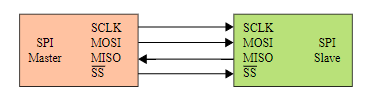
**Giao thức SPI (Serial Peripheral Interface):**

* Một chút lịch sử: SPI là một chuẩn đồng bộ nối tiếp do công ty Motorola thiết kế nhằm đảm bảo sự liên lạc giữa các vi điều khiển và thiết bị ngoại vi một cách đơn giản giá thành rẻ vào năm 1980 . Và thật không may nó đã trở thành một tiêu chuẩn thực tế mà mỗi chúng ta phải tìm hiểu.

**Lý do tại sao có uart và i2c mà ta vẫn phải có thêm spi:**

* Theo cá nhân mình thì do nó sài xung clock nếu ở một tầm nào đó nó sẽ chính xác hơn uart và còn với i2c thủ tục nó đơn giản hơn, cộng với truyền song công nên SPI tồn tại được là như vậy, còn khách quan thì do thích thì nó còn thôi.
* **Note**: khi dữ liệu chỉ có thể truyền theo một hướng (master->slave hay slave->master) trong một thời điểm thì đó được gọi là bán song công (half duplex), còn trong một thời điểm dữ liệu có thể truyền theo cả hai hướng được gọi là song công (full duplex)

Cấu tạo của một giao thức SPI:

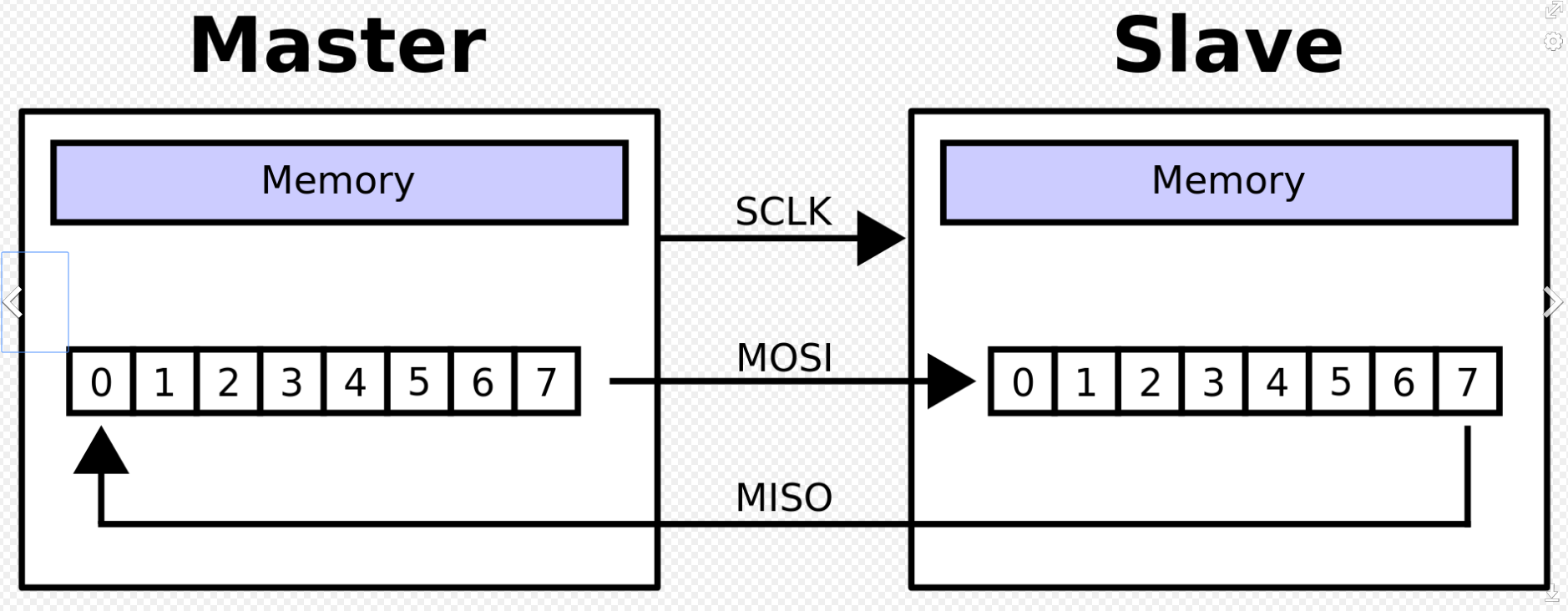


Vì là giao thức đồng bộ nên việc truyền nhận dữ liệu bằng tín hiệu xung clock phát ra từ vi điều khiển.

Giao thức SPI cơ bản giữa một master và slave cần 3 dây, với nhiều hơn 2 slave cần thêm dây slave select. Công dụng các dây:

* SCLK: Cung cấp xung clock.
* MOSI: Master out slave in, dành cho việc truyền dữ liệu từ master đến slave.
* MISO: Master in slave ou, dành cho việc nhận dữ liệu từ slavs về master.
* SS: slave select, chân chọn slave, active

Nguyên lý truyền nhận:



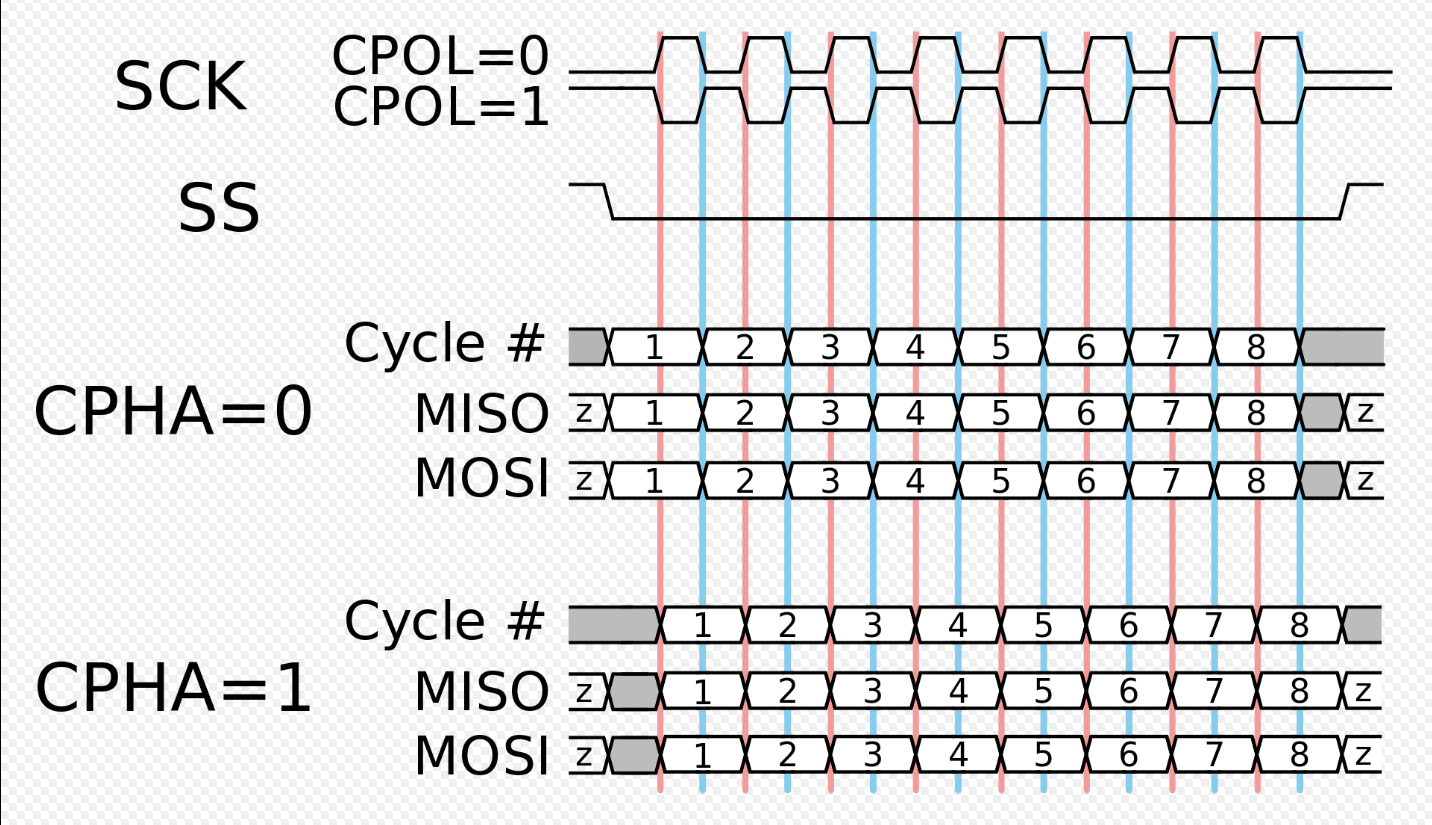
Khi master muốn nói chuyện với 1 slave nó sẽ kéo chân SS xuống mức 0. Sau đó quá trình thuền nhận bắt đầu.

