BÁO CÁO MÔN HỌC

Real Time System

CÀI ĐẶT VÀ SỬ DỤNG HĐH COOCOX TRÊN BO MẠCH

STM32F4

**Lịch sử thay đổi**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Ver.** | **Reason** | **Owner** | **Approve** | **Date** |
| 10/12/2014 | 0.1 | Khởi tạo template | N.Đ.Tuấn |  |  |
|  |  |  |  |  |  |

Mục lục

[BÁO CÁO MÔN HỌC 1](#_Toc407443454)

[Real Time System 1](#_Toc407443455)

[Lịch sử thay đổi 2](#_Toc407443456)

[Mục lục 3](#_Toc407443457)

[1 GIỚI THIỆU 4](#_Toc407443458)

[1.1 MÔ TẢ CHUNG VỀ BÁO CÁO 4](#_Toc407443459)

[1.2 MỤC ĐÍCH TÀI LIỆU 4](#_Toc407443460)

[1.3 PHẠM VI NGHIÊN CỨU 4](#_Toc407443461)

[1.4 THUẬT NGỮ 4](#_Toc407443462)

[1.5 TÀI LIỆU THAM KHẢO 4](#_Toc407443463)

[2 NỘI DUNG 4](#_Toc407443464)

[2.1 GIỚI THIỆU HĐH COOCOX 4](#_Toc407443465)

[2.1.1 Lịch sử phát triển của CooCox 4](#_Toc407443466)

[2.1.2 Đặc tính của CooCox 4](#_Toc407443467)

[2.1.3 Quản lý tiến trình trong CooCox 5](#_Toc407443468)

[2.1.4 Bộ lập lịch – cơ chế lập lịch của CooCox 5](#_Toc407443469)

[2.2 CÀI ĐẶT CÁC CÔNG CỤ 5](#_Toc407443470)

[2.2.1 Công cụ 5](#_Toc407443471)

[2.2.2 Cài đặt 5](#_Toc407443472)

[2.2.3 Lưu ý khi cài đặt và các lỗi thường gặp 5](#_Toc407443473)

[2.3 CÀI ĐẶT HĐH COOCOX 5](#_Toc407443474)

[2.3.1 Các công cụ cần thiết 5](#_Toc407443475)

[2.3.2 Quá trình cài đặt 5](#_Toc407443476)

[2.3.3 Load firmware lên board STM32F4 5](#_Toc407443477)

[2.4 CHẠY CHƯƠNG TRÌNH DEMO TRÊN STM32F4 5](#_Toc407443478)

[2.4.1 Mô tả chương trình 5](#_Toc407443479)

[2.4.2 Chạy Demo – Debug 5](#_Toc407443480)

[2.4.3 Đánh giá 5](#_Toc407443481)

[3 KẾT LUẬN 5](#_Toc407443482)

# GIỚI THIỆU

## MÔ TẢ CHUNG VỀ BÁO CÁO

## MỤC ĐÍCH TÀI LIỆU

## PHẠM VI NGHIÊN CỨU

## THUẬT NGỮ

## TÀI LIỆU THAM KHẢO

# NỘI DUNG

## GIỚI THIỆU HĐH COOCOX

### Lịch sử phát triển của CooCox

### Đặc tính của CooCox

* Ưu nhược điểm ---

### Quản lý tiến trình trong CooCox

### Bộ lập lịch – cơ chế lập lịch của CooCox

## CÀI ĐẶT CÁC CÔNG CỤ

### Công cụ

Môi trường phát triển: Windows

Các công cụ cần thiết :

**CoIDE**: <http://coocox.org/software.html>

IDE do chính nhóm phát triển CooCox đưa ra. Dựa trên nền tảng IDE quen thuộc là eclipse.

**CoFlash**: <http://coocox.org/software.html>

Tool dùng để download firmware vào mạch STM32F4. Tool này có thể kết hợp với CoIDE để download và debug ngay trên CoIDE.

**GCC ARM**:  <https://launchpad.net/gcc-arm-embedded/+download>

Compiler dành cho stm32 (sử dụng phiên bản 32bit). Cần cài đặt trước khi

**ST-LINK driver**: Đây là driver cho cổng usb của board STM32F4. Cần phải cài đặt driver này, nếu không CoFlash sẽ không kết nối được với board.

