HƯỚNG DẪN VIẾT THƯ VIỆN C CHO ỨNG DỤNG NHÚNG

Thuong Nguyen

TỔNG QUAN - MỤC ĐÍCH THƯ VIỆN

TAPiT

Lợi ích khi viết thành file thư viện:

- Project lớn, phức tạp => dễ quản lý
- Sử dụng nhiều lần
- Dễ kế thừa và chia sẻ
- Compile 1 lần phần thư viện lần đầu hoặc chỉ khi thay đổi, tối ưu tgian.

TỔNG QUAN - CÁC THÀNH PHẦN LIÊN QUAN

- File **header** (*.h) chứa các **khai báo** (declaration)
- File **source** (*.c) chứa các **định nghĩa hàm** (definition)
- File **main.c** sử dụng các khai báo, các hàm (thực hiện lời gọi hàm)

FILE HEADER

Có phần mở rộng (phần đuôi) .h

Bao gồm:

- **Khai báo** nguyên mẫu hàm // function prototype declare, thư viện, định nghĩa, macro, hằng và biến để các file source, project sử dụng.

Có 2 loại file header:

- File do người lập trình viết. // do mình viết
- File của hệ thống đi kèm compiler. // đi kèm với phần mềm
- #include <stdio.h>
- include <math.h>
- include “ds1307.h”

File Source

Có phần mở rộng (phần đuôi) .c

Bao gồm:

- Khai báo thư viện (include file header), định nghĩa , marco, biến, hằng (phục vụ cục bộ)
- **Định nghĩa hàm**

Có 2 loại file Source:

- File do người lập trình viết.
- File của hệ thống đi kèm compiler. string.c / stdio.c / math.c

CÁCH HOẠT ĐỘNG CỦA THƯ VIỆN

Khai báo Header file tại Source file bằng tiền chỉ thị #include

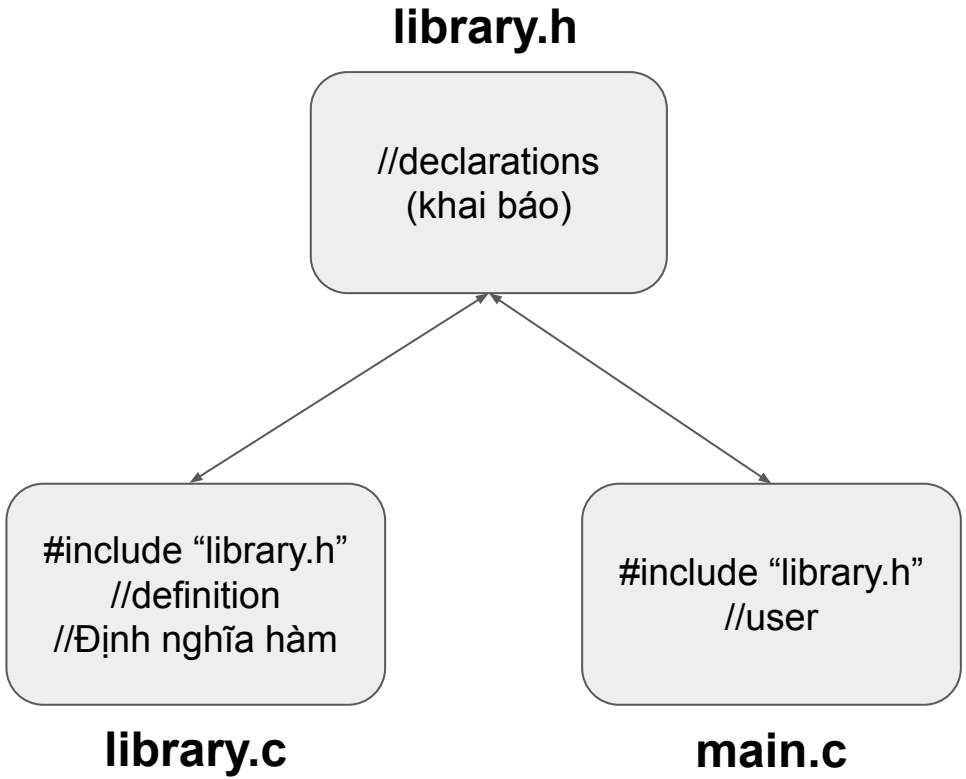
=> Bản chất của việc này là?

