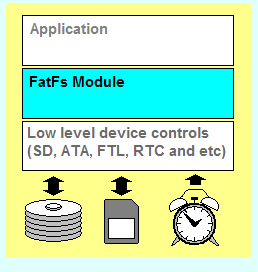
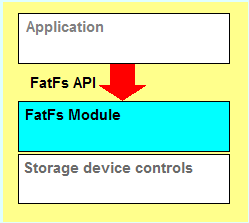
Giới thiệu về thư viện fatfs.

1. FATfs là gì?

FatFs là 1 module hệ thống tệp dữ liệu dành cho các hệ thống nhỏ. Do fatfs là đọc lập với các nền tảng vi xử lí lí nên nó có thể tích hợp vào các bộ vi điều khiển nhỏ để quản lí dữ liệu dưới dạng tệp.

Về bản chất , đây chỉ là 1 tầng trung gian để ta có thể truy cập và quản lí dữ liệu dạng file từ tầng ứng dụng xuống để đọc ghi dữ liệu xuống các lớp thấp hơn như thẻ SD hay bộ nhớ ngoài flash. Sơ đồ sau thể hiện vai trò của FatFs trong hệ thống dữ liệu của hệ thống nhúng:

2. FatFs và stm32

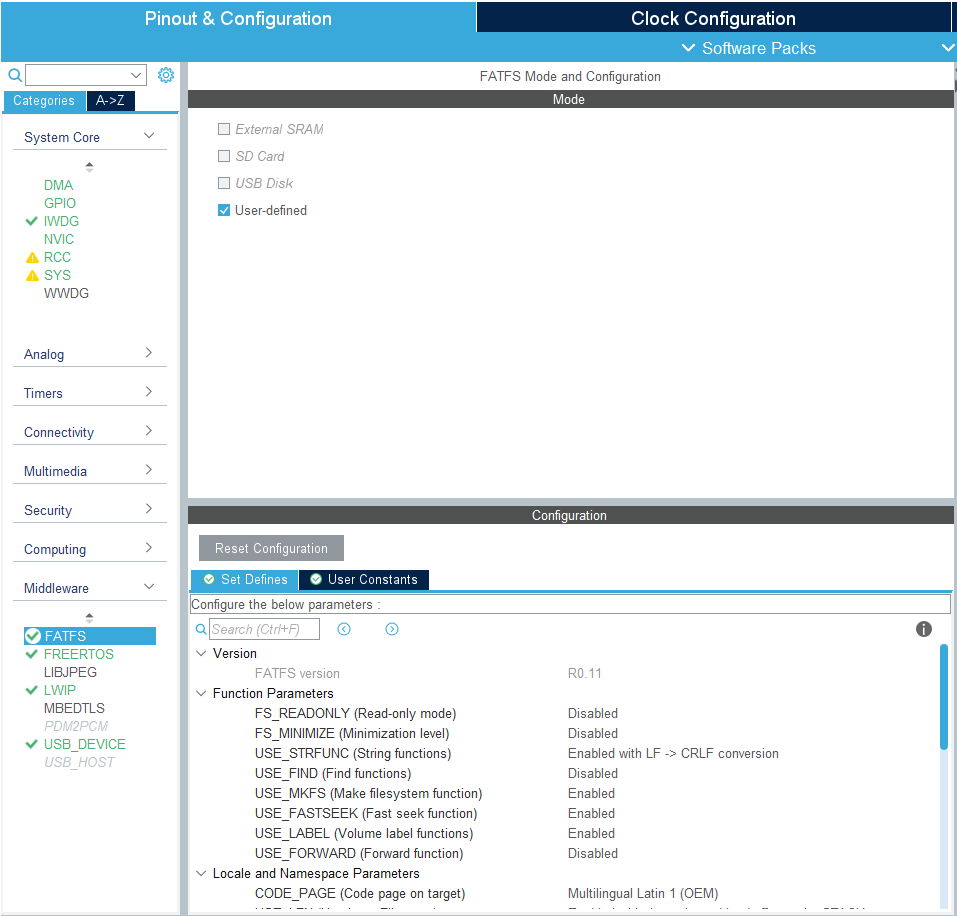
Trong cubeMX cung cấp 4 tùy chọn cài đặt cho fatfs, trong đó có 3 tùy chọn có sẵn dành cho thẻ SD, SRAM ngoài và USB MSC và thêm 1 tùy chọn do người dùng định nghĩa.