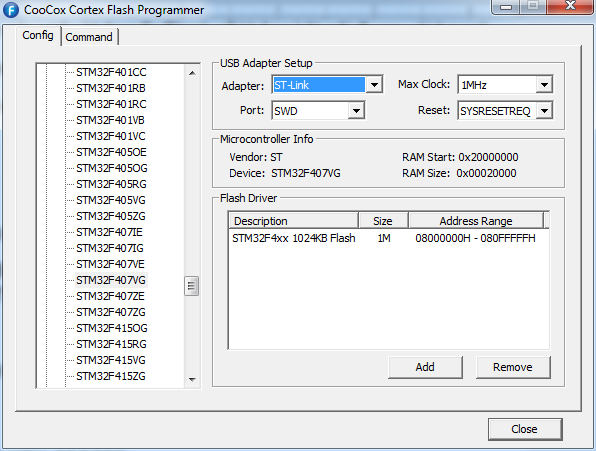
Phiên bản win7,xp: <http://www.st.com/web/en/catalog/tools/PF258167>

Phiên bản win8: <http://www.st.com/web/en/catalog/tools/PF259459>

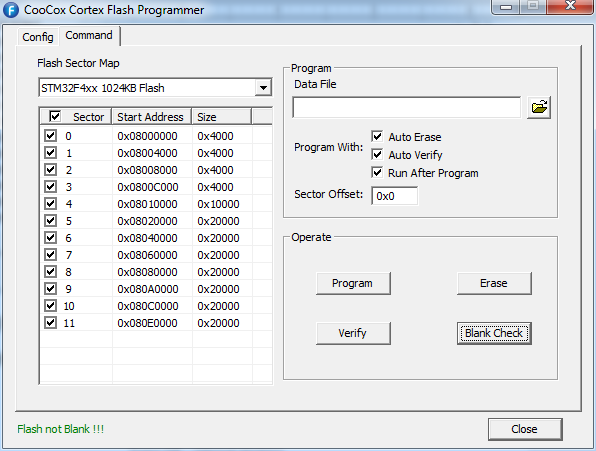
### Cài đặt

Cài đặt như các phần mềm khác trên windows.

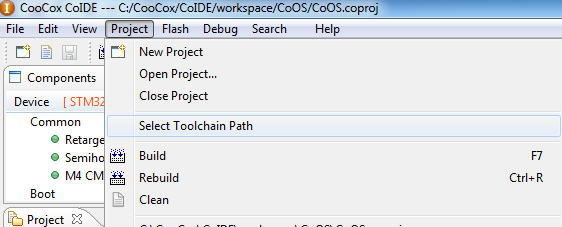
Sau khi cài đặt tất cả xong, Cắm board STM32F4 vào cổng USB. bật CoFlash lên, chọn chip STM32F407VG, adapter là ST-Link như hình dưới:



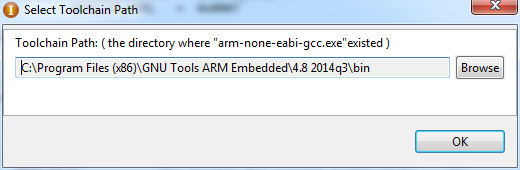
Sau đó sang tab Command, nhấn vào “blank check”, nếu hiện “Flash not blank” hoặc “Flash is blank” thì có nghĩa là kết nối thành công CoFlash và board STM32F4.



Sau đó bật CoIDE để cấu hình tool chain. Chọn “Project > Select Toolchain Path”



Sau đó tìm đường dẫn đến thư mục “bin” trong thư mục cài GCC ARM toolchain. Ví dụ:



Đến đây là xong các bước cấu hình, bắt đầu vào code.