- Tại quá trình tiền xử lý
- Dữ liệu toàn bộ của header file được sao chép vào vị trí #include thư viện tại source file

  => Các hàm, marco, biến, hằng, kiểu định nghĩa đã được khai báo.

https://tapit.vn/qua-trinh-bien-dich-mot-chuong-trinh-cc/

CÁCH HOẠT ĐỘNG CỦA THƯ VIỆN



- Header file (library.h) xác định liên kết giữa code người dùng và code định nghĩa (thường trong thư viện)
- Các khai báo #include giống nhau trong cả hai Source file (định nghĩa và sử dụng) dễ dàng kiểm tra tính nhất quán.

CÁC BƯỚC VIẾT THƯ VIỆN: TẠO FILE

- Tạo file .h và file.c // lưu ý tên file giống nhau và không viết hoa
- Tạo sườn (cấu trúc) cho file

ds1307.h
ds1307.c

Cấu trúc file header .h

- Chống trùng lặp // One Definition
- Include thư viện khác cần dùng
- Khai báo struct -- khai báo kiểu dữ liệu
- Khai báo enum (có thể trước struct)
- Define macro giá trị // Hằng số
- Define function macro - kiểu hàm
- Khai báo nguyên mẫu hàm

Cấu trúc file source .c

- Include thư viện khác cần dùng
- Khai báo struct -- khai báo kiểu dữ liệu phục vụ nội bộ trong file
- Khai báo enum (có thể trước struct) phục vụ nội bộ trong file
- Define macro giá trị // Hằng số phục vụ nội bộ trong file
- Define function macro - kiểu hàm phục vụ nội bộ trong file
- **Định nghĩa hàm**

Chống trùng lặp

Thư viện abc.h cần sử dụng hàm được định nghĩa trong thư viện string.h nên thư viện abc.h cần include string.h

Người dùng họ k quan tâm đến việc bên trong thư viện sử dụng những hàm nào của thư viện nào khác. mà người dùng chỉ quan tâm đến những hàm được khai báo, định nghĩa mà người dùng cần dùng.

Người dùng phát triển chương trình ở file main.c mà chương trình cần những hàm của thư viện string.h nên main.c cần include string.h

Vậy chuyện gì xẩy ra nếu main.c include abc.h và include string.h

Function macro (smart macro)

- Sự thân thiện, dễ đọc
- Hiệu năng
- Tài nguyên bộ nhớ

#define __HAL_GPIO_EXTI_GET_FLAG(__EXTI_LINE__) (EXTI->PR & (__EXTI_LINE__))

(EXTI->PR & (GPIO_PIN_0) Phép toán

uint8_t HAL_GPIO_EXTI_GetFlag (uint8_t GPIO_PIN) Thực thi hàm Vào hàm, stack hàm, tính toán, trả dữ liệu, thoát hàm

```
{
  return (EXTI->PR & (GPIO_PIN))
}
```

Chủ yếu dùng fm khi làm việc với các thanh ghi, và thực hiện 1 phép toán nào đó đơn giản

CÁC BƯỚC VIẾT THƯ VIỆN - XÁC ĐỊNH HÀM

Xác định các chức năng - các function,

- Khởi tạo và xoá khởi tạo: ABC_Init(), ABC_DeInit();

- Nhập xuất, giao tiếp: ABC_Read(), ABC_Write(),ABC_Transmit(), ABC_Receive()

- Điều khiển, giám sát: ABC_Set(), _Clear(), ABC_Get().

