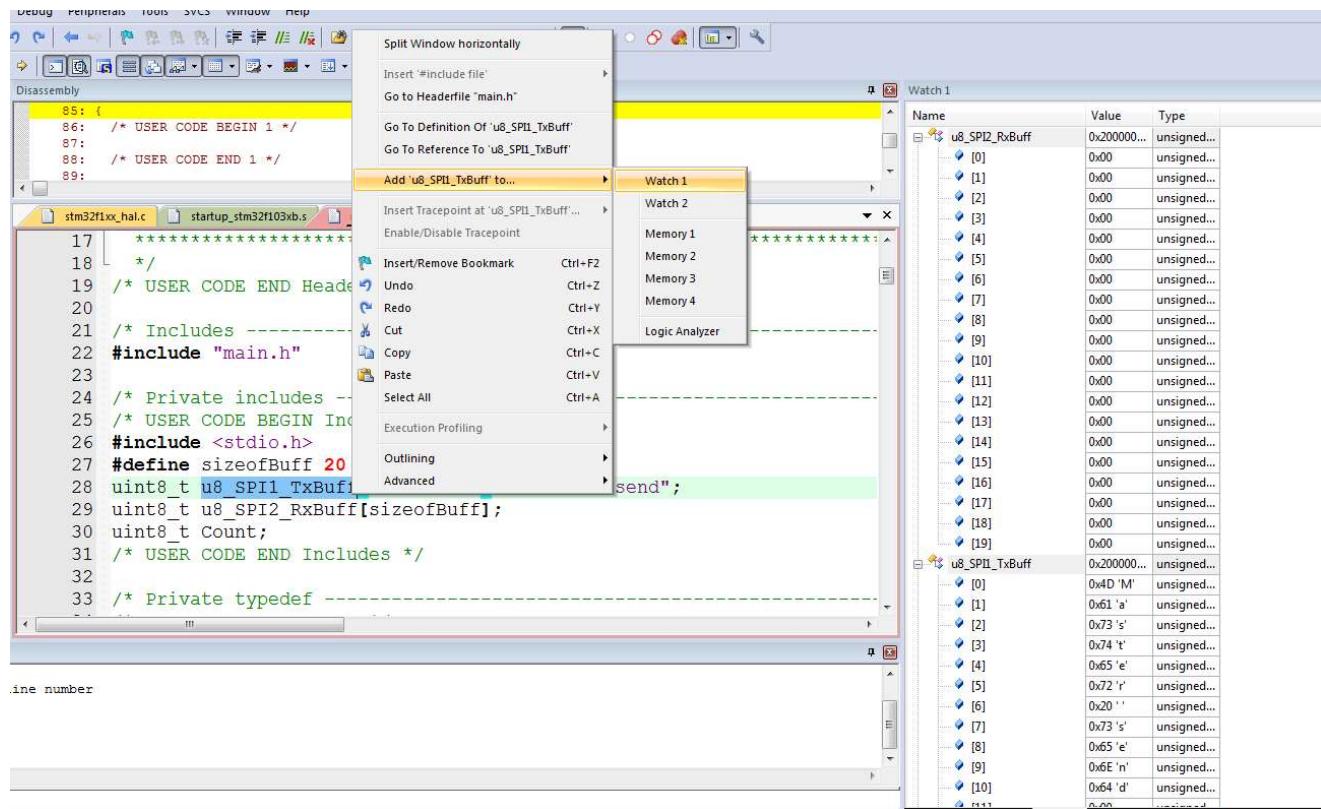
Nhấn Build và nạp chương trình. Kết nối phần cứng như sau:

Các chân MOSI MISO CLK nối với nhau, chân CS của SPI1 nối với chân NSS của SPI2



## Bài 12: Lập trình STM32 với giao thức SPI 70

Vào chế độ debug, chuột phải vào 2 Buffer truyền và nhận, chọn Add to watch 1



## Bài 12: Lập trình STM32 với giao thức SPI 71

Nhấn Run và xem kết quả

Name	Value	Type
[4]	0x6C 'l'	unsigned...
[5]	0x61 'a'	unsigned...
[6]	0x6E 'n'	unsigned...
[7]	0x20 ''	unsigned...
[8]	0x74 't'	unsigned...
[9]	0x68 'h'	unsigned...
[10]	0x75 'u'	unsigned...
[11]	0x3A ':'	unsigned...
[12]	0x20 ''	unsigned...
[13]	0x35 '5'	unsigned...
[14]	0x31 '1'	unsigned...
[15]	0x20 ''	unsigned...
[16]	0x0A	unsigned...
[17]	0x00	unsigned...
[18]	0x00	unsigned...
[19]	0x00	unsigned...
u8_SPI1_TxBuff	0x200000...	unsigned...
[0]	0x47 'G'	unsigned...
[1]	0x75 'u'	unsigned...
[2]	0x69 'i'	unsigned...
[3]	0x20 ''	unsigned...
[4]	0x6C 'l'	unsigned...
[5]	0x61 'a'	unsigned...
[6]	0x6E 'n'	unsigned...
[7]	0x20 ''	unsigned...
[8]	0x74 't'	unsigned...
[9]	0x68 'h'	unsigned...
[10]	0x75 'u'	unsigned...
[11]	0x3A ':'	unsigned...
[12]	0x20 ''	unsigned...
[13]	0x35 '5'	unsigned...
[14]	0x31 '1'	unsigned...
[15]	0x20 ''	unsigned...

## Bài 12: Lập trình STM32 với giao thức SPI 72

Giá trị truyền và nhận đã giống nhau

Nhận dữ liệu bằng DMA

Trong SPI2 ta chuyển sang tab DMA, nhấn Add

DMA Request là SPI2\_RX, Mode là Circular (ta chỉ cần gọi hàm nhận DMA 1 lần , không cần phải gọi lại trong ngắn )

## Data Width là Byte

SPI2 Mode and Configuration

**Mode**

Mode: Full-Duplex Slave

Hardware NSS Signal: Hardware NSS Input Signal

**Configuration**

**Reset Configuration**

**DMA Request** | **Channel** | **Direction** | **Priority**

SPI2_RX	DMA1 Channel 4	Peripheral To Memory	Low
---------	----------------	----------------------	-----

**Add** | **Delete**

**DMA Request Settings**

Mode: Circular	Increment Address: <input type="checkbox"/>	Peripheral	Memory: <input checked="" type="checkbox"/>
Data Width: Byte		Byte	Byte

## Bài 12: Lập trình STM32 với giao thức SPI 73

Trong Keil C comment hoặc xóa phần xử lý ngắn (không dùng tới hoặc dùng để làm tác vụ nào đó khi truyền xong)

Phần trước While (1) thay hàm Recive\_IT bằng Recive\_DMA

```
105  /* USER CODE END SysInit */
106
107
108  /* Initialize all configured peripherals */
109  MX_GPIO_Init();
110  MX_DMA_Init();
111  MX_SPI1_Init();
112  MX_SPI2_Init();
113  /* USER CODE BEGIN 2 */
114  //HAL_SPI_Receive_IT(&hspi2, u8_SPI2_RxBuff, sizeofBuff);
115  HAL_SPI_Receive_DMA(&hspi2, u8_SPI2_RxBuff, sizeofBuff);
116  HAL_SPI_Transmit(&hspi1, u8_SPI1_TxBuff, sizeofBuff, 100);
117  /* USER CODE END 2 */
118
119  /* Infinite loop */
120  /* USER CODE BEGIN WHILE */
121  while (1)
```