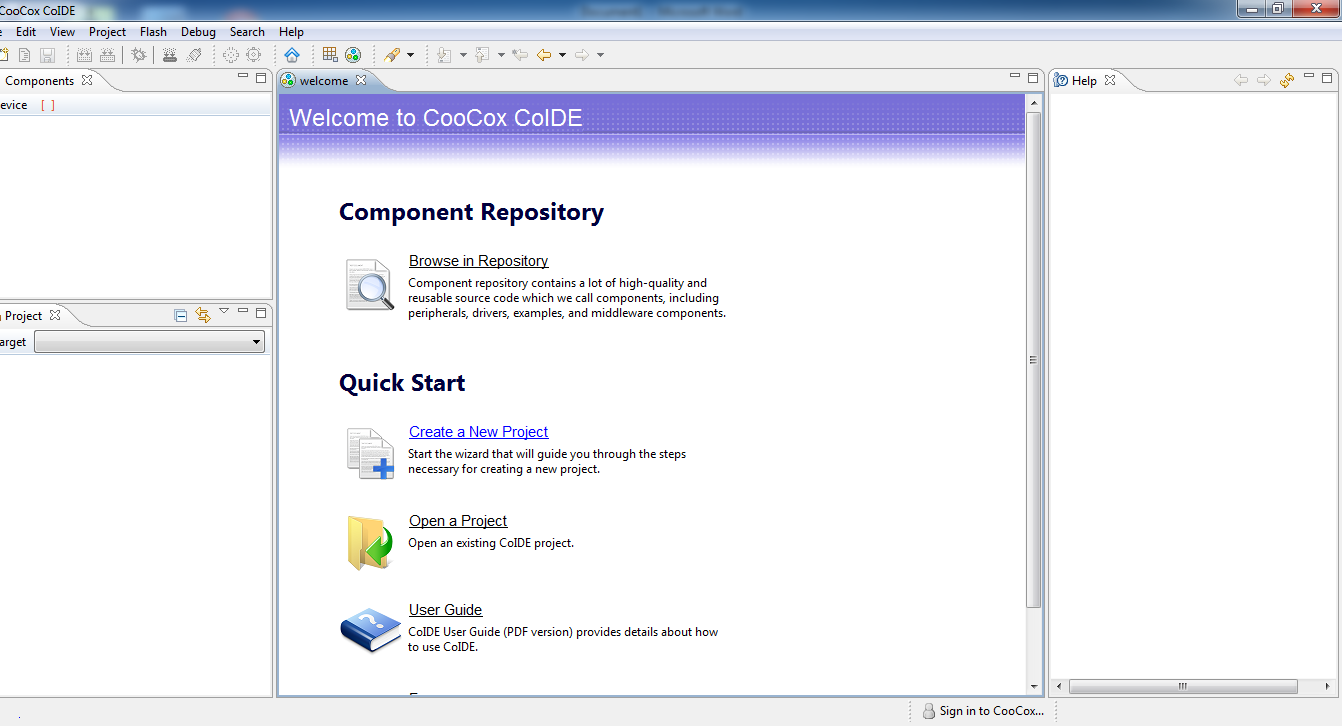
### Lưu ý khi cài đặt và các lỗi thường gặp

* Phiên bản trong hướng dẫn là CoIDE 1.7.7, hiện tại đã có phiên bản 2 (beta) nhưng khuyến nghị dùng bản 1.7.7 cho đến khi có phiên bản 2 chính thức.
* Nếu sau khi cài driver mà cắm board không nhận thì cần restart máy tính.
* Board hoạt động không tốt với cổng USB 3.0, nên dùng cổng 2.0 để kết nối.