- Trạng thái và báo lỗi: ABC_GetState(), ABC_GetError()

⇔ Giống các thanh ghi Cấu hình / dữ liệu / trạng thái

Tên hàm: động từ, thực hiện 1 cái gì đó cụ thể, độc lập (không dùng biến toàn cục).

CÁC BƯỚC VIẾT THƯ VIỆN - GIAO DIỆN HÀM

Tham số truyền vào

- Bao nhiêu tham số
- Kiểu dữ liệu của các tham số

Dữ liệu trả về

- Kiểu dữ liệu trả về
- Trả về thông qua return hay thông qua tham số (địa chỉ)
- Trả về các trạng thái lỗi (thông qua return)

CÁC TÀI NGUYÊN DÙNG CHUNG

- Hằng
- Biến

PHẠM VI SỬ DỤNG

- Hàm/File/Project
- static, extern

Cấu trúc main.c

- Thêm thư viện
- Define macro
- Biến toàn cục
- Khai báo nguyên mẫu hàm cho những hàm không có trong thư viện
- Hàm int main()
- Định nghĩa hàm

CÁC LƯU Ý KHI VIẾT THƯ VIỆN

- Phần header đầu thư viện //One Definition
- Document Function
  - Chú thích hàm
  - Mô tả chức năng lệnh trong hàm nếu phức tạp
- Phân chia bố cục các hàm theo các chức năng chính

Quy tắc 1: Mỗi thư viện (file .h và file .c) nên phục vụ cho một nhóm chức năng rõ ràng

VD:

- Thư viện string.h được xây dựng phục vụ riêng cho xử lý chuỗi
- Thư viện math.h được xây dựng riêng cho các hàm tính toán
- ....

Quy tắc 2: Luôn sử dụng “include guards” trong file header / Quy tắc One Definition

VD:

#ifndef GEOMETRY_BASE_H

#define GEOMETRY_BASE_H

and end with:

#endif

Lưu ý: Không sử dụng bắt đầu với dấu gạch dưới
:~~__GEOMETRY_BASE_H~~

Tên thư viện viết thường toàn bộ, define OneDefinition Viết hoa toàn bộ

TAPiT

Quy tắc 3: Tất cả các khai báo cần có để sử dụng trong thư viện cần được khai báo ở file header.

- Cung cấp đầy đủ thông tin cần thiết cho trình biên dịch.
- Tránh khai báo lặp khi header include 1 thư viện cần dùng khác (đã được đảm bảo nhờ Quy tắc One Definition.)

Quy tắc 4: File header chỉ bao gồm các khai báo và nó được include bởi file source (file .c include file.h)

- Chỉ đặt các khai báo kiểu cấu trúc, nguyên mẫu hàm và khai báo biến toàn cục extern trong file header;
- Đưa các định nghĩa hàm và các định nghĩa và khởi tạo biến toàn cục vào file source. File source của một thư viện phải include file header
- Trình biên dịch có thể phát hiện sự khác biệt giữa hai file header và source và từ đó giúp đảm bảo tính nhất quán.

Quy tắc 5: Khai báo biến toàn cục với từ khoá extern trong file header, và định nghĩa, khởi tạo giá trị trong file source. (ngoài phạm vi của file)

- Đối với biến toàn cục, khai báo với từ khoá **extern** trong file source sử dụng (file main.c)

  VD: **extern** *int* g_number_of_entities;

- File source sử dụng (main.c) sẽ include file header.
- File header (vì bên ngoài thư viện sẽ sử dụng) sẽ định nghĩa và khởi tạo giá trị cho biến toàn cục của thư viện

  VD: *int* g_number_of_entities = 0;

Lưu ý: Nên khởi tạo giá trị kể cả những biến có giá trị ban đầu bằng 0

Quy tắc 6: Các khai báo cục bộ nên nằm ở file source của thư viện

- Trong trường hợp thư viện cần sử dụng một số thành phần cục bộ mà **không có sự truy cập từ bên ngoài**.
- Ví dụ các bạn cần khai báo struct, biến toàn cục, hàm mà chỉ sử dụng chúng trong file soucre thì hay khai báo hoặc/và định nghĩa chúng ở file source, **không thực hiện việc này ở file header**.
- Khai báo kèm từ khoá **static** (==> internal linkage)

Quy tắc 7: Include đúng những file header khác mà thư viện cần sử dụng vào file header cần để biên dịch, không thêm thừa.