Ở đây chúng ta sẽ tập chung vào phần người dùng định nghĩa vì 3 tùy chọn còn lại đã có thư viện có sẵn hỗ trợ. Trong phần Middleware chọn tùy chọn user-define. Chọn tùy chọn

USE LONG FILENAME.

\_MIN\_SS 512

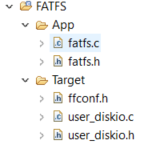
\_MAX\_SS 4096



3. Các file cần lưu ý và ý nghĩa.

Sau khi gen code chúng ta cần lưu ý đến các thư mục sau:

3.1. FATFS



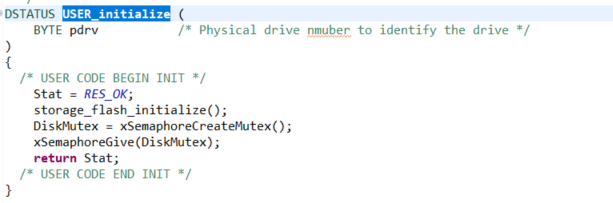
Trong thư mục này chứa 2 thư mục con là app và taget.

Trong đó, app chứa file fatfs.c và fatfs.h là nơi chúng ta viết các hàm ứng dụng cho ổ đĩa ví dụ như: hàm đọc file, tạo file, copy file….Trong file này cũng chứa 1 hàm cần lưu ý là void MX\_FATFS\_Init(void). Đây là hàm liên kết driver với link của của ổ đĩa. Trong trường hợp có 2 ổ đĩa (2 bộ nhớ cùng dùng thư viện fatfs để truy cập file) thì ổ đĩa sẽ được đánh từ 0:/ rồi đến 1:/ ứng với tùy chọn

#define \_STR\_VOLUME\_ID 0 /\* 0:Use only 0-9 for drive ID, 1:Use strings for drive ID \*/

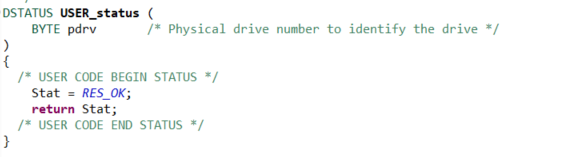
Thư mục target chứa file ffconf.h định nghĩa các tính năng của fatfs và các file định nghĩa tầng dưới truy nhập vào bộ nhớ cần quản lý dữ liệu. Nếu các bạn chọn tùy chọn ví dụ như thẻ SD cube ide sẽ cung cấp luôn thư viện SDcardio.c(& .h) (cái này ko nhớ rõ lắm). Nhưng đối với phần user\_diskio thì ta cần định nghĩa lại 1 số hàm như sau:

**USER\_initialize**

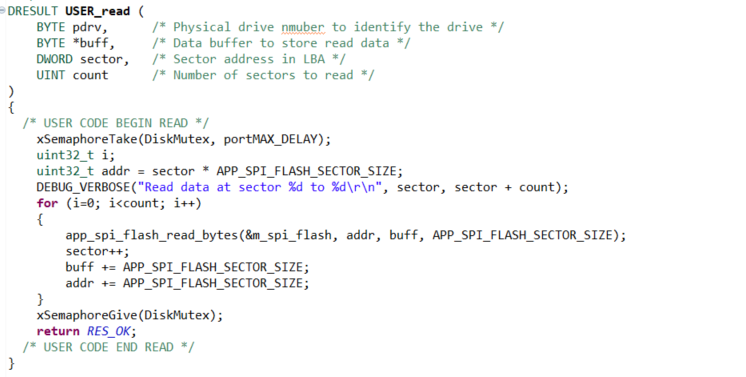


Là hàm cài đặt định nghĩa về thông tin ổ đĩa, các tác vụ khác liên quan.

**USER\_status**

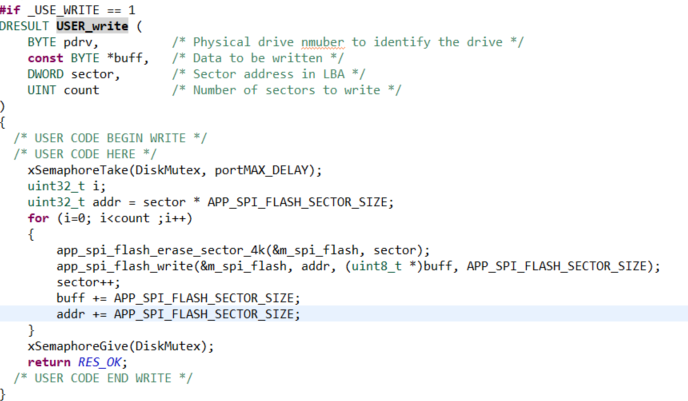
  
Hàm trả về trạng thái ổ đĩa. Thường thì hàm này chỉ cần trả về trạng thái OK.