## Bài 12: Lập trình STM32 với giao thức SPI 74

Nhấn Build và nạp chương trình, chạy debug để xem kết quả

[4]	0x6C 'l'	unsigned...
[5]	0x61 'a'	unsigned...
[6]	0x6E 'n'	unsigned...
[7]	0x20 ''	unsigned...
[8]	0x74 't'	unsigned...
[9]	0x68 'h'	unsigned...
[10]	0x75 'u'	unsigned...
[11]	0x3A ':'	unsigned...
[12]	0x20 ''	unsigned...
[13]	0x34 '4'	unsigned...
[14]	0x20 ''	unsigned...
[15]	0x0A	unsigned...
[16]	0x00	unsigned...
[17]	0x00	unsigned...
[18]	0x00	unsigned...
[19]	0x00	unsigned...
⋮	⋮	⋮
u8_SPI1_TxBuff	0x200000...	unsigned...
[0]	0x47 'G'	unsigned...
[1]	0x75 'u'	unsigned...
[2]	0x69 'i'	unsigned...
[3]	0x20 ''	unsigned...
[4]	0x6C 'l'	unsigned...
[5]	0x61 'a'	unsigned...
[6]	0x6E 'n'	unsigned...
[7]	0x20 ''	unsigned...
[8]	0x74 't'	unsigned...
[9]	0x68 'h'	unsigned...
[10]	0x75 'u'	unsigned...
[11]	0x3A ':'	unsigned...
[12]	0x20 ''	unsigned...
[13]	0x34 '4'	unsigned...
[14]	0x20 ''	unsigned...
[15]	0x0A	unsigned...

Bài 12: Lập trình STM32 với giao thức SPI 75

## Kết luận

Giao thức SPI là một trong những giao thức cơ bản và phổ biến nhất trong lập trình vi điều khiển. Với SPI bạn có thể sử dụng để giao tiếp với nhiều loại device khác nhau như: TFT LCD, RFID, Camera ....

Giao thức SPI có thể được bắt gặp tại bất cứ một sản phẩm nhúng nào, hãy rèn luyện và làm chủ giao thức này nhé.

5/5 - (3 bình chọn)

## Related Posts:

1. [Lập trình STM32 từ A tới Z](#)
2. [Bài 11: Lập trình STM32 với Giao thức UART](#)
3. [Bài 10: Giao thức I2C, lập trình STM32 với module RTC DS3231](#)
4. [Bài 9: Lập trình STM32 ADC nhiều kênh với DMA](#)
5. [Bài 8: Lập trình STM32 đọc ADC một kênh](#)
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

## 11 THOUGHTS ON “BÀI 12: LẬP TRÌNH STM32 VỚI GIAO THỨC SPI”



*xuân duy* says:

anh có thể làm về giao tiếp stm32 với thẻ sd card qua spi đc ko ạ.

27/01/2021 AT 5:18 CHIỀU

TRẢ LỜI



*Khuê Nguyễn* says:

Sẽ có bài đấy sau tết nhé

29/01/2021 AT 2:48 CHIỀU

TRẢ LỜI



**thương** says:

ad làm bài giao tiếp stm32 với thẻ sd card giúp em với ad ơi

06/05/2021 AT 11:38 CHIỀU

TRẢ LỜI



**Lâm** says:

Em làm theo không biết sao chỉ có các ký tự từ 0 đến 12 là bên truyền và nhận giống nhau, còn từ 13 đến 19 thì khác nhau?

27/06/2021 AT 1:39 CHIỀU

TRẢ LỜI



**Khuê Nguyễn** says:

Nếu độ dài chuỗi mỗi lần truyền khác nhau thì nên xóa buffer trc khi nhận dữ liệu mới nhé

28/06/2021 AT 4:43 CHIỀU

TRẢ LỜI



**An** says:

anh ơi em làm theo nhưng trong phần u8\_SPI2\_Rxbuff chỉ có kí tự 0 là chữ 'g' còn lại từ 1-19 nó k hiện gì q

29/10/2021 AT 12:39 CHIỀU

TRẢ LỜI

**Khuê Nguyễn** says:

Em thử down phần example anh viết rồi chạy thử xem

05/11/2021 AT 1:41 CHIỀU

TRẢ LỜI

**Hieu Do** says:

Anh ơi anh thử làm stm32 giao tiếp SPI với Lora dc k q, em rất hóng hướng dẫn của anh!!