Header nào là cần thiết phải thêm vào trong file header? - Cần cho quá trình biên dịch

- VD: Nếu một struct có kiểu X được sử dụng làm struct member của struct kiểu A, thì bạn phải #include X.h trong A.h để trình biên dịch có thể biên dịch đúng

Không bao gồm các file header khác vào file header của thư viện mà chỉ file source mới cần.

- VD: Nếu cần sử dụng các định nghĩa hàm của một thư viện khác thì chỉ cần #include header của thư viện khác vào file source.

Quy tắc 8: Nội dung của file header cần được biên dịch kiểm tra

VD: Tạo ra 1 file test.c chỉ include file header cần kiểm tra và biên dịch.

Quy tắc 9: Cần include file header đi kèm trước các header khác.

Luôn #include file header của chính mình trước khi include header của các thư viện khác để tránh ẩn đi các thiếu sót của thư viện mình mà bị che đi bởi các thư viện khác.

Include thư viện khác ở file header hay file soucre?

• Nếu tệp X.h là một yêu cầu không thể tránh khỏi về mặt logic đối với các khai báo trong A.h để biên dịch, thì #including nó trong A.c là thừa, vì X.h đã được include vào A.h. Vì vậy, bạn có thể không #include X.h trong A.c.

• (nên) Khai báo #including X.h trong A.c là một cách để người đọc hiểu rõ rằng chúng tôi đang sử dụng X và giúp đảm bảo rằng các khai báo của X có sẵn ngay cả khi nội dung của A.h thay đổi do thiết kế thay đổi.

gpio.c

#include “gpio.h”

Quy tắc 10: Không bao giờ include một file source

- Gây nhầm lẫn cho người sử dụng
- Không thuận tiện khi sử dụng IDE, file source cần biên dịch độc lập nếu không có file header. => Dễ sai sót.

=> Cần xem xét chỉnh sửa code (trong main.c) hay phải tạo ra 1 thư viện.

Tài liệu tham khảo

1. C Header File Guidelines
   http://www.umich.edu/~eecs381/handouts/CHeaderFileGuidelines.pdf
2. Variable declaration
   http://www.it.uc3m.es/pbasanta/asng/course_notes/variables_en.html
3. Reserved Names
   https://www.gnu.org/software/libc/manual/html_node/Reserved-Names.html
4. Chapter 8— Libraries, C Programming for Embedded Systems
   http://dsp-book.narod.ru/CPES.pdf
5. C - Header Files
   https://www.tutorialspoint.com/cprogramming/c_header_files.htm
6. Functions and Header/Source files in C++
   https://cs.brynmawr.edu/Courses/cs246/spring2013/slides/04FunctionsAndHeaderFiles.pdf
7. C: Macro Function vs Regular Function vs Inline Functions
   https://embeddedinventor.com/c-macro-function-vs-regular-function-vs-inline-functions/

Thực hành

Viết thư viện:
compute.h và compute.c
Để thực hiện các hàm tính tổng (hàm), hiệu (hàm), tính diện tích hình tròn (function macro)
yêu cầu có nguyên tắc chống trùng lặp
Add vào project stm32.
Gọi các hàm ra để sử dụng.

—----------

ds1302, 1307, 3231

—----------

Viết thư viện ds1307
Tạo project và add thư viện ds1307 vào, đảm bảo compile không có lỗi. gọi hàm để cài đặt thời gian, đọc thời gian và debug để kiểm tra.