Giả sử cả bên master và slave đều truyền nhận bằng thanh ghi 8 bit .

Bit có trọng số thấp nhất được truyền nhận tại mỗi cạnh xung, mỗi bit dữ liệu truyền nhận thì các bit lại được đổi trọng số một lần, sau khi xác nhận thanh ghi đã được truyền nhận đúng hai bên thay đổi thanh ghi và tiếp tục quá trình truyền nhận. Quá trình này diễn ra cho đến khi hết data cần truyền.

Thanh ghi dịch có thể là 8 bit hoặc một kich thước khác ví dụ 16 bit như trong màn hình cảm ứng, 12 bit trong các bộ chuyển đổi ADC hay DAC

**Clock và phase ( theo Motorola SPI block guide):**



CPOL loại clock :

CPOL=0 là loại clock bằng 0 ở trạng thái bình thường và lên 1 ở trên xung: xung mở đầu sẽ là xung cạnh lên, xung kết thúc là xung cạnh xuống.

CPOL=1 là loại clock bằng 1 ở trạng thái bình thường và xuống 0 khi xảy ra xung: xung mở đầu sẽ là xung cạnh xuống, xung kết thúc là xung cạnh lên.

CPHA cách lấy dữ liệu:

CPHA=0 : dữ liệu sẽ được lấy ở xung cạnh lên và thay đổi ở cạnh xuống, Trong chu kì đầu bit đầu cần nạp vào MISO trước khi clock diễn ra.

CPHA=1 : dữ liệu sẽ được lấy ra ở xung cạnh xuống của clock và thay đổi ở cạnh lên của xung clock. Trong chu kì cuối, slave nạp vào giá trị MOSI khi chân slave select chuyể trạng thái từ 0 sang 1.

Khi kết hợp 2 bit quy định CPOL là bit cao và CPHA là bit hấp lại ta được.

Với các dòng microchip PIC, ARM-base , với dòng (PIC32MX: set SPI mode bằng cách set CKP=1, và CKP, CKE config theo bảng

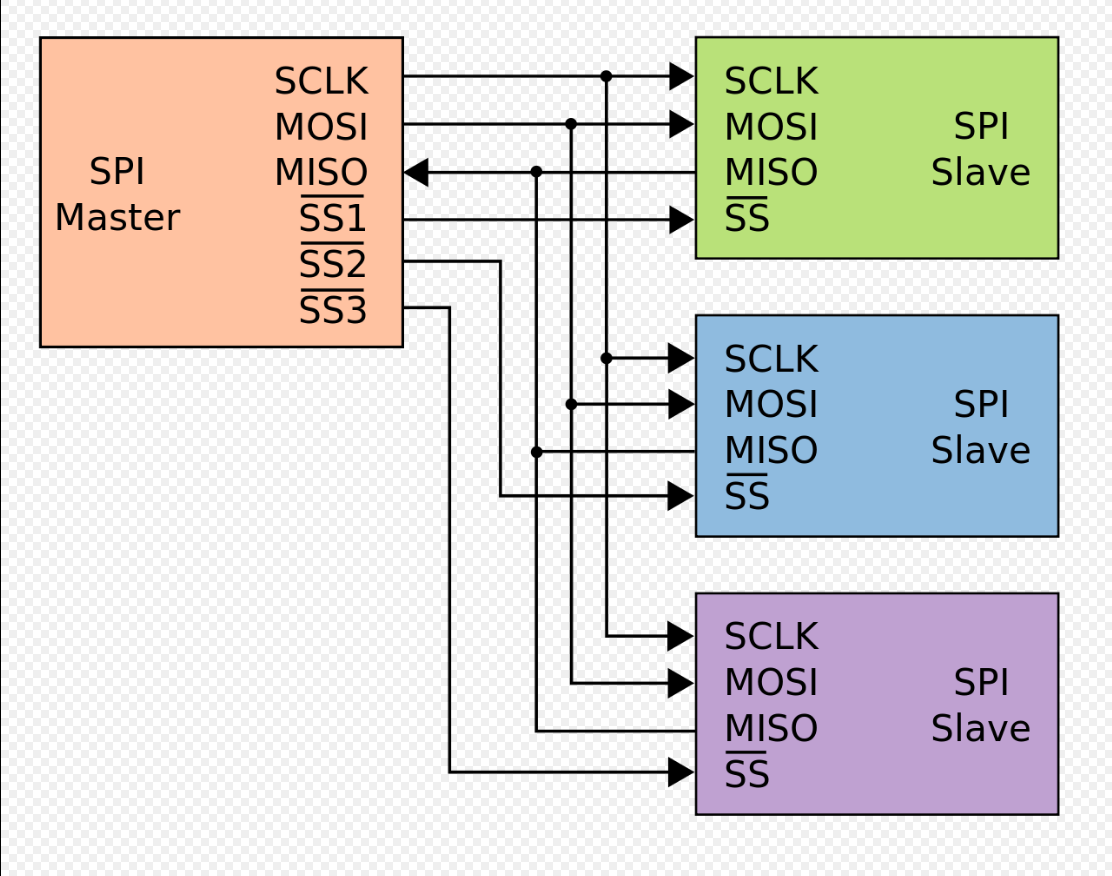
|  |  |  |  |
| --- | --- | --- | --- |
| **SPI mode** | **(CPOL/CKP)** | **(CPHA)** | **Cạnh xung lấy dữ liệu (CKE/NCPHA)** |
| 0 | 0 | 0 | 1(xung cạnh lên) |
| 1 | 0 | 1 | 0(xung cạnh xuống) |
| 2 | 1 | 0 | 1(xung cạnh lên) |
| 3 | 1 | 1 | 0(xung cạnh xuống) |

Với các vi điều khiển khác cần.

|  |  |  |
| --- | --- | --- |
| **Mode** | **CPOL** | **CPHA** |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |

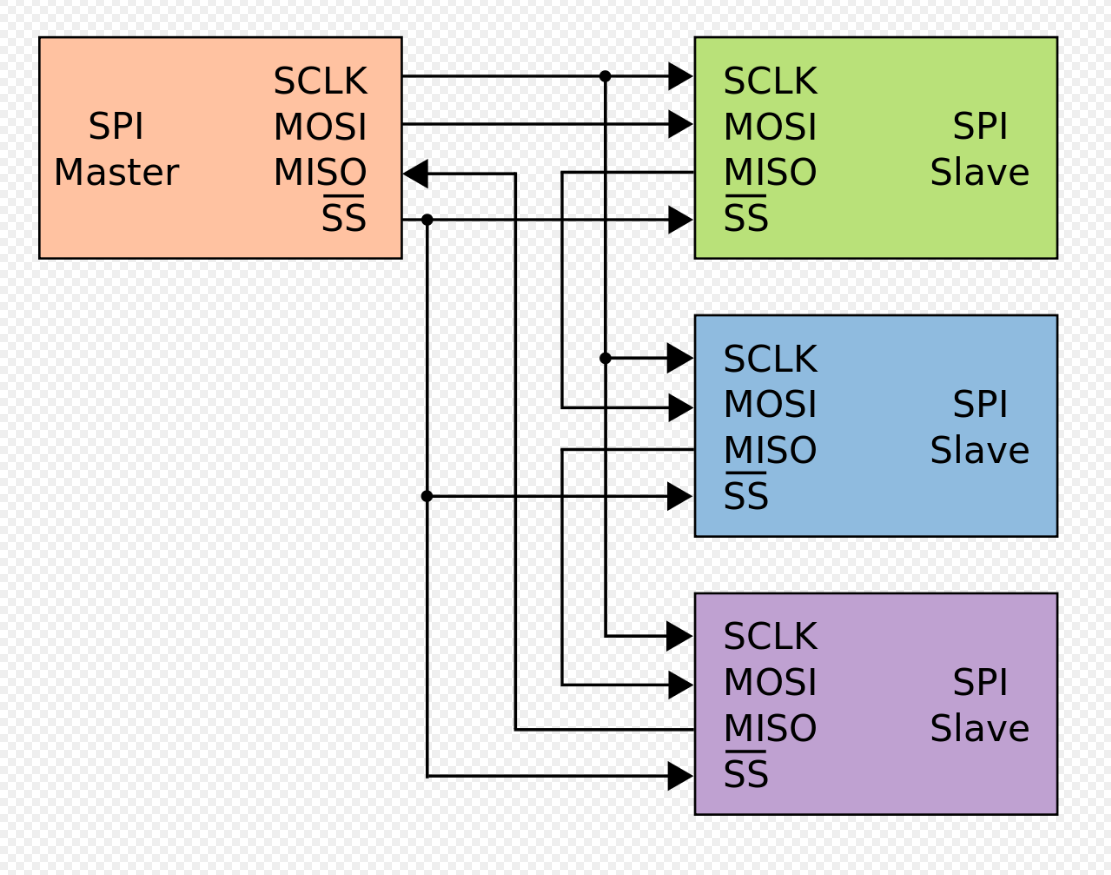
**Sơ đồ kết nối:**

1. Các slave độc lập với nhau( independent slave configuration).



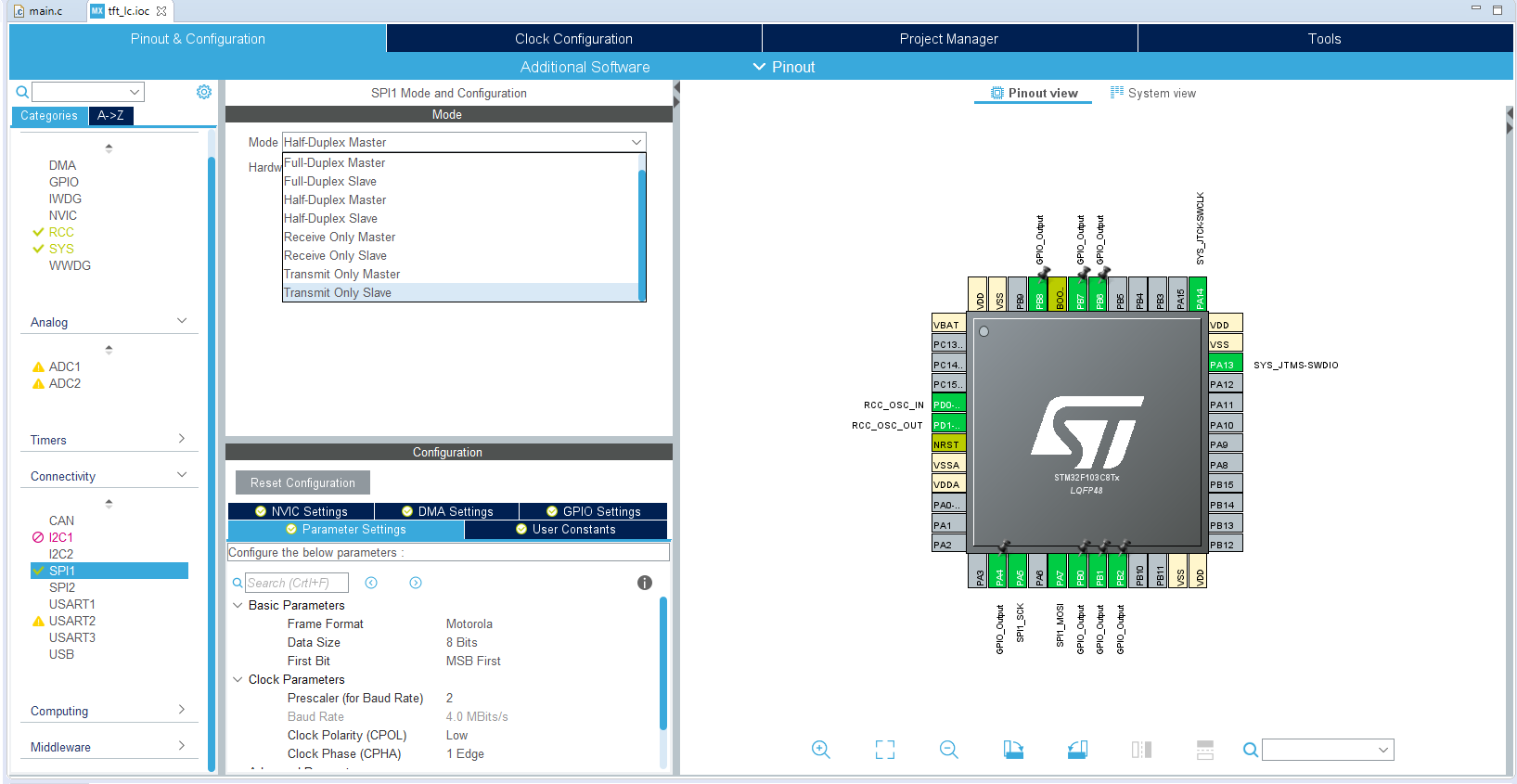
Trong kiểu nối này, mỗi con chip có một chân select nối thằng đến master chân này sẽ được kéo xuống mức 0 khi master muốn giao tiếp với slave đó. Điều kiện cần của các slave khi MOSI, hay MISO phải là cổng 3 trạng thái, nếu không phải thì cần phải có một ic đệm.

Để tiết kiệm chân chip select ta có thể dùng thêm các ic giải mã để tiết kiệm chân chip select.

1. **Mắc theo chuỗi** 

Trong kiểu nối dây này các dây slave select được nối với nhau , một MOSI của slave đầu được nối với master, MISO của slave thứ nhất nối với MOSI của slave thứ hai và cứ tiếp tục như vậy cho tới slave cuối cùng.

Cách giao tiếp dữ liệu. Sau khi slave nhận dữ liệu từ master, slave sẽ sử lý dữ liệu và truyền đi một bản copy tương tự đến slave thứ 2 cho đến slave cuối cùng hoặc khi chân chip select được đưa lên mức cao.

**Thực hành: cho stm32c8t6 giao tiếp với màn hình lcd st 7735** 

Config: chọn nguồn cấp xung clock từ thạch anh ngoài trong RCC, chọn debug bằng serial wire.

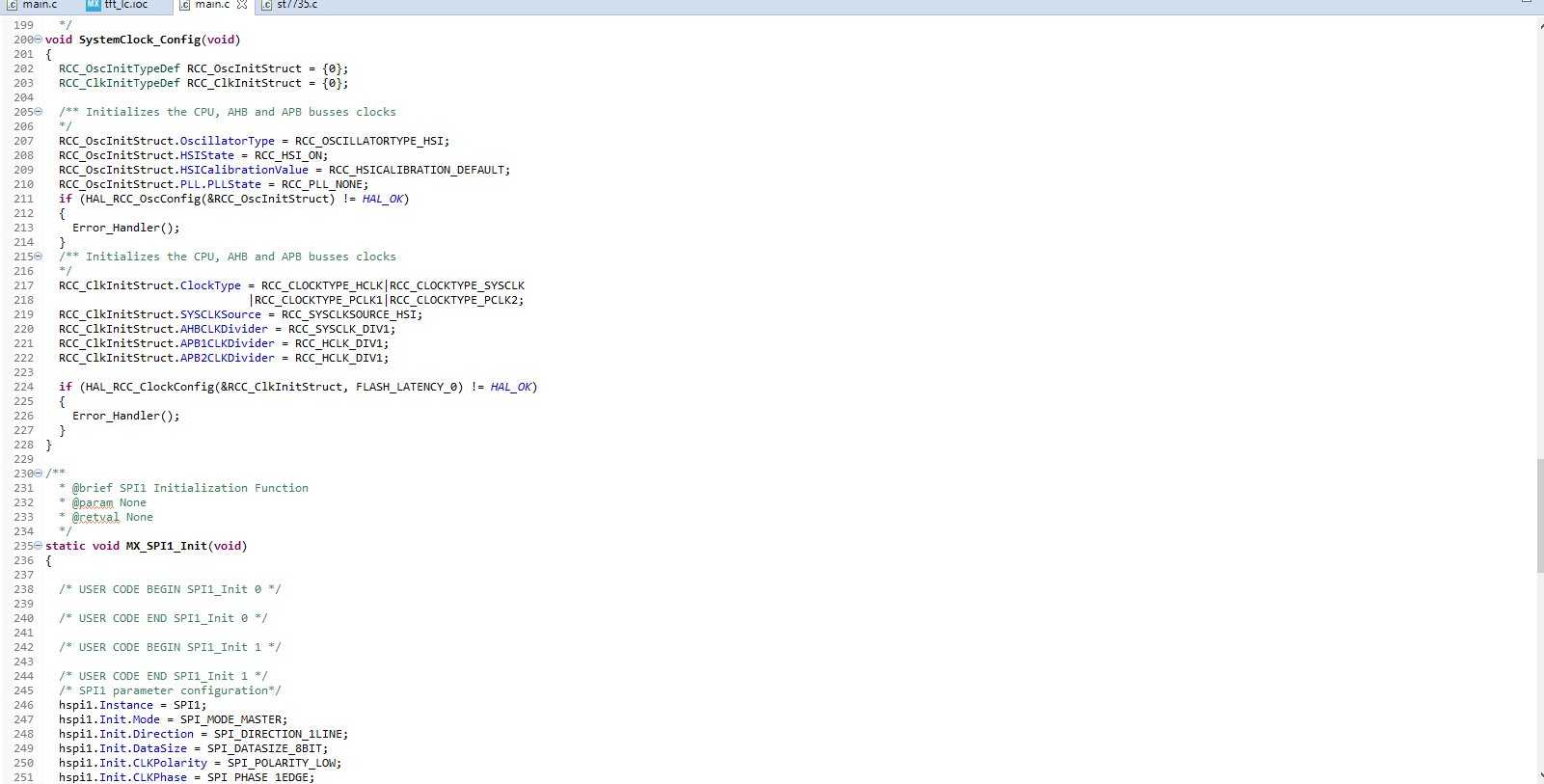
Vào mục connectivity:

Chọn mode :half-duplex Master vì chỉ cần giao tiếp theo 1 chiều.

Chọn theo frame Motorola vì nó nghĩ ra cái này.

Data size 8 bít, Fist bit MSB, prescaler 2 ( Cpol) low :idle state = 0, CPHA : 1 edge lấy mẫu tại cạnh lên.

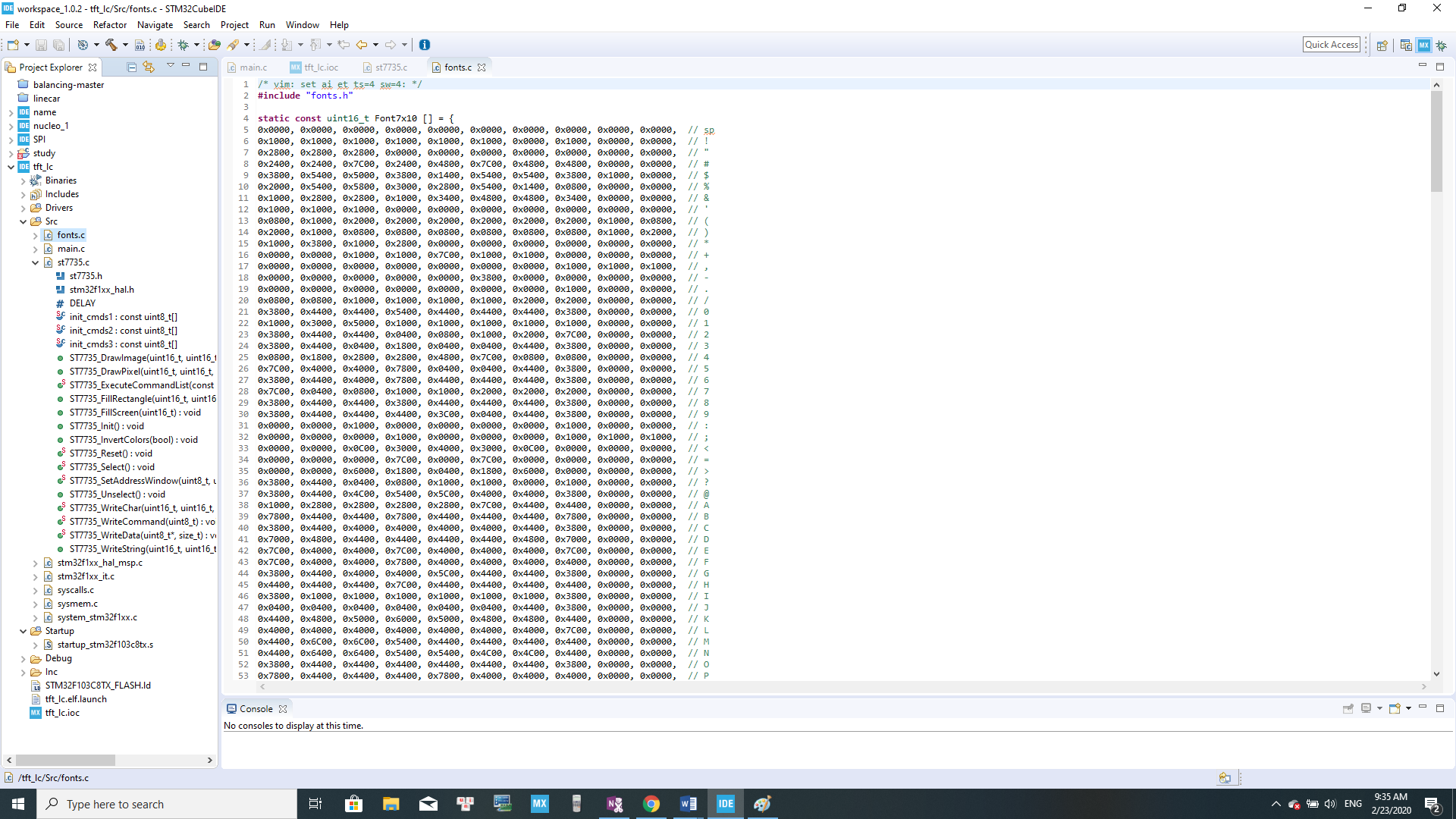
Sau khi generate code ta được:



Hàm systemClock\_Config và hàm MX\_SPI1\_Init là hàm để config các thông số cơ bản.

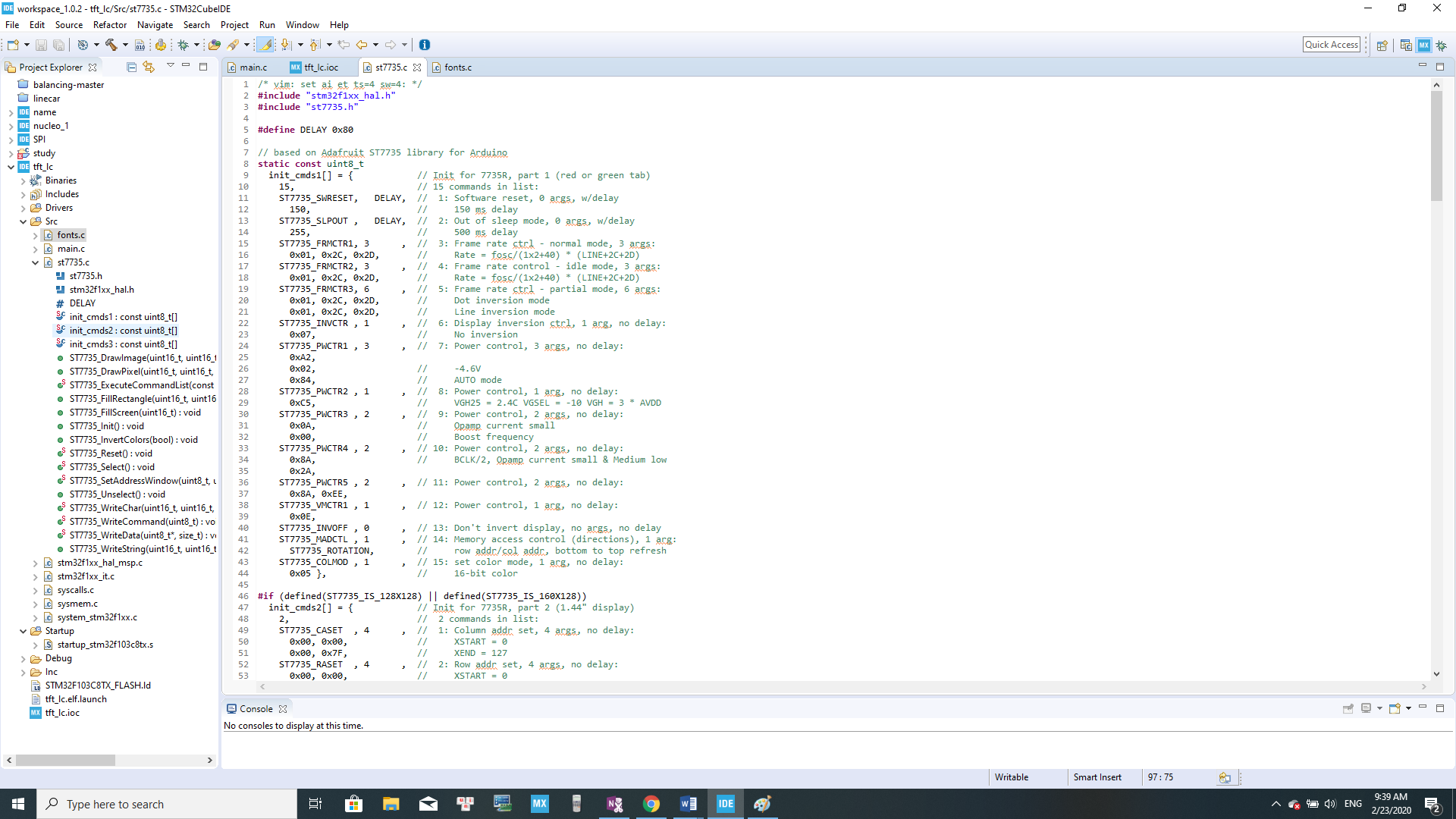
Để có thể điều khiển được các con led ( một điểm ảnh ta cần một đoạn code theo datasheet). Mà ở đây cần xây dựng một thư viện nên cách nhanh nhất là đi địa thư viện và học cách sài nó.