05/03/2022 AT 7:07 CHIỀU

TRẢ LỜI

**Khuê Nguyễn** says:

Anh sẽ cố gắng làm hết

14/03/2022 AT 5:31 CHIỀU

TRẢ LỜI

---

**Khoa** says:

Bạn có code example phần này không cho mình xin

04/07/2022 AT 10:22 SÁNG

TRẢ LỜI

**Khuê Nguyễn** says:

Bạn đọc bài tài liệu STM32 có tất cả example trên github nhé

28/07/2022 AT 11:38 CHIỀU

TRẢ LỜI

## Trả lời

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

**Bình luận \***

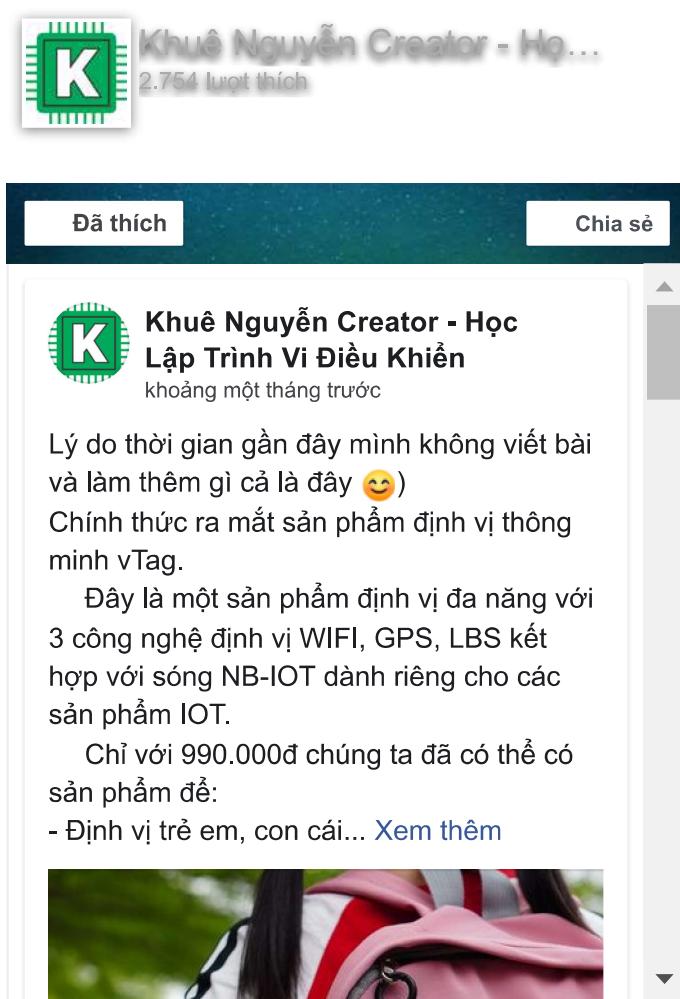
**Tên \***

**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 đẽ:

- 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 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

### 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

## 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



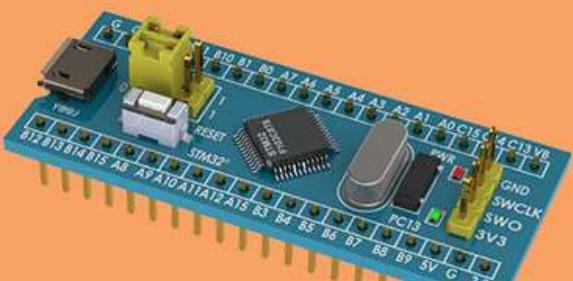
## Lập trình STM32F407 DAC chuyển đổi số sang tương tự

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**



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: Nghịen Lập Trình

Fanpage: Khuê Nguyên Creator

My Shop

## Thông Tin

Tác Giả

Chính Sách Bảo Mật



Copyright 2022 © Khuê Nguyễn