**USER\_read**



Hàm đọc ổ đĩa. Hàm được truyền vào dữ liệu ở con trỏ buff, địa chỉ của sector và số sector cần đọc. Việc Sector có độ lớn bao nhiêu phụ thuộc vào ứng dụng. Ở ví dụ trong hình mình dùng flash gd25q64 với độ lớn sector = 4096.

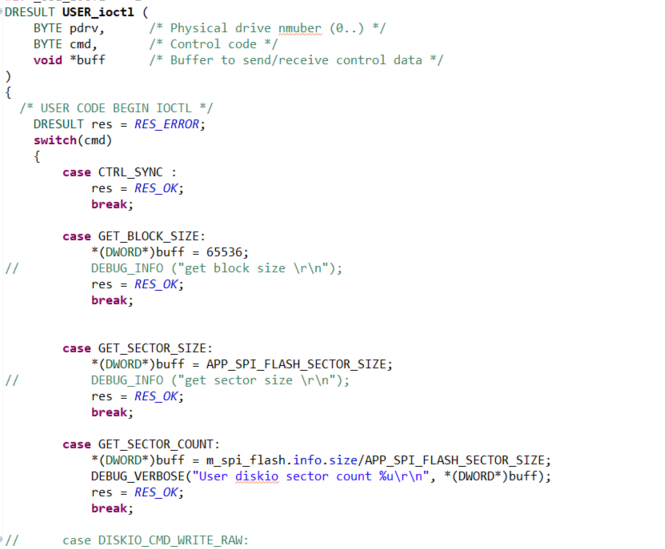
**USER\_write**



Hàm viết vào bộ nhớ với các tham số như hàm đọc.

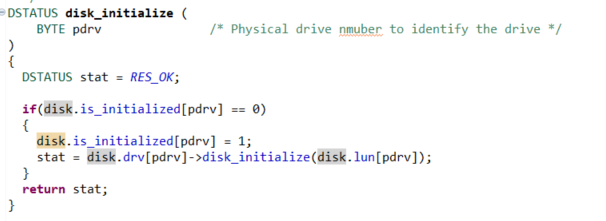
**USER\_ioctl**

Hàm điều khiển dữ liệu vào ra bạn có thể viết như sau. Các giá trị Block, sector size tùy chỉnh theo ứng dụng.



3.2. Middleware->Third\_Party->FatFs->src

Trong thư mục này chứa file điều hướng ổ đĩa. Thường ta ko cần quan tâm đến thư mục này, tuy nhiên nếu bạn dung từ 2 ổ đĩa trở lên cần kiểm tra tham số pdrv có ứng với ổ đĩa thiết lập ban đầu ko?



Như trên đây nếu địa chỉ vật lý ổ đĩa với 0 thì ứng với lun[0]. Thường giá trị này trùng với path của ổ đĩa.

4. Các lưu ý khi dung thư viện fatfs

4.1. các lỗi có thể mắc phải

Thư viện fatfs cung cấp lượng lớn các hàm và các hàm này đều có tham số mode. Nếu bạn chọn sai mode thì sẽ gây ra lỗi.

VD: Nếu bạn cố gắng ghi vào 1 file được mở với chế độ READ\_ONLY thì sẽ gây ra lỗi.

Khi dùng 2 ổ đĩa trở lên cần ghi rõ FilePath để tránh gây ra lỗi. Khi không ghi rõ ổ đĩa, fatfs sẽ mặc định coi đó là ổ 0:/

VD: f\_open &USERFile, file, FA\_READ); -> mở file trong ổ đĩa 0:/file

f\_open &USERFile, 1:/file, FA\_READ);-> mở thư mục tại ổ đĩa 1.

Mở thư mục 2 lần (f\_open nhưng không f\_close).

Đối với lỗi này sẽ gây ra hỏng liên kết vì vậy cần format lại ổ đĩa nếu xảy ra trường hợp trên. Khi có lỗi này sẽ trả về FR\_NO\_FILESYSTEM.

Quên ko cấu hình cho disk\_status