Bài 27. Giao tiếp STM32 với thẻ nhớ microSD và sử dụng tập tin với thư viện fatFS

Kiến thức cần chuẩn bị18

1. Chuẩn SDIO

SDIO dùng để giao tiếp giữa Bus ngoại vi APB2 và MultiMediaCards(MMC) , thẻ nhớ SD , SDIO và thiết bị CE-ATA.

Tham khảo thêm
https://www.st.com/content/ccc/resource/technical/document/reference_manual/9b/53/39/1c/f7/01/4a/79/DM00119316.pdf/files/DM00119316.pdf/jcr:content/translations/en.DM00119316.pdf

Truyền nhận dữ liệu từ thẻ nhớ SD/SDIO được thực hiện trong các khối dữ liệu

SDIO chỉ hộ trợ mode giao tiếp 1bit (mặc định) và 4bit . SDIO không tương thích với chuẩn SPI

Figure 229. SDIO "no response" and "no data" operations

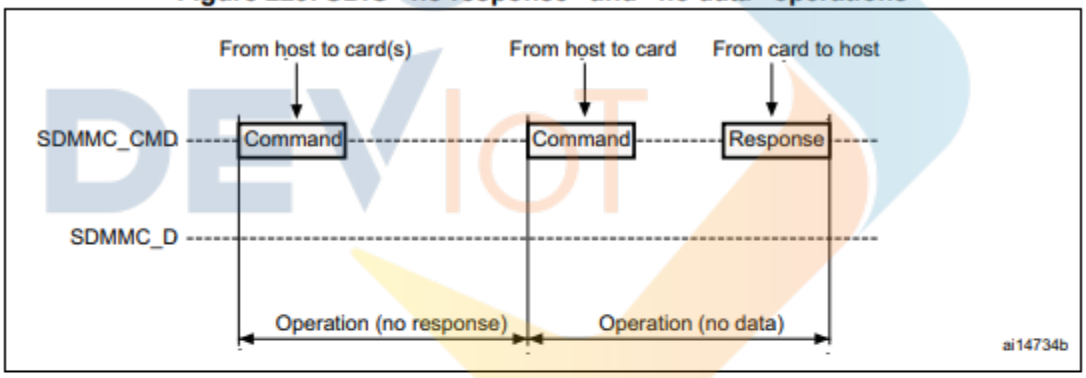


Figure 230. SDIO (multiple) block read operation

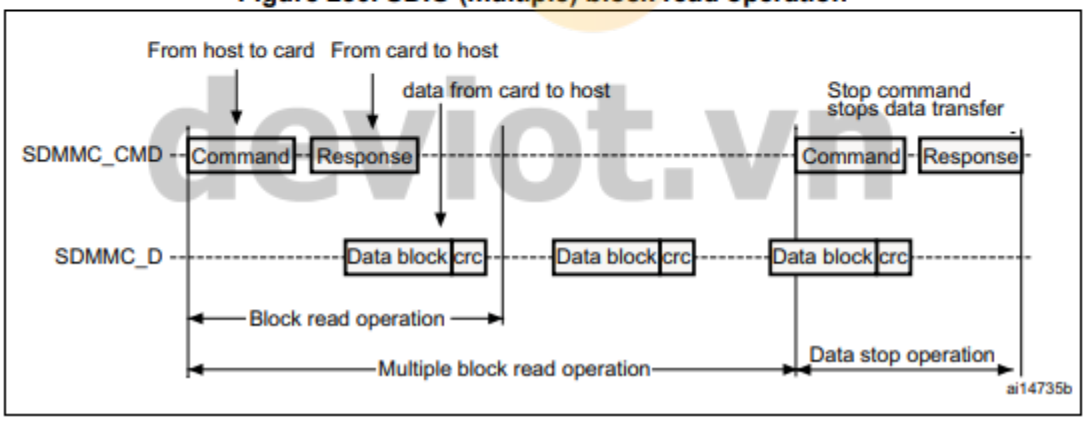
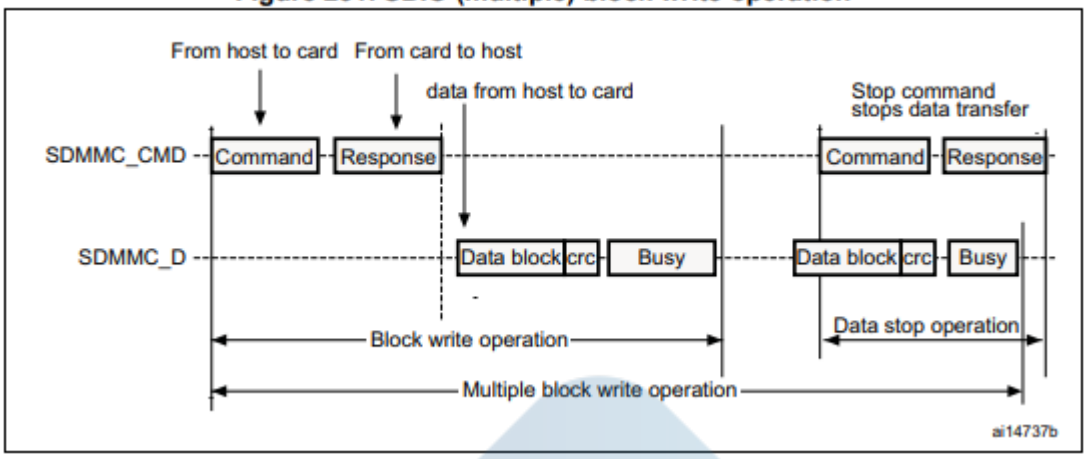