**Thư viện font.c**



Cách điều khiển các mà hình dạng điểm ảnh là đưa con trỏ đến vị trí mong muốn và hiện kí tự ra để làm được việc này ta cần một thư viện nữa.

**St7735.c**

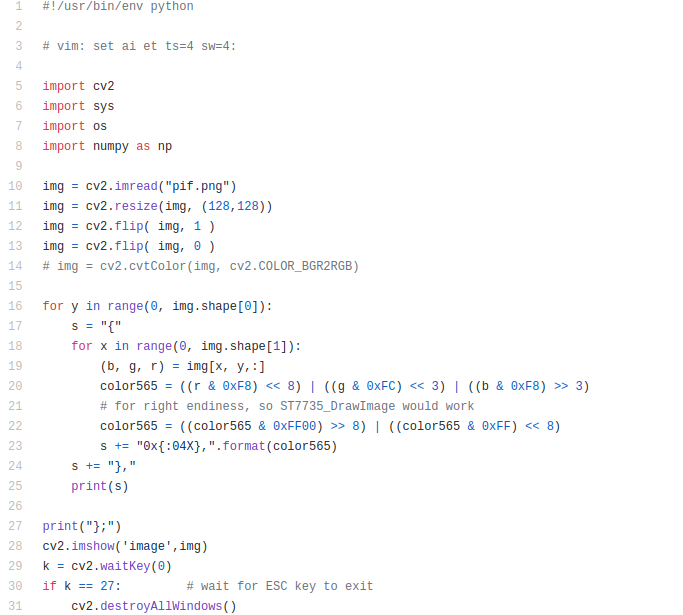


Để sử dụng màn hình này các bạn cần vào file st7735.h trong thư mục Inc và sửa các trạng thái như sau:



Trên board Chicken đã có sẵn sơ đồ nối, các em nối chân và setup project như board và chạy thử.

Để tạo ảnh, chúng ta sử dụng đoạn code gen\_img.py đã có trong github của anh (code python, đã yêu cầu cài opencv):



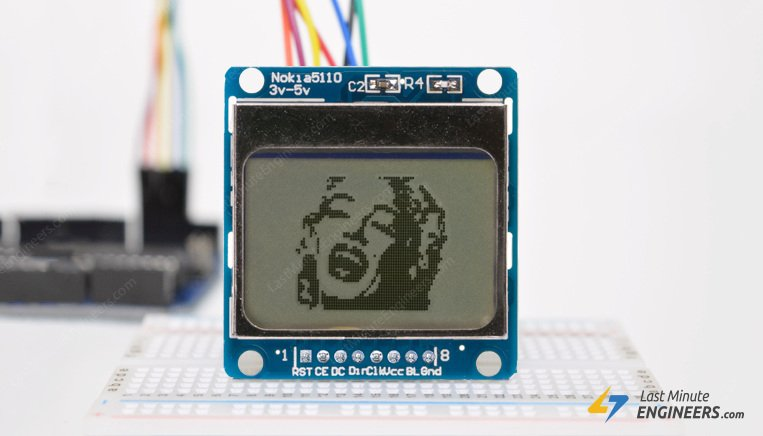
Phần giải thích sẽ có trong các bài tut OpenCV sau này, để sử dụng cần lưu một ảnh định dạng pif.png vào chung thư mục source và chạy file này (có thể dùng terminal hoặc IDE).

\*Bài tập: Hiển thị hình người yêu lên màn hình :)))))

Ví dụ về software SPI:

Trong thực tế, ta có thể sử dụng chân GPIO để thực hiện các giao tiếp UART, I2C, SPI,…

Trong đó anh có một chương trình mẫu là một game nhỏ trên màn hình Nokia5110:

**Màn Hình Nokia 5110**

Màn hình Nokia5110 sử dụng chip driver PCD8544 của Philips, driver này cho phép sử dụng và quản lý màn hình graphic 84\*48 pixel Nokia5110. Thông số của IC này như sau:

Single chip LCD controller/driver

•48 row, 84 column outputs

• Display data RAM 48 × 84 bits

• On-chip: – Generation of LCD supply voltage (external supply also possible) – Generation of intermediate LCD bias voltages – Oscillator requires no external components (external clock also possible).

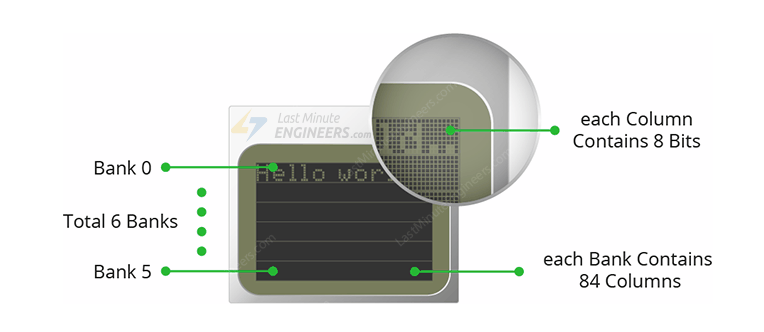
• External RES (reset) input pin

• Logic supply voltage range VDD to VSS: 2.7 to 3.3 V • Display supply voltage range VLCD to VSS – 6.0 to 8.5 V with LCD voltage internally generated (voltage generator enabled) – 6.0 to 9.0 V with LCD voltage externally supplied (voltage generator switched-off).

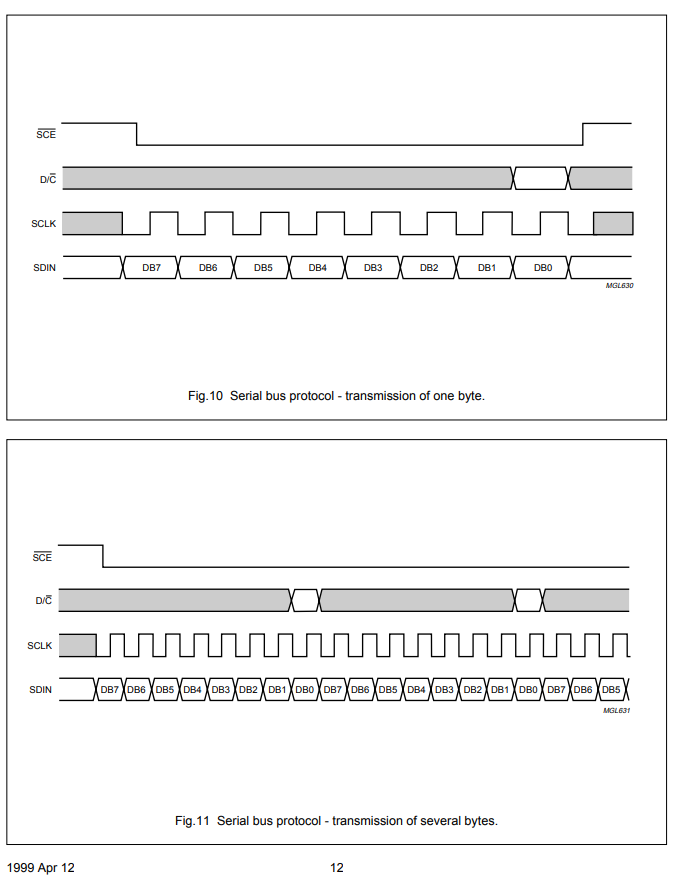
(Trích datasheet của IC).