Add Thư viện

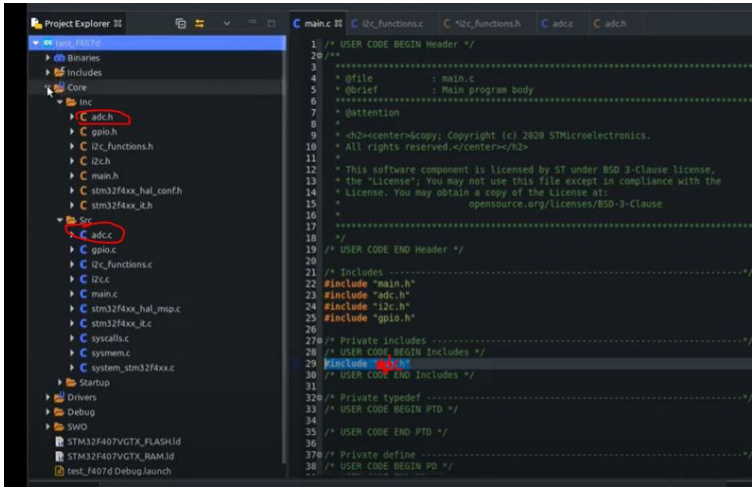
File .c và File .h

https://www.youtube.com/watch?v=MUZj4YwKVac

Cách 1:

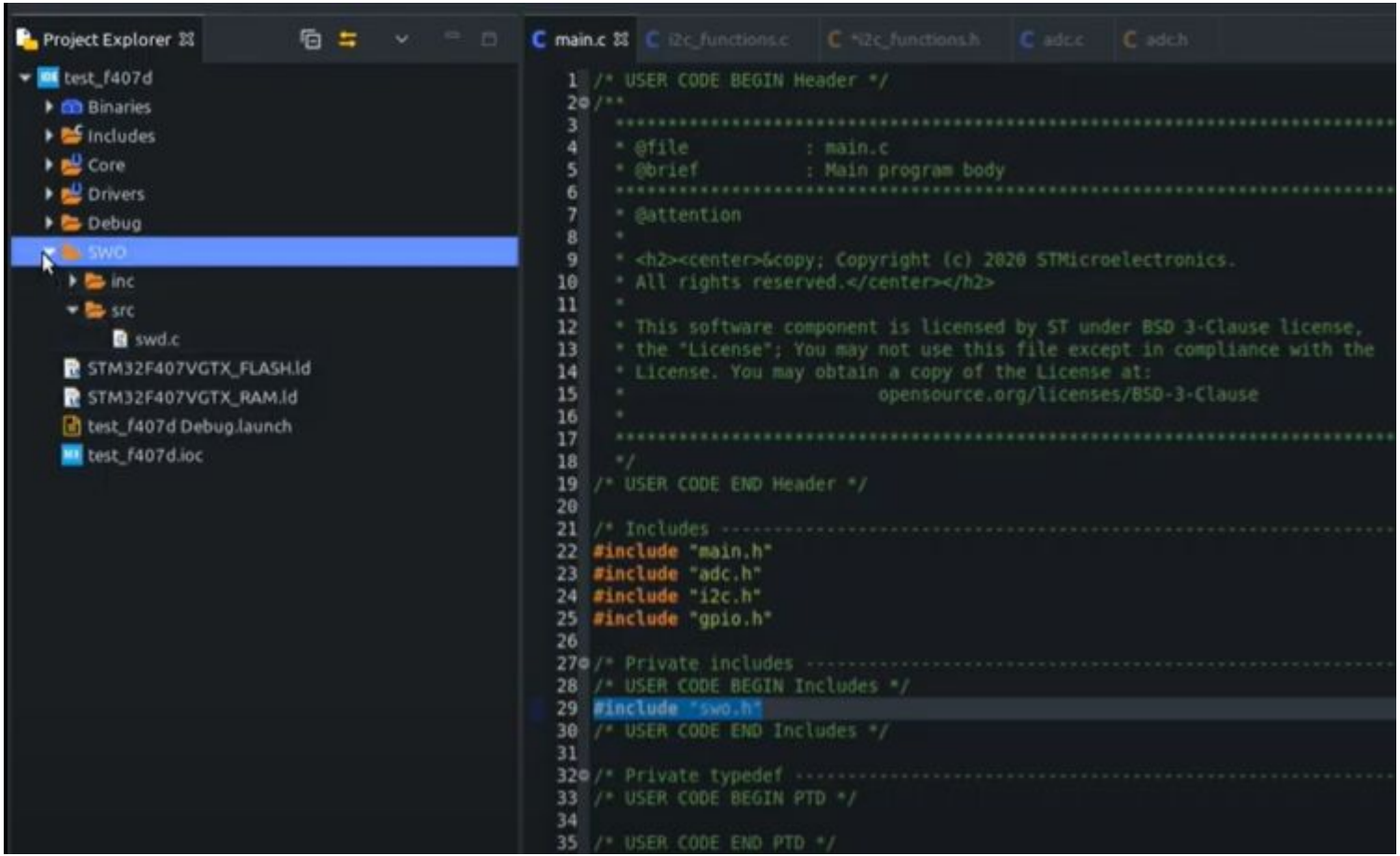
1. Vào thư mục project trong cửa sổ project explorer, vào thư mục core sẽ có thư mục Inc và Src
2. Lưu file .h của thư viện vào thư mục Inc và file .c vào thư mục Src
3. Include file .h vào file main.c để sử dụng thư viện