Figure 231. SDIO (multiple) block write operation



Khối SDIO đọc cùng lúc nhiều kênh

Figure 232. SDIO sequential read operation

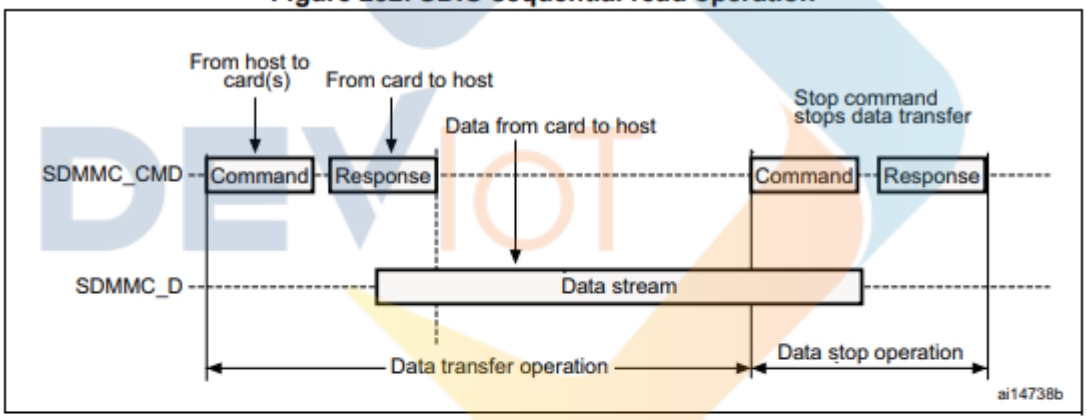
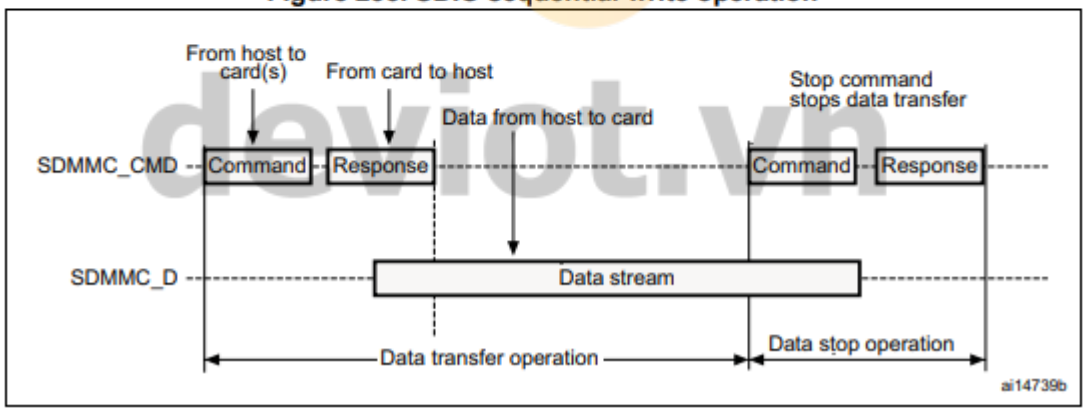
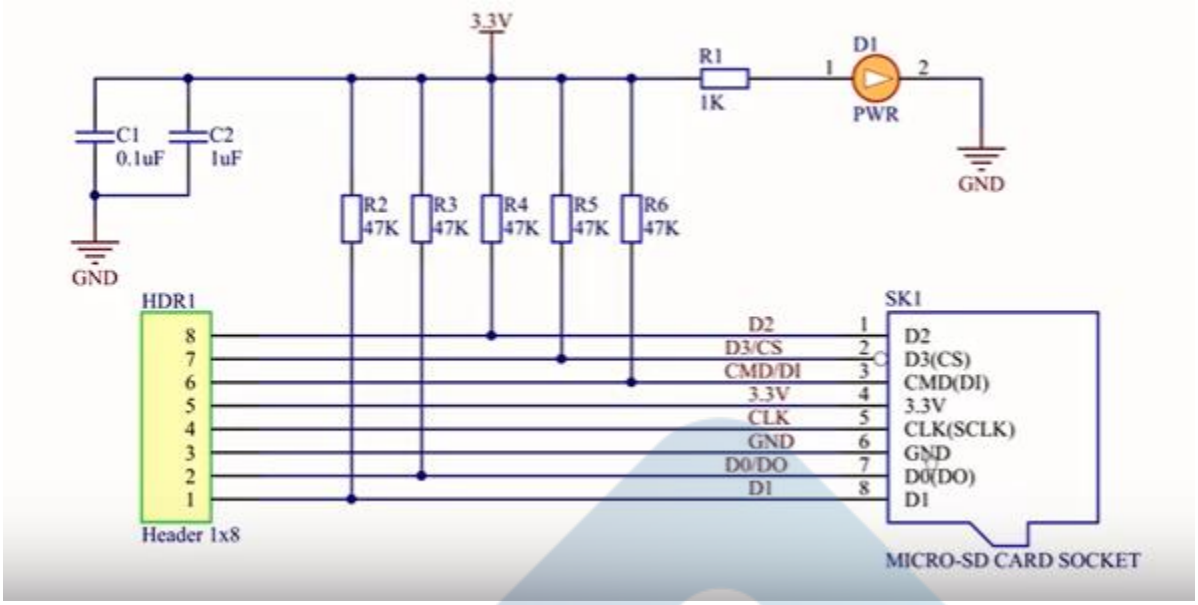