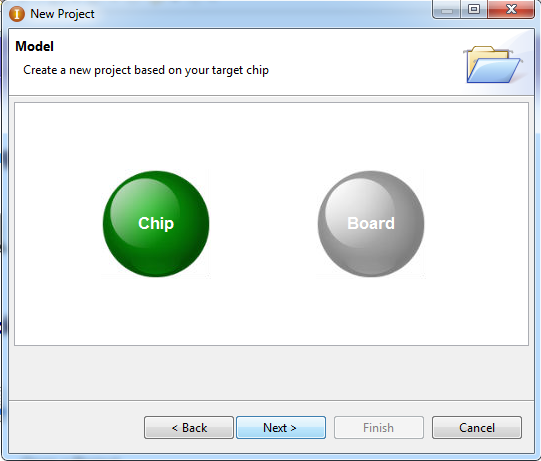
## CÀI ĐẶT HĐH COOCOX

### Lấy mã nguồn:

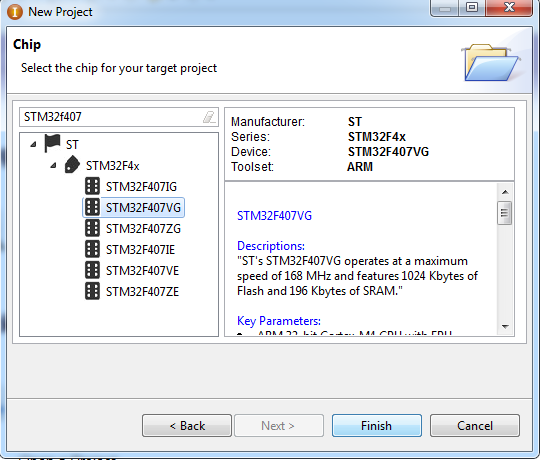
1. Tạo mới project.



Nhập tên, chọn “CHIP”



Gõ chip STM32F407VG



1. Lấy mã nguồn CooCox:

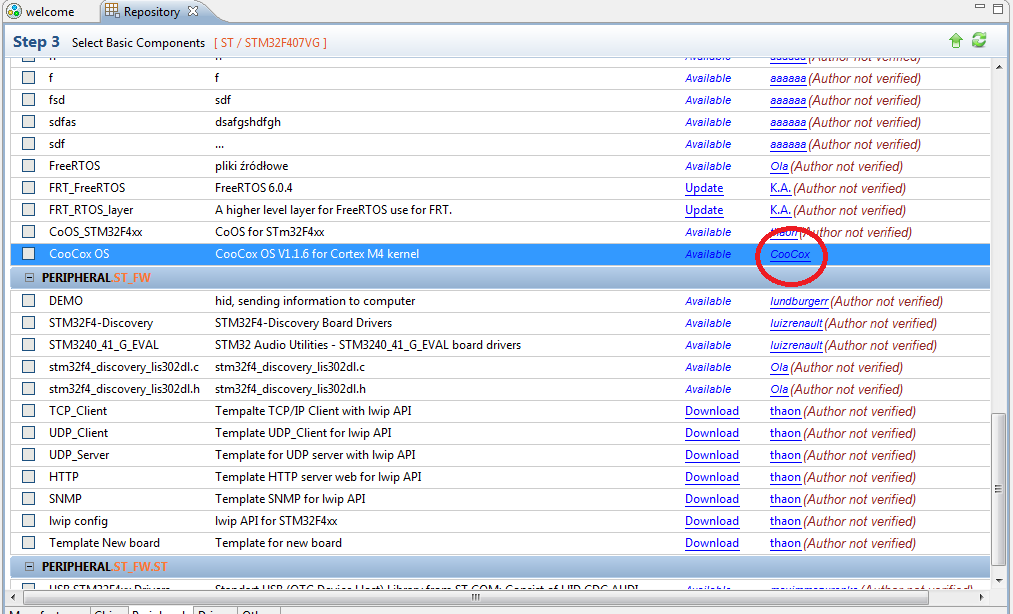
Chuyển qua Tab “repository”. Màn hình Repository là nơi chứa các đoạn mã nguồn đã được viết sẵn. Nguồn từ cộng đồng, vì thế nên không đảm bảo về mặt chất lượng code. Muốn code “sạch” và “chuẩn” thì nhìn vào phần tác giả, nếu là “CooCox” thì đảm bảo về mặt chất lượng code. Các đoạn code khác lấy về có thể cần phải chỉnh sửa mới hoạt động được.

Hoặc có thể lấy từ github của nhóm phát triển CoOS:

<https://github.com/coocox/CoOS>

Hoặc có thể lấy từ github của nhóm:

<https://github.com/trunghieuhust/CooCox-for-STM32F4-discovery-board> (branch *CoOS\_only*)



1. Lấy mã nguồn CMSIS cho board STM32F4:

CMSIS (*Cortex Microcontroller Software Interface Standard*) là 1 lớp trừu tượng của phần cứng (*vendor-independent hardware abstraction layer*) cho dòng chip Cortex M và định nghĩa giao diện debug. Cần phải có mã nguồn phần này thì mới thao tác được với board mạch.

Cần tích vào 2 phần : **M4 CMSIS Core** và **CMSIS BOOT**

1. Các mã nguồn bổ sung khác:

Để board chạy được thì chỉ cần 3 phần mã nguồn nêu trên là đủ. Tuy nhiên nếu ta cần thêm các tính năng như printf hay các driver cho ngoại vi thì cần lấy thêm các đoạn mã nguồn khác.