**Memory Map**

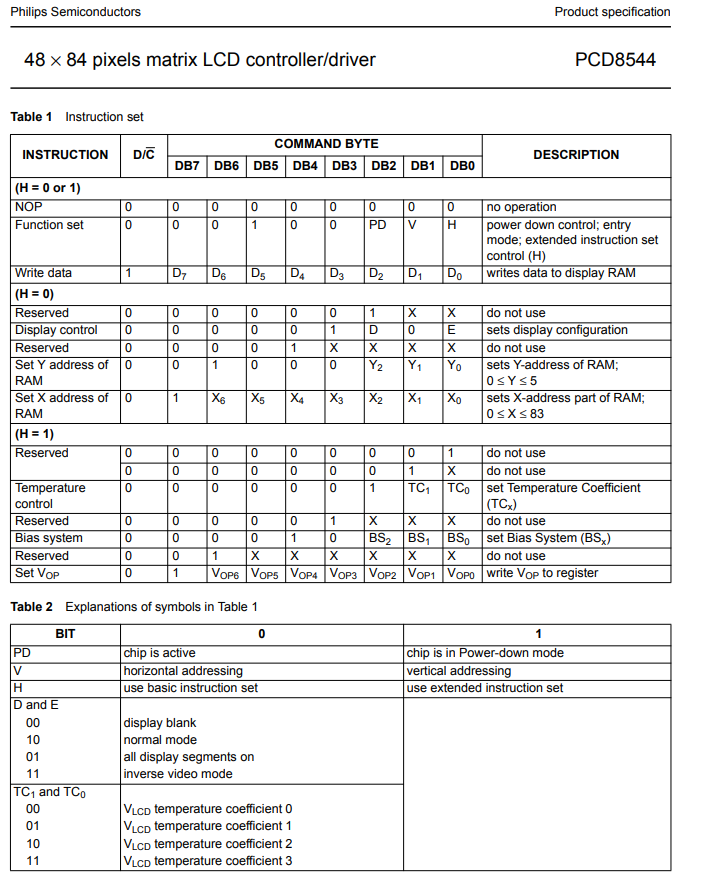
Bộ nhớ 84 \* 48 được tổ chức thành 6 băng thanh ghi với mỗi băng gồm 8bit mỗi hàng và 84 hàng quản lý toàn bộ các điểm ảnh của màn hình. Tổng số bit do đó là 8\*84\*48 = 4032 bits



Màn hình Nokia5110 sử dụng chip điều khiển PCD8544 của Philips, sử dụng giao thức SPI, trong đề tài này phương pháp software SPI được sử dụng.



Cấu trúc frame truyền của IC driver.



Bảng command của IC

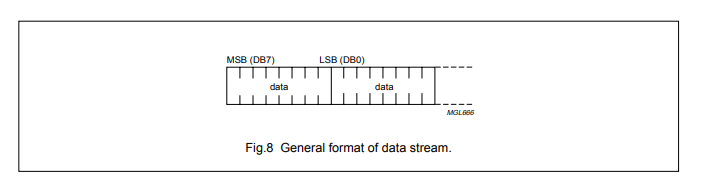
\*Giải thích ngắn gọn quy trình truyền:

Dựa theo cấu trúc frame truyền đã được kể ở trên, ta thấy quá trình truyền data có thể được mô tả như sau:

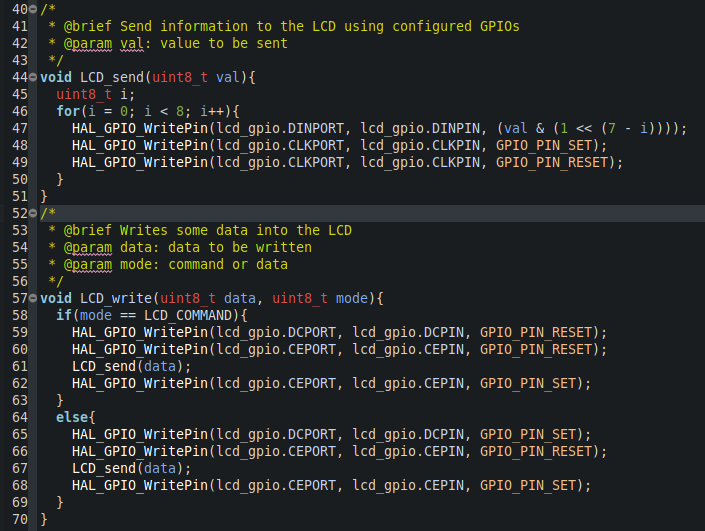
1. Để bắt đầu qúa trình truyền, chân CE của IC được kéo xuống thấp.

2. Quá trình truyền data được thực hiện khi có cạnh lên của tín hiệu chân CLK, IC sẽ đọc data ở chân DIN

3. Một frame truyền có độ dài 8bit (1byte) được ghi vào thanh ghi dịch của chip để đọc 1byte 1 lần.



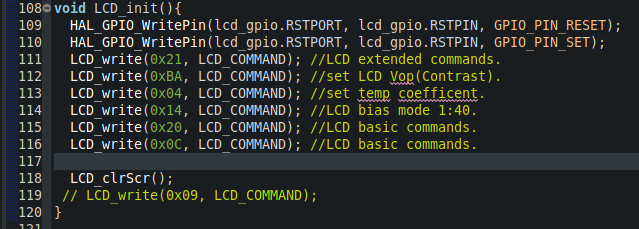
Tùy thuộc vào trạng thái của chân DC mà IC driver có thể hiểu lệnh được thực thi là commnad điều khiển IC hoặc là dữ liệu được đưa vào IC.

\*Hàm truyền dữ liệu:

Hàm LCD\_send sẽ gửi đi một byte dữ liệu, hàm LCD\_write sẽ thực hiện việc set pin DC để chọn mode hoạt động send data hoặc send command.

Để màn hình hoạt động, đầu tiên ta cần Init màn hình như sau:

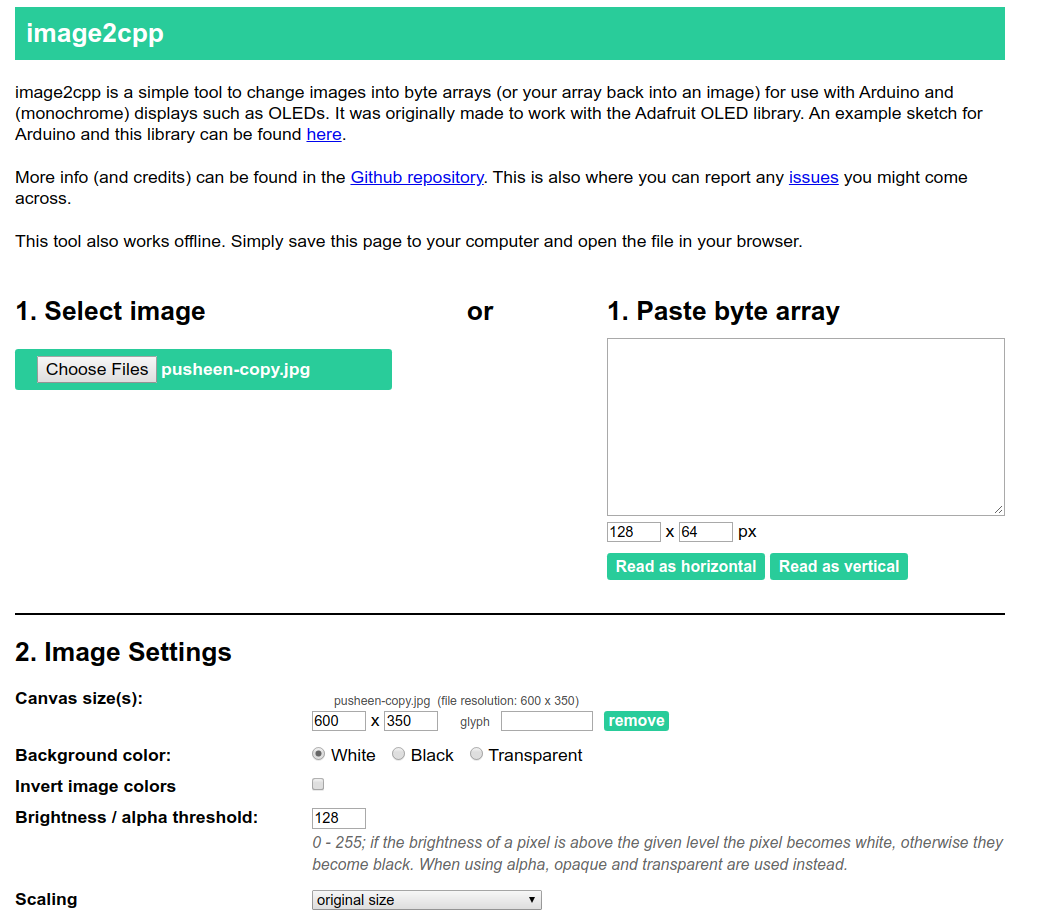
Các thông số init này dựa trên bảng command đã được đề cập.