Figure 233. SDIO sequential write operation



Khối SDIO đọc lần lượt từng kênh

Ta cần chuẩn bị một Micro SD card



Các chân D0 D1 D2 D3 là các chân dữ liệu , tối đa là 4 bit

Với chế độ 1bit sẽ mặc định là chân D0 ,các chân dữ liệu có trở kéo lên

Chân CMD (Command) để gửi lệnh xuống

Chân GND và 3.3(V) là chân cấp nguồn , CLK là chân cấp xung clock

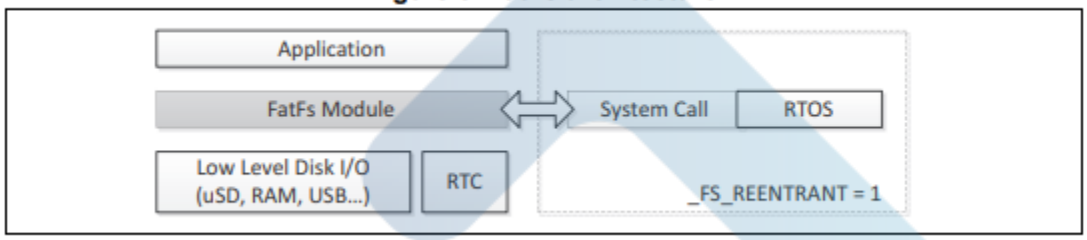
2. Thư viện fatFS

https://www.st.com/content/ccc/resource/technical/document/user_manual/61/79/2b/96/c8/b4/48/19/DM00105259.pdf/files/DM00105259.pdf/jcr:content/translations/en.DM00105259.pdf

ST hướng dẫn rất chi tiết về FAT và thư biện fatFS hỗ trợ định dạng và lưu trự trên ổ đĩa và thiết bị bộ nhớ

fatFS là một loại module hệ thống tệp FAT chung cho các hệ thống nhúng. fatFS được viết phù hợp với ANCI C và tách biệt với lớp I/O nên nó độc lập với phần cứng

Figure 3. FatFs architecture



fatFS là một middleware cung cấp chức năng để truy cập vào khối FAT trong bài ta sẽ dùng