Ở đây sẽ ví dụ về lấy mã nguồn để viết chương trình in ra đoạn text qua semihosting. Chương trình có 2 tiến trình, in ra console kí tự A và B.

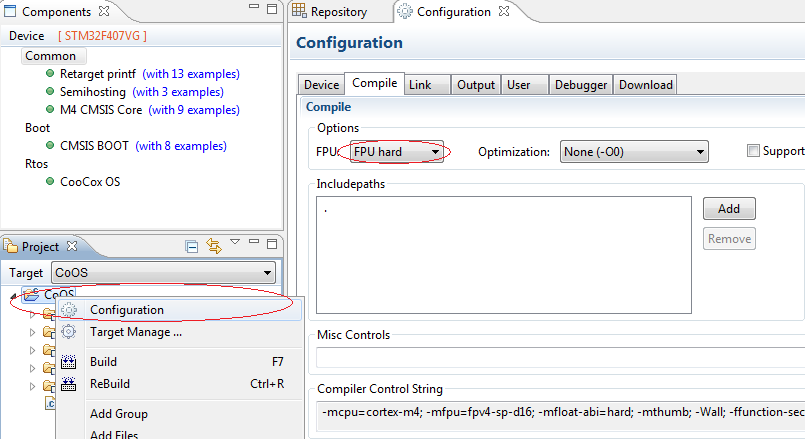
Các đoạn code cần dùng:

**Retarget printf** : cung cấp implement của hàm printf.

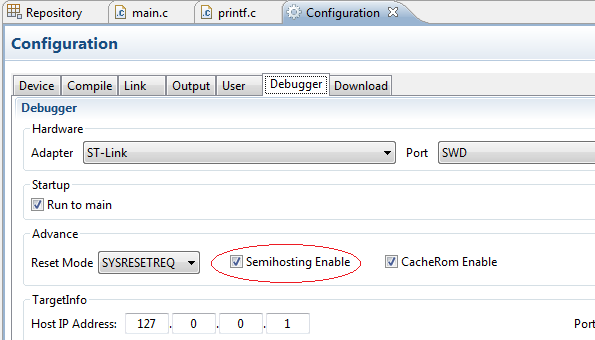
**Semihosting** : implement của hàm getchar/putchar ra đầu ra semihosting.

Sửa 1 số thông số cho chương trình dịch:

FPU đổi về FPU Hard



Bật chế độ Semihosting trong debug.



Trong file /stdio/printf.c , thêm hàm **SH\_SendChar(c);** vào trong hàm **PrintChar(c);**

Thêm vào hàm main.c

Sửa hàm main.c

**#include** "semihosting/semihosting.h"

**#include** "stm32f4xx.h"

**#include** "CoOS.h"

**#include** <stdio.h>

**#define** STACK\_SIZE\_TASK\_A 128

**#define** STACK\_SIZE\_TASK\_B 128

OS\_STK task\_A\_stk[STACK\_SIZE\_TASK\_A];

OS\_STK task\_B\_stk[STACK\_SIZE\_TASK\_B];

**void** **taskA**(**void** \*pdata) {

**while** (1) {

**printf**("A\n");

CoTickDelay(100);

}

}

**void** **taskB**(**void** \*pdata) {

**while** (1) {

**printf**("B\n");

CoTickDelay(100);

}

}

**int** **main**(**void**) {

CoInitOS();

CoCreateTask(taskA, 0, 0, &task\_A\_stk[STACK\_SIZE\_TASK\_A - 1],

STACK\_SIZE\_TASK\_A);

CoCreateTask(taskB, 0, 1, &task\_B\_stk[STACK\_SIZE\_TASK\_B - 1],

STACK\_SIZE\_TASK\_B);

CoStartOS();