Cách này đơn giản, nhưng thư viện không được quản lý độc lập tại thư mục riêng



Cách 2:

1. Tạo 1 thư mục trong thư mục project chứa 2 thư mục inc và src để chứa file header và source của thư viện
2. Các thư viện này không được phần mềm STM32CubeIDE tự động nhận diện vì không nằm trong thư mục trong core
3. Cấu hình cho phần mềm nhận: Project -> Properties -> C/C++ general -> Path & Symbol
   + Include: Add, tick chọn is a workspace path -> workspace button -> chọn project đang làm việc -> chọn thư mục chứa thư viện -> chọn thư mục inc chứa header file -> ok
   + Source Location -> Add Folder -> Chọn thư mục chứa thư viện (chứ k phải thư mục src) -> Apply
   +


Project Explorer
test_f407d
Binaries
Includes
Core
Drivers
Debug
SWO
inc
src
swd.c
STM32F407VGTX_FLASH.ld
STM32F407VGTX_RAM.ld
test_f407d Debug.launch
test_f407d.ioc
main.c
i2c_functions.c
*i2c_functions.h
adc.c
adc.h
1 /* USER CODE BEGIN Header */
2 /**
3 **
4 * @file : main.c
5 * @brief : Main program body
6 **
7 * @attention
8 *
9 * <h2> © Copyright (c) 2020 STMicroelectronics.
10 * All rights reserved. </h2>
11 *
12 * This software component is licensed by ST under BSD 3-Clause license,
13 * the "License"; You may not use this file except in compliance with the
14 * License. You may obtain a copy of the License at:
15 * opensource.org/licenses/BSD-3-Clause
16 *
17 **
18 */
19 /* USER CODE END Header */
20
21 /* Includes --
22 #include "main.h"
23 #include "adc.h"
24 #include "i2c.h"
25 #include "gpio.h"
26
27 /* Private includes --
28 /* USER CODE BEGIN Includes */
29 #include "swo.h"
30 /* USER CODE END Includes */
31
32 /* Private typedef ---
33 /* USER CODE BEGIN PTD */
34
35 /* USER CODE END PTD */

Add Thư viện

File đã được biên dịch sẵn
file .a

Xem hướng dẫn tại mục 2.5.8 tài liệu STM32CubeIDE User guide

https://www.st.com/resource/en/user_manual/dm00629856-stm32cubeide-user-guide-stmicroelectronics.pdf

TAPiT

Hướng dẫn tạo và sử dụng một static library

https://www.youtube.com/watch?v=ab9Tj2LbRUo