f_mount(): Register/Unregister a work area

• f_open(): Open/Create a file

• f_close(): Close a file

• f_read(): Read a file

• f_write(): Write a file

• f_lseek(): Move read/write pointer, Expand a file size

• f_truncate(): Truncate a file size

• f_sync(): Flush cached data

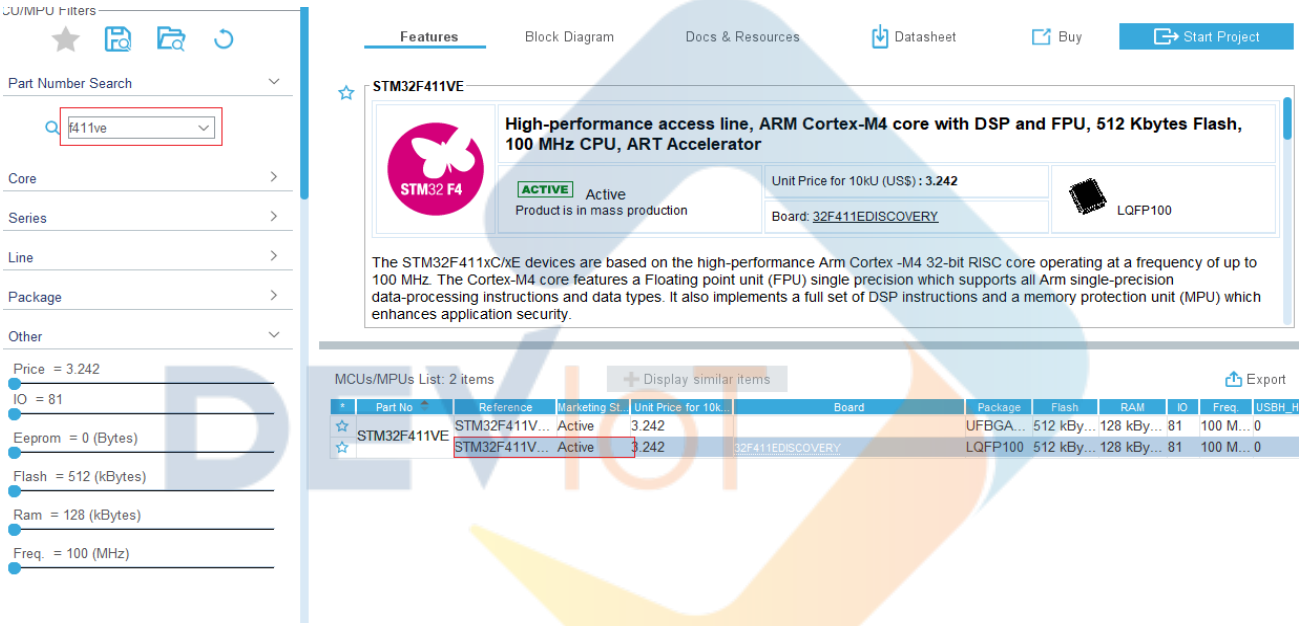
• f_opendir(): Open a directory

Các bạn tham khảm them tại link đầu bài viết

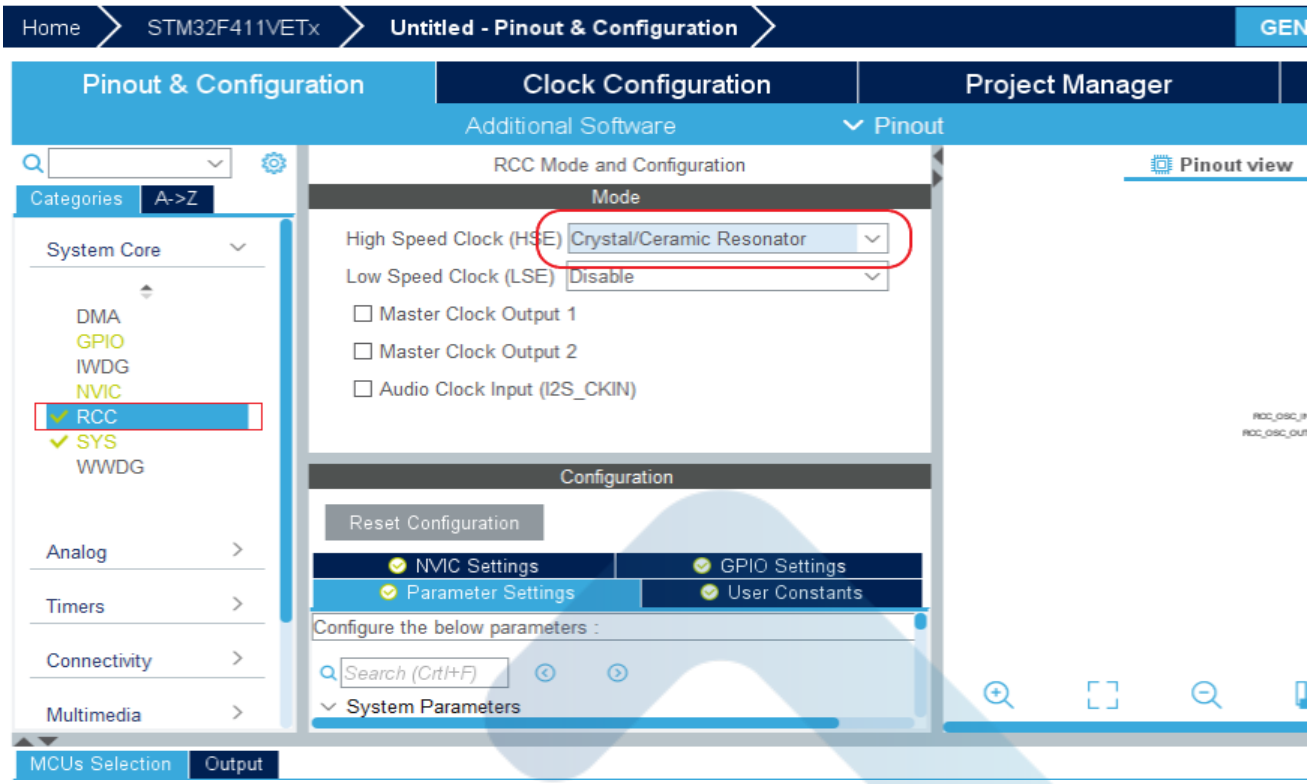
Lập trình

Mở phần mềm STM CubeMX, chọn dòng chip bạn sử dụng. Ở đây mình chọn chip STM32F411VE

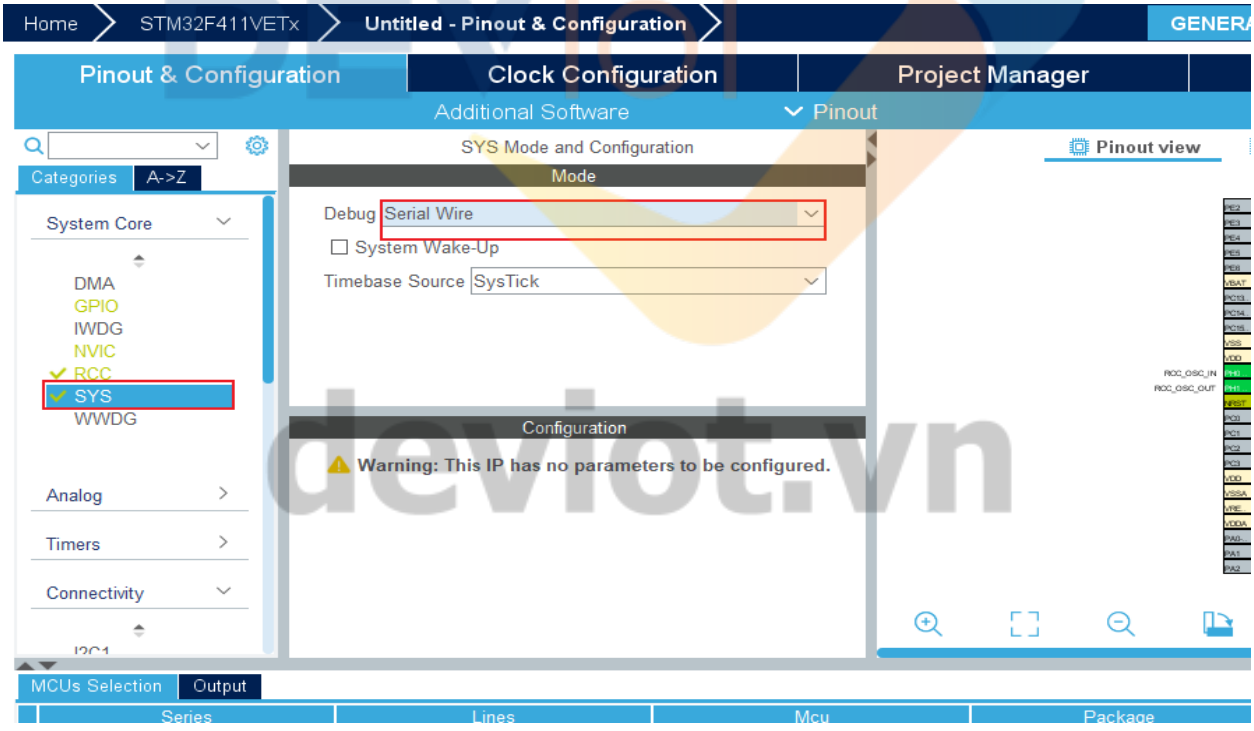
Đối với các dòng chip STM32 đời 4, tất cả mọi câu lệnh khi sử dụng thư viện HAL đều giống nhau. Chỉ khác nhau phần cấu hình Clock phụ thuộc riêng vào mỗi ch