**while** (1) {

}

}

**Lưu ý: Hàm printf() không in được 1 kí tự. Nếu in 1 kí tự sẽ báo lỗi lúc build.**

**Ví dụ: printf(“A”) -> báo lỗi.**

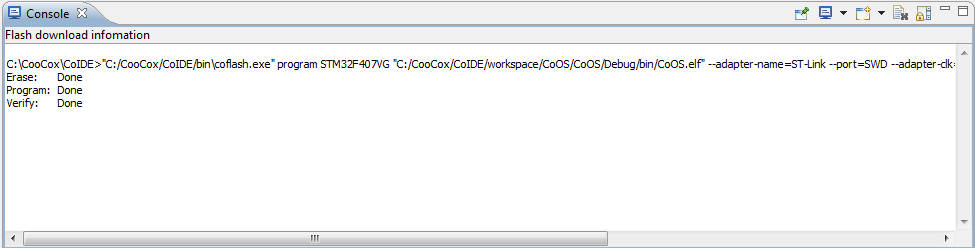
**Printf(“A\n”) -> không lỗi.**

### Load firmware lên board STM32F4

Sau khi code xong, bấm F7 để Build hoặc Ctrl + R để Rebuild lại toàn bộ mã nguồn. Sau đó bấm “Flash > Program Download ” trong cửa sổ CoIDE để nạp mã nguồn lên board.

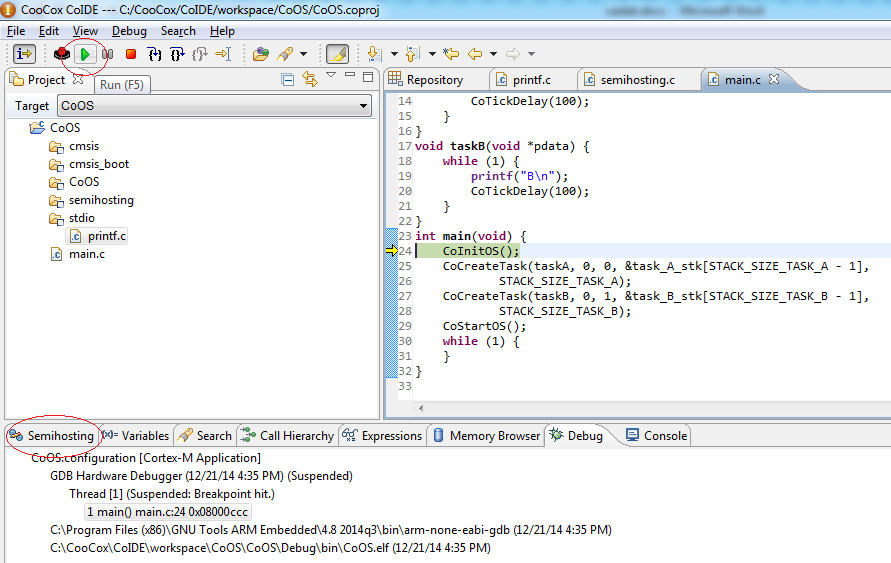
Lưu ý: Nếu file main.c không có nội dung,hoặc chỉ có vòng while(1) thì build sẽ ra file có kích thước bằng 0 và không thể download lên mạch được.

Kết quả sẽ được hiển thị ở khung Console. Đây là download thành công:



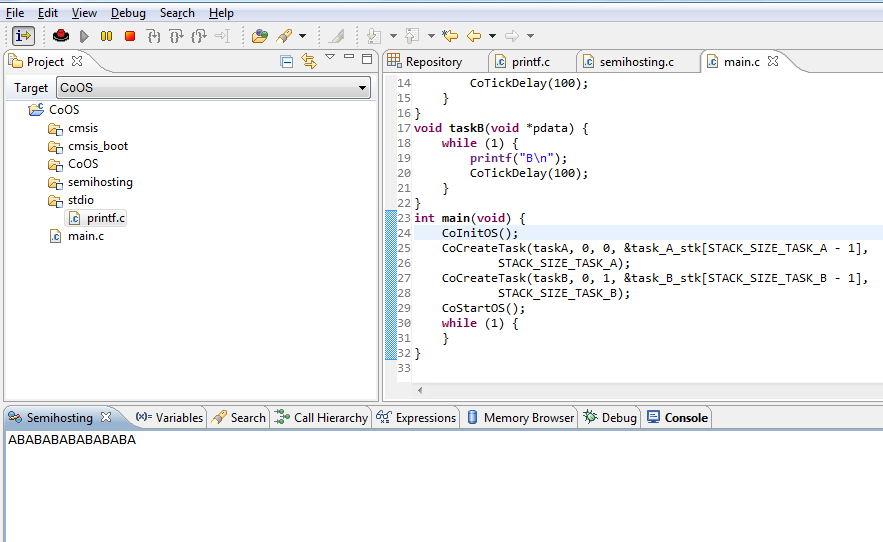
### Debug:

Bấm Ctrl + F5 để vào chế độ debug. Khi nhấn debug thì mã nguồn sẽ được build lại, nạp lên board và chuyển màn hình sang debug.