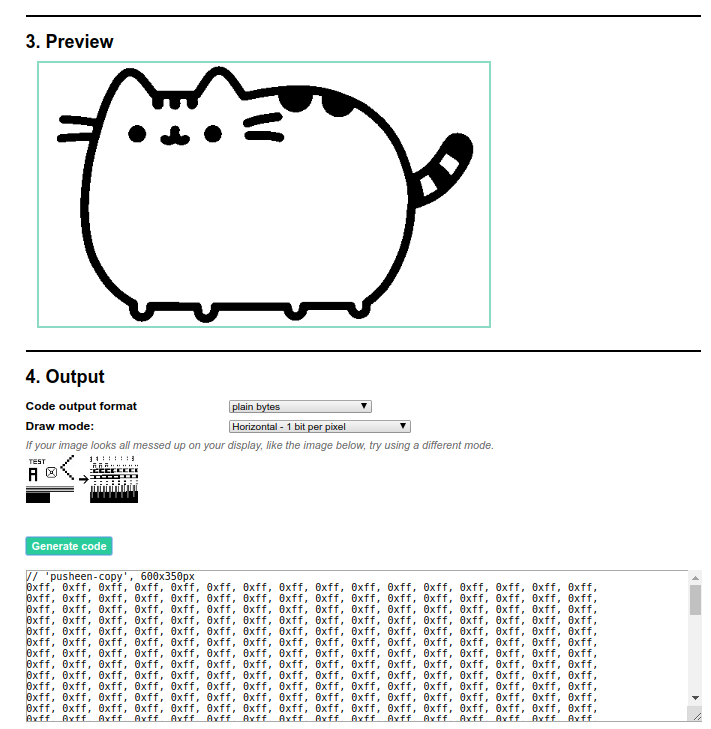


Để hiển thị ảnh trên màn hình:

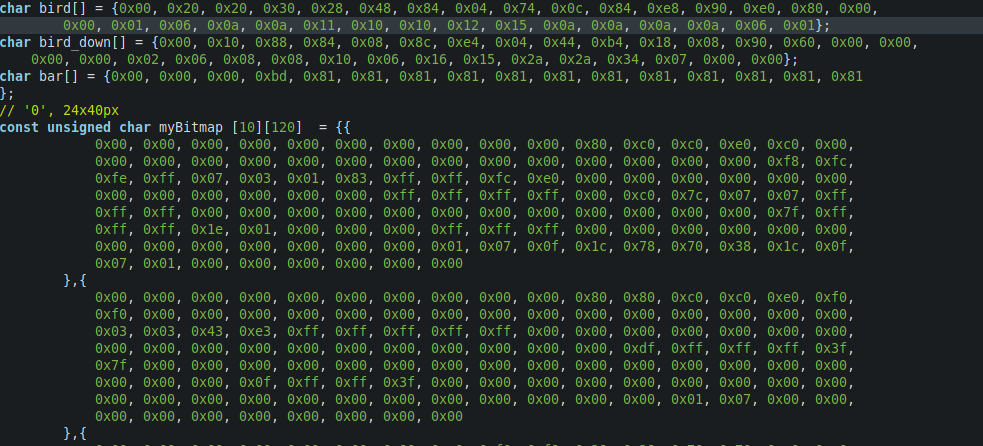
Khi truyền data vào màn hình, từng byte data sẽ được đưa vào lần lượt theo từng băng thanh ghi từ byte 0 đến byte 504, mỗi byte sẽ quy đinh mỗi pixel là sáng hay tối. Do đó để hiển thị ảnh trên màn hình ta cần lưu vào một buffer 504 byte để hiển thị ra màn hình

Để hiển thị một ảnh bitmap bất kì ta sử dụng công cụ Image2Cpp như sau:

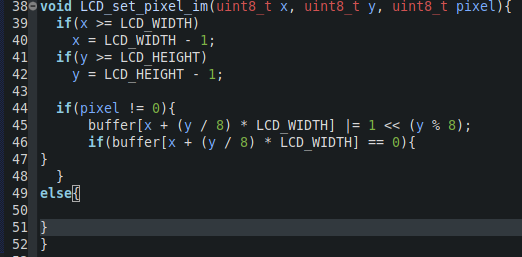




Các file ảnh này được lưu trong các mảng:

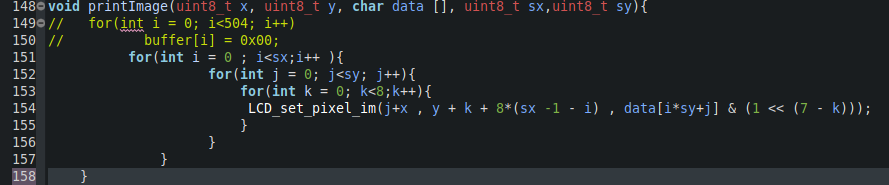


Để thực hiện in hình ra buffer ở một điểm xác định ta sử dụng hàm in từng pixel, sau đó ta in từng pixel ra buffer:



Như đã nói ở phần trên, buffer là một mảng 504 phần tử uint8\_t, mỗi phần tử thể hiện 8 bits của mỗi thanh ghi trong băng thanh ghi. Để truy cập đến pixel có tọa độ (x,y), trong đó x có kích thước 84, y là số kí hiệu băng thanh ghi, gồm 1 - > 6, mỗi băng thanh ghi có 8 vị trí, do đó để chuyển đến vị trí (x,y) ta cần truy cập vào thanh ghi thứ x\*84 + y/8 (phép chia lấy phần nguyên) và sau đó thực hiện phép dịch bit 1 << (y%8) để truy cập đến phần tử thứ y. Như trong code ở trên đã thể hiện phương pháp thực hiện này.

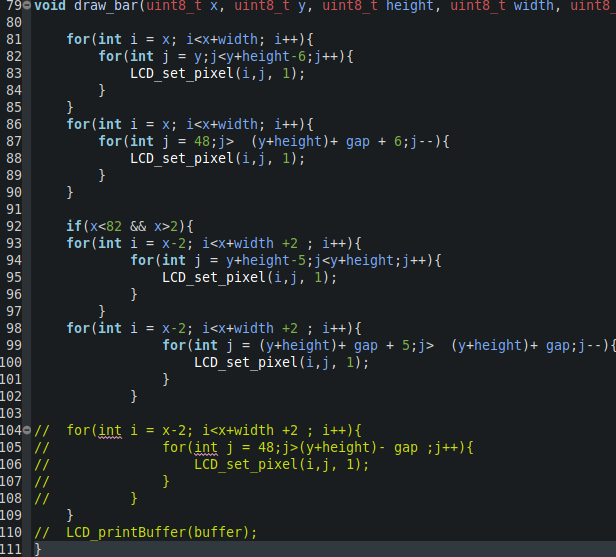
Sau đó để thực hiện in ảnh trong buffer ở vị trí (x,y) ta sử dụng hàm sau:



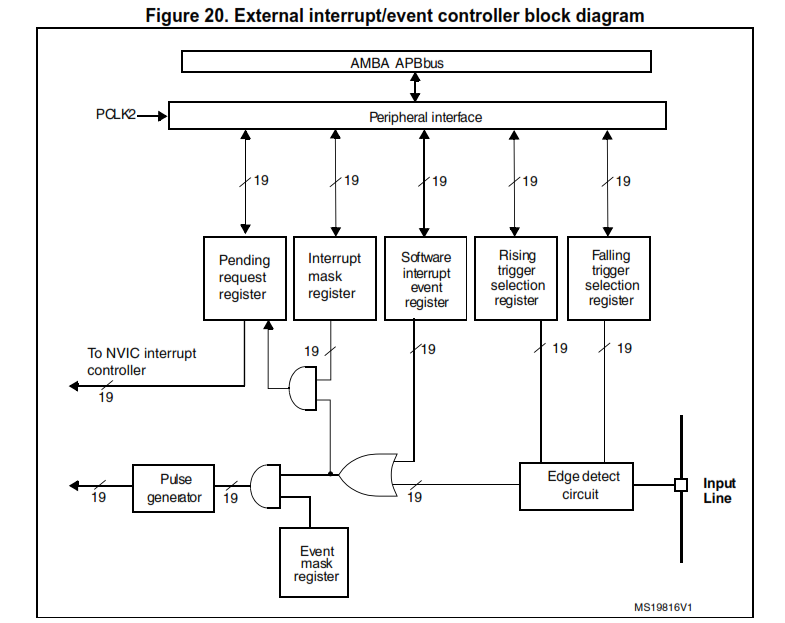
Input của hàm là vị trí ảnh được in (x,y), ảnh được đưa vào là mảng char data [], 2 thông số kích thước ảnh sx và sy, để thực hiện in ảnh ta cần set từng pixel ảnh vào từng vị trí trong buffer. 2 vòng lặp đầu tiên quét qua tất cả các pixel của ảnh, vòng lặp trong cùng quét qua từng vị trí trong thanh ghi của buffer, trong đó từng pixel của data cũng được lấy ra nhờ câu lệnh data[i\*sy + j] & (1<<7 - k)).

Hàm vẽ thanh chắn của game:

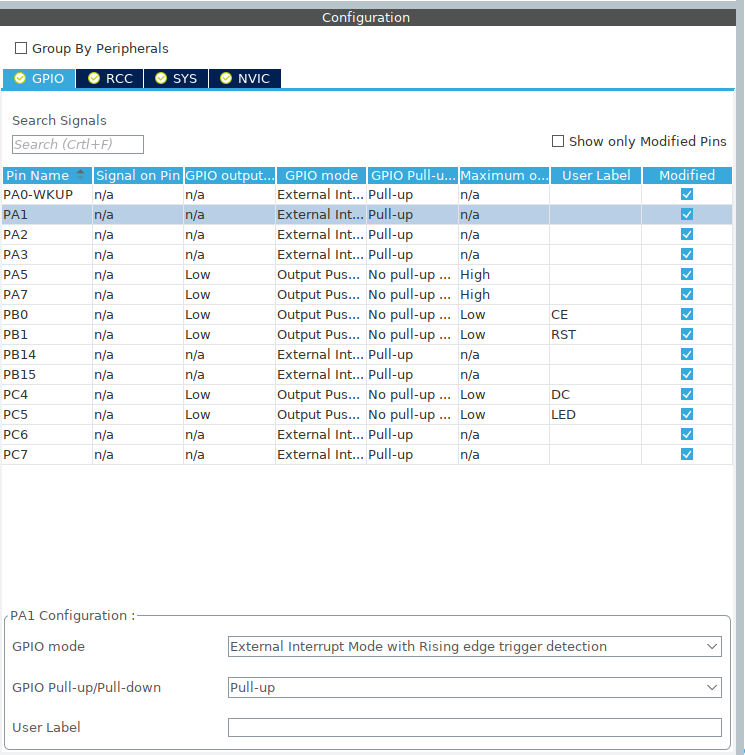
Hàm này vẽ các hình chữ nhật biểu thị cho các thanh chắn của game Flappy Bird, các thông số cần sử dụng là vị trí thanh chắn, chiều cao thanh chắn, độ rộng thanh chắn, khoảng giữa 2 thanh chắn để nhân vật vượt qua.



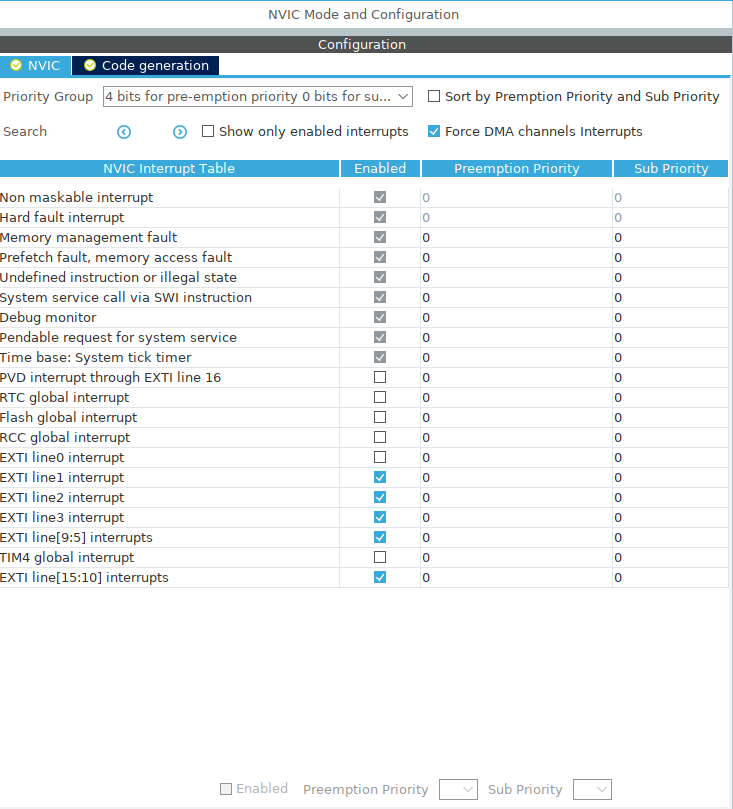
1. **Điều Khiển Nhân Vật:**

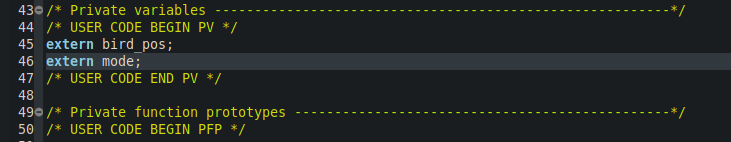


Để điều khiển nhân vật, External interrupt/event controller (EXTI) của STM32 được sử dụng, cho phép kiểm soát ngắt trên chân GPIO của STM32. Các GPIO này được setup như trong hình dưới, được nối với các button đã được chống rung hardware và được kéo trở pull up nội.



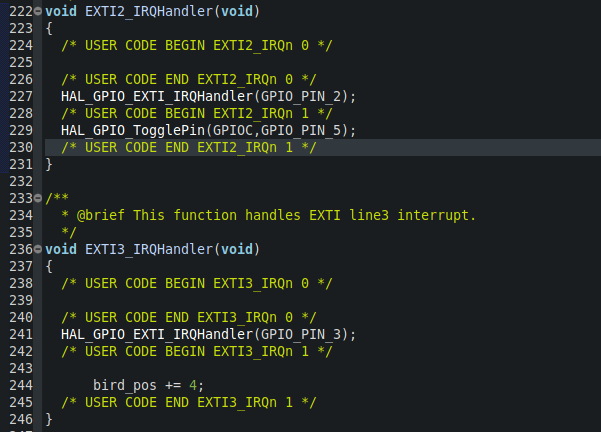
Bảng ưu tiên ngắt NVIC như sau:





Trong file stm32f1xx\_it.c có các hàm xử lý ngắt EXTI như sau

Các biến toàn cục bird\_pos để điều chỉnh vị trí của nhân vật game và mode để điều khiển mode chạy của máy chơi game.



**Các hàm EXTIx\_IRQHandler để thực hiện xử lý ngắt từ EXTI**

**A picture containing indoor

Description automatically generated**

Sản phẩm :3

Dưới đây là toàn bộ chia sẻ về SPI .

Bài tập kết thúc. Các bạn hãy thử làm mà hình hiện tên mình lên nhé.

Note: sau khi nắm rõ các cách thức hãy sáng tạo cho mình một hình ảnh đặc biệt.

Link code tham khảo code :<https://github.com/DangLamTung/stm32code/tree/master/tft_lc/Src?fbclid=IwAR0HYTLboxxsqee5T-nlIi4xsgeZ0rSf4f-eFI03qezALd9iK-0pxIZ3CL0>

<https://github.com/DangLamTung/stm32Gamer>

Link cho datasheet: <https://html.alldatasheet.com/html-pdf/326213/SITRONIX/ST7735/383/1/ST7735.html>