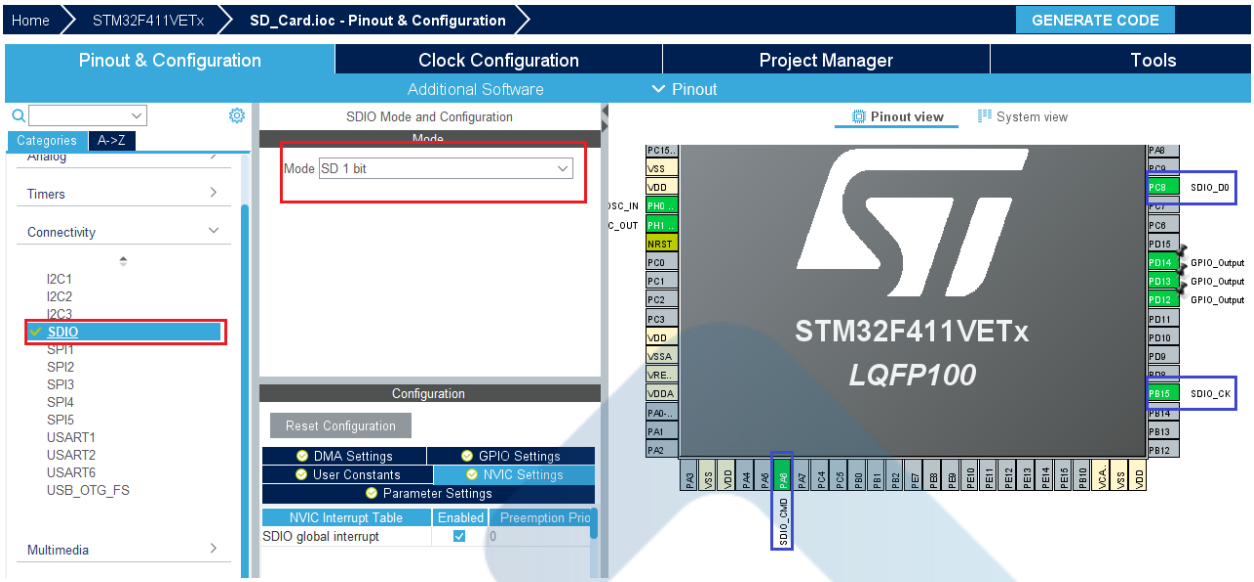
Sau đó cấu hình Chip sử dụng thạch anh ngoài hàn sẵn trên board mạch



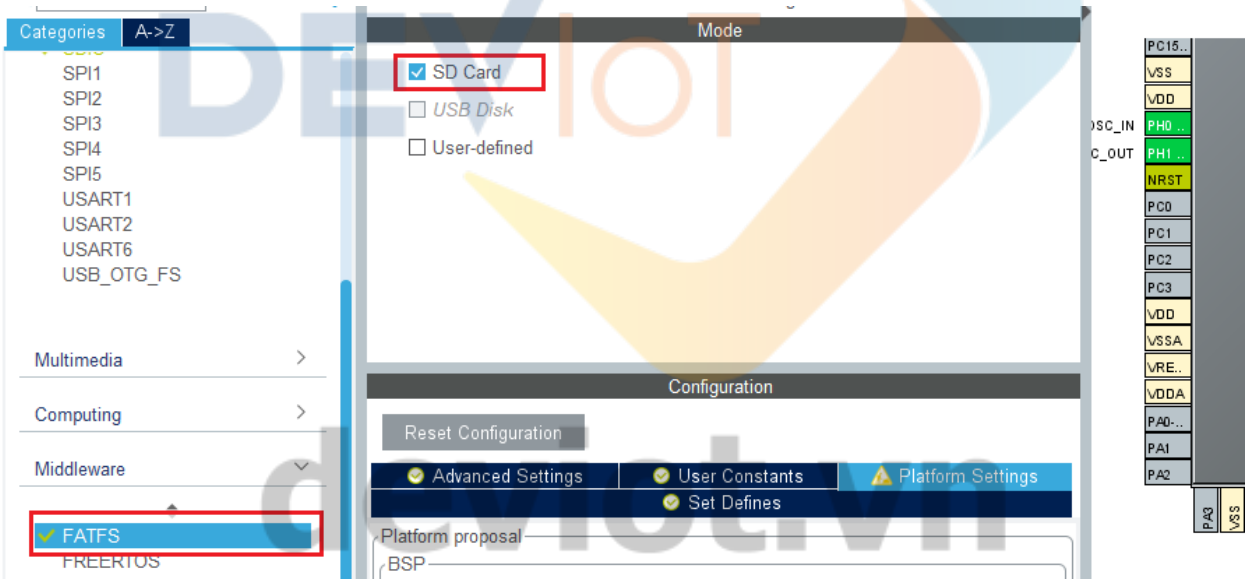
Chip Debug bằng SWD



Cấu hình SDIO với mode SD 1bit mặc định chân truyền dữ liệu là SDIO_D0 ba chân được cấu hình là PA6 PB15 PC8 . Nếu như dùng 4 bit sẽ mất them 3 chân cấu hình cho D1 D2 D3