Con trỏ lệnh sẽ trỏ vào lệnh đầu tiên của hàm main. Bấm nút tam giác để chạy, hoặc Step into(F11), Step over(F10) để chạy từng lệnh một.

Bấm F5 để chạy và xem kết quả ở Semihosting.



## CHẠY CHƯƠNG TRÌNH DEMO TRÊN STM32F4

### Mô tả chương trình

Phát nhạc trên STM32F4

Phát âm thanh trên STM32F4:

Board STM32F4 được tích hợp sẵn chip xử lý âm thanh CS43L22. Đây là chip DAC tích hợp ampli class D, công suất ~1W với tải 8Ω, nguồn nuôi 5v ở chế độ stereo.

Chip CS43L22, để hiểu đơn giản, thì nó có 2 đường: 1 đường dữ liệu âm thanh digital , 1 đường điều khiển. Hai đường này được điều khiển qua 2 giao thức, đường âm thanh digital qua I2S, đường điều khiển qua I2C. Ứng với board STM32F4 thì đường I2S kết nối với chân SPI3, còn đường I2C kết nối với chân I2C1. Ngoài 2 chân này, ta cần phải cấu hình một vài chân GPIO khác (clock, reset, enable…).

Cấu hình Clock:

Trước hết phải bật clock lên. Để làm việc này thì cần bật clock cho các chân GPIO sau: GPIOA (I2S\_WS signal), GPIOB (I2C\_SDA & I2S\_SCL), GPIOC (I2S\_MCK, I2S\_SCK, I2S\_SD) và GPIOD (Reset CS43L22). Việc cấu hình và bật clock thông qua thanh ghi Reset and Clock Control” (RCC) (nằm trong thư viện **stm32f4xx\_rcc.h**). Sử dụng lệnh sau để làm các việc trên:

***RCC\_AHB1PeriphClockCmd****(RCC\_AHB1Periph\_GPIOA | RCC\_AHB1Periph\_GPIOB | RCC\_AHB1Periph\_GPIOC | RCC\_AHB1Periph\_GPIOD, ENABLE);*

Tiếp theo, ta cần bật clock cho chân SPI3 và I2C1 (chân data và chân điều khiển chip CS43L22).

***RCC\_AHB1PeriphClockCmd****(RCC\_AHB1Periph\_GPIOA | RCC\_AHB1Periph\_GPIOB | RCC\_AHB1Periph\_GPIOC | RCC\_AHB1Periph\_GPIOD, ENABLE);*

I2S có module PLL dùng để tạo tần số lấy mẫu chuẩn. Để bật, dùng lệnh sau:

**RCC\_PLLI2SCmd**(ENABLE);

Cấu hình GPIO:

Cần set chế độ làm việc của chân (in, out), cấu hình đầu ra (push-pull hay là open-drain, tốc độ…). Thư viện “stm32f4xx\_gpio.h” cung cấp 1 struct cho phép làm những việc này **GPIO\_InitTypeDef**. Ví dụ với tín hiệu reset thì struct sẽ như sau:

*GPIO\_InitTypeDef PinInitStruct;*

*PinInitStruct.GPIO\_Pin = GPIO\_Pin\_4;*

*PinInitStruct.GPIO\_Mode = GPIO\_Mode\_OUT;*

*PinInitStruct.GPIO\_OType = GPIO\_OType\_PP;*

*PinInitStruct.GPIO\_PuPd = GPIO\_PuPd\_DOWN;*

*PinInitStruct.GPIO\_Speed = GPIO\_Speed\_50MHz;*

***GPIO\_Init****(GPIOD, &PinInitStruct);*

Chế độ làm việc của I2S và I2C sẽ là “Alternate function” (GPIO\_Mode\_AF). Đầu ra của chân I2C sẽ là “open drain” (GPIO\_OType\_OD) và “no pull” (GPIO\_PuPd\_NOPULL). Đầu ra của chân I2S sẽ là “push-pull” (GPIO\_OType\_PP). Do hầu hết cá chân đều được ghép giữa nhiều “alternate functions”, nên ta cần phải gán nó cho 1 cái thích hợp. Việc này sẽ được thực hiện nhờ câu lệnh sau:

***GPIO\_PinAFConfig****(GPIOA, GPIO\_PinSource4, GPIO\_AF\_SPI3);*

Cấu hình I2S:

Bây giờ thì các chân GPIO đã được cấu hình, và ta có thể sử dụng được. Do I2S là 1 phần của SPI nên các hàm cần thiết sẽ nằm trong “stm32f4xx\_spi.h”

*I2S\_InitTypeDef I2S\_InitType;*

*I2S\_InitType.I2S\_AudioFreq = I2S\_AudioFreq\_48k;*

*I2S\_InitType.I2S\_MCLKOutput = I2S\_MCLKOutput\_Enable;*

*I2S\_InitType.I2S\_Mode = I2S\_Mode\_MasterTx;*

*I2S\_InitType.I2S\_DataFormat = I2S\_DataFormat\_16b;*

*I2S\_InitType.I2S\_Standard = I2S\_Standard\_Phillips;*

*I2S\_InitType.I2S\_CPOL = I2S\_CPOL\_Low;*

***I2S\_Init****(SPI3, &I2S\_InitType);*

Sau khi khởi tạo xong, ta có thể bật ngoại vi này lên:

***I2S\_Cmd****(SPI3, ENABLE);*

Cấu hình I2C và kết nối đến CS43L22:

Giống GPIO và I2S, chân I2C cũng cần phải được cấu hình để giao tiếp được với DAC. Các hàm cần thiết nằm trong thư viện “stm32f4xx\_i2c.h”

*I2C\_InitType.I2C\_ClockSpeed = 100000;*

*I2C\_InitType.I2C\_Mode = I2C\_Mode\_I2C;*

*I2C\_InitType.I2C\_OwnAddress1 = 99;*

*I2C\_InitType.I2C\_Ack = I2C\_Ack\_Enable;*

*I2C\_InitType.I2C\_AcknowledgedAddress = I2C\_AcknowledgedAddress\_7bit;*

*I2C\_InitType.I2C\_DutyCycle = I2C\_DutyCycle\_2;*

***I2C\_Init****(I2C1, &I2C\_InitType);*

***I2C\_Cmd****(I2C1, ENABLE);*

Đến đây là hoàn thành các bước cấu hình ở phía STM32F4.

Cấu hình DAC. Đầu tiên, ta cần bật nó lên bằng các đưa mức tín hiệu cao vào chân reset:

***GPIO\_SetBits****(GPIOD, GPIO\_Pin\_4);*

CS43L22 giống như 1 con chip vi điều khiển, nó có các thanh ghi , và ta sử dụng giao thức I2C để đọc, ghi vào các thanh ghi đó. Các hàm để đọc ghi như sau:

* *I2C\_GenerateSTART*
* *I2C\_Send7bitAddress*
* *I2C\_SendData*
* *I2C\_ReceiveData*
* *I2C\_GenerateStop*

Trước khi truyền tải, ta cần kiểm tra xem CS43L22 có sẵn sàng cho việc truyền tải không bằng 2 lệnh:

* I2C\_CheckEvent  sử dụng để kiểm tra các event nếu có
* I2C\_GetFlagStatus để kiểm tra trạng thái các cờ (ví dụ kiểm tra xem I2C bus có bận không trước khi truyền tải)

Bây giờ thì có thể bắt đầu công việc truyền tải. Sử dụng lệnh sau:

*SPI\_I2S\_SendData(SPI3, theData);*

theData là 1 giá trị 16 bit không dấu (unsigned integer). Khi gọi liên tục hàm này thì sẽ tự đổi từ kênh trái kênh phải. Trước khi truyền thì nên kiểm tra xem buffer truyền có trống không bằng lệnh sau (hoặc dùng ngắt):

*SPI\_I2S\_GetFlagStatus(SPI3, SPI\_I2S\_FLAG\_TXE);*

### Chạy Demo – Debug

### Đánh giá

# KẾT LUẬN