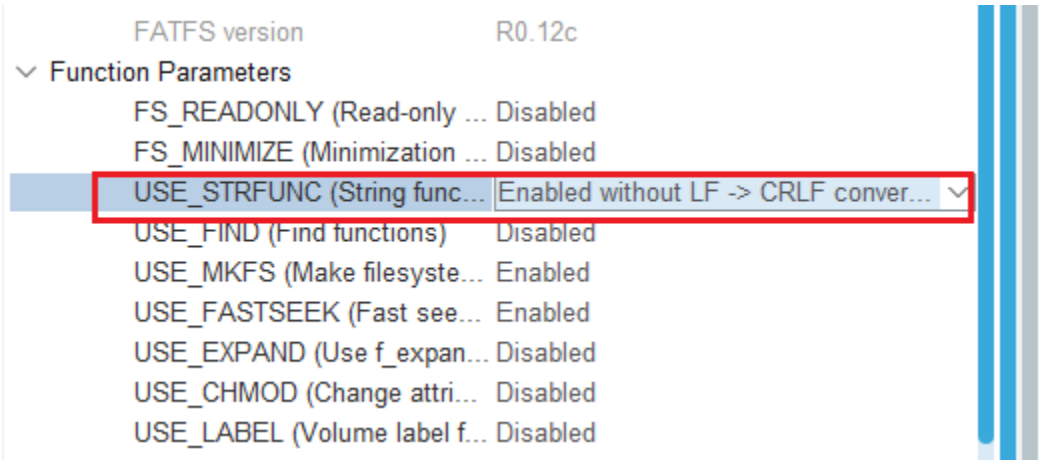


Sau đí tới mục Middleware \ FATFS chọn sử dụng SD Card

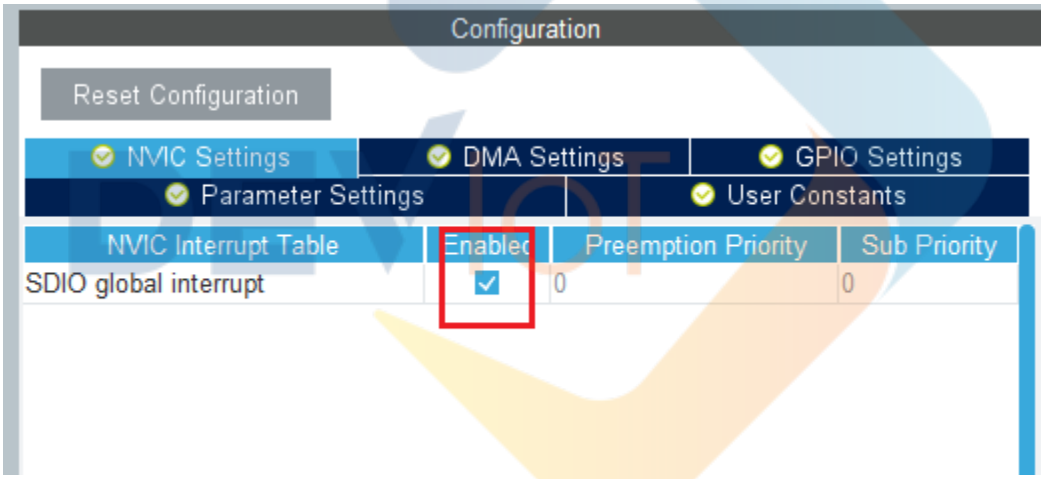


Tại Set Defines / Function Parameters / USE_STRFUC ta chọn

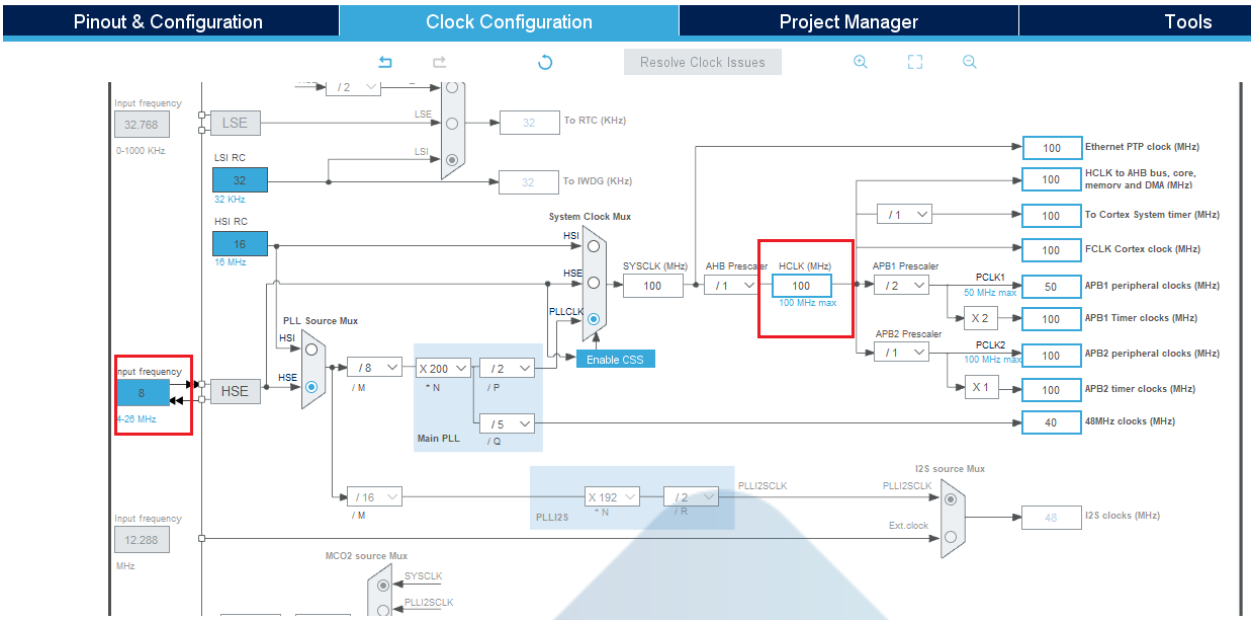
Enabled without LF -> CRLF conversion



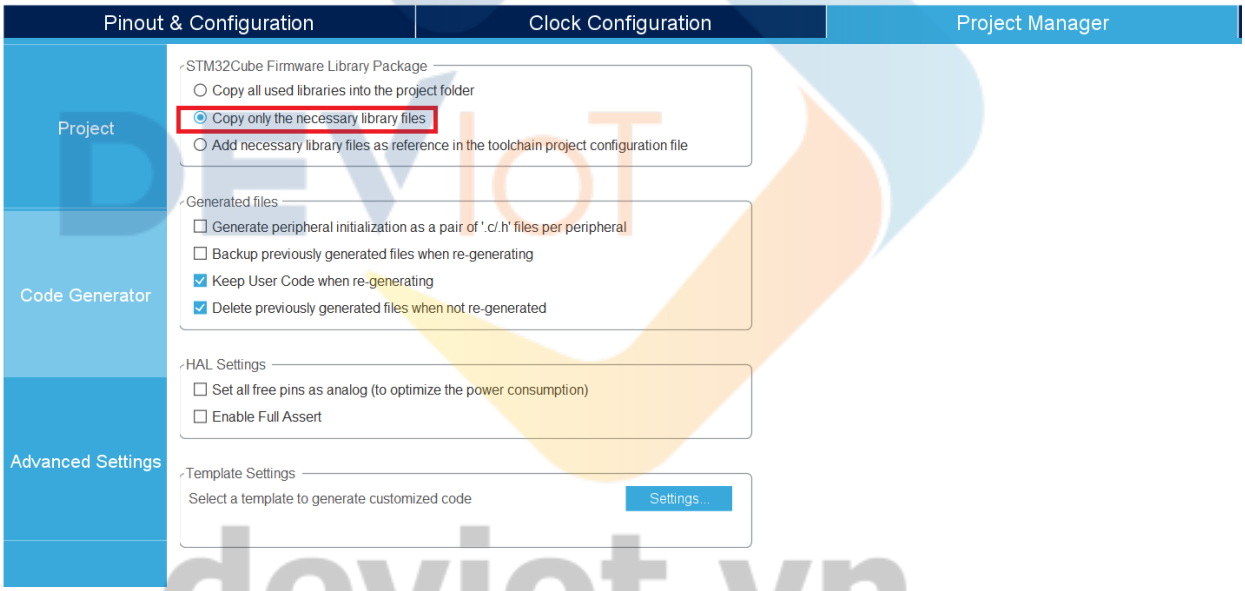
Cho phép ngắt



Sau đó cấu hình tần số hoạt động ta sử dụng thạch anh ngoài với Input frequency 8Mhz và xung lớn nhất Chip hỗ trợ là 100Mhz



Cuối cùng chọn file và sinh code cho Project



Chọn những thư viện cần thiết để sinh code nhanh hơn và giảm dung lượng Project nhé.

Sau đó ở Kiel ta sẽ thấy

```
FATFS fatfs;
FIL myfile;
FRESULT fresult;
int a = 10,byte_read=0;
float b = 0.211351;

uint8_t buffer[50], receiver_ar[100];
```

Tạo các biến cần thiết để truyền cho SD . FRESULT là giá trị mình kiểm tra xem việc giao tiếp với file có lỗi không

Trong int main() tại main.c ta them code

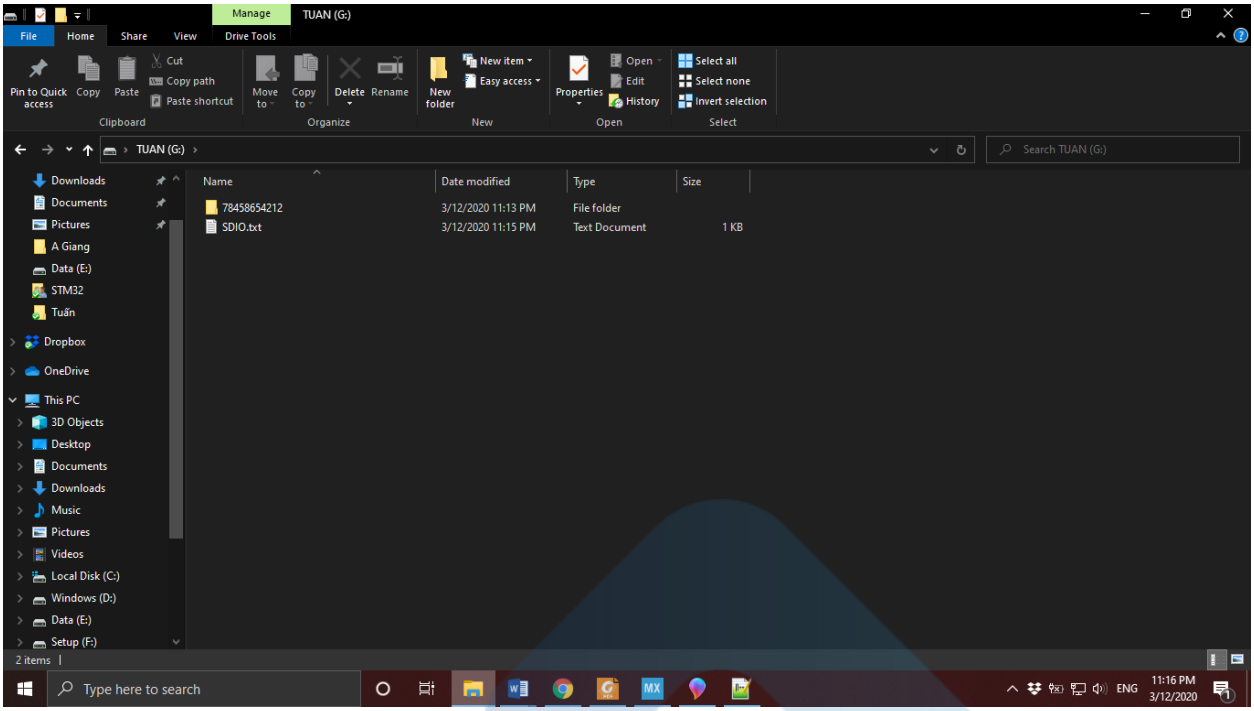
```
if(BSP_SD_Init()==MSD_OK)
        {
              fresult = f_mount(&fatfs,"",1);
              fresult =
f_open(&myfile,"SDIO.txt",FA_CREATE_ALWAYS|FA_WRITE);
              f_printf(&myfile,"Tuan_Doan\n2020");
              f_printf(&myfile,"%d",a);
              sprintf((char*)buffer,"%0.6f",b);
              f_printf(&myfile,(const char*)buffer);
                 f_close(&myfile);


//            fresult=f_open(&myfile,"SDIO.txt",FA_READ);
//            fresult=
f_read(&myfile,receiver_ar,f_size(&myfile),(UINT*)&byte_read);
//            f_close(&myfile);
        }
```

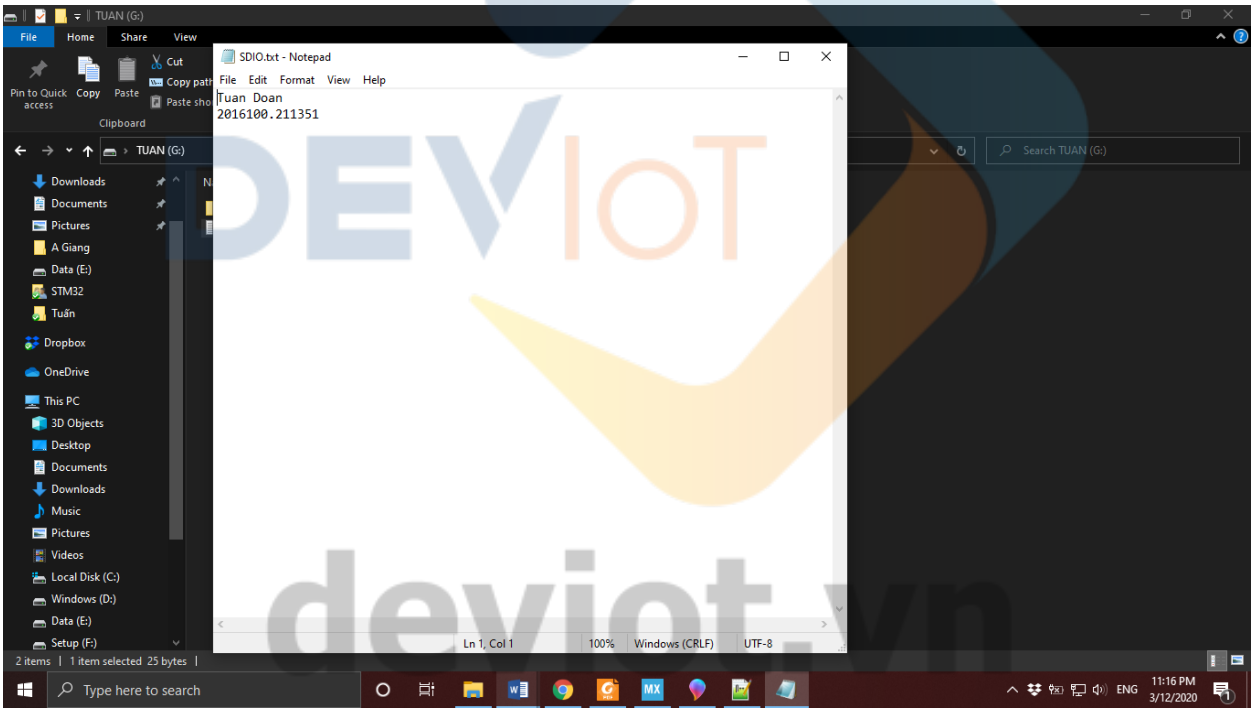
BSP_SD_Init để khởi tạo thẻ nhớ , f_mount khởi tạo 1 đường dẫn cho file

f_open để mở 1 file có sẵn hoặc tạo mới 1 file , hàm f_printf không hỗ trợ ghi số thực “%f” nên ta phải ép kiểu và truyền qua mảng buffer để hiển thị .

Các marco `FA_CREATE_ALWAYS|FA_WRITE` Nghĩa là (File_Access_xxx) quyền truy nhập file

Sau khi ghi phải đóng file thì tất cả các dữ liệu mới được ghi lại





Sau đó ta sẽ tiến hành đọc dữ liệu từ SD

ở hàm f_open ta thay File Access từ `FA_CREATE_ALWAYS|FA_WRITE -> FA_READ`

DEVIOT - CÙNG NHAU HỌC LẬP TRÌNH IOT

📌 Website: deviot.vn

📌 FanPage: Deviot - Thời sự kỹ thuật & IoT

📌 Group: Deviot - Cùng nhau học lập trình IOT

📌 Hotline: 0969.666.522

📌 Address: Số 101C, Xã Đàn 2

📌 Đào tạo thật, học thật, làm thật