OSX下编译STM32程序

大二的这个暑假，被同学拉着去参加了电设，由于比赛用到了`STM32`来作为处理器，所以就需要折腾一下`STM32`。由于我算半个果粉，所以免不了要去折(zhuang)腾(B)。大家都知道32这类的单片机的开发流程都是利用`C`或`C++`写成代码后，利用工具编译成32能够识别的二进制或16进制码写到32的`Flash`中。那么`windows`下能完成的流程，`OS X`下也一定能完成,嘻嘻。简述一下我们所需要的工具或者软件:-)

- `Macbook`或者装了黑苹果的电脑一台
- `STM32`开发板一块（笔者使用的是`F407Discovery`和`Alienware`生产的`F103VCT6`都已经过测试，不是这两种型号的小伙伴也不要着急，可以先试着读下去，也许你就有了灵感）
- `ST-Link`下载器一个（某宝盗版大约几块到几十不等，质量还算可以，但本人还是觉得支持正版比较好）

以上就是我们所需要的硬件设备了，其实并没有多复杂，很多做开发板的厂商喜欢用`CH340`这类的串口芯片来写入32的存储器，这样的方法我个人觉得巨慢无比，很多时候浪费的时间都够写一次FPGA芯片了。所以这里只使用最方便的工具`ST-Link`，至于串口烧写工具肯定是有，不过**不建议**这样去做。

剩下的就是软件工具了，开始之前，首先确定你的系统有这些依赖项：

- `libusb`， `libusb-compat`（看名字就知道了吧，是有关usb的依赖项）
- `pkg-config`（编译时期要用到，大概是编译过程中告诉编译器库信息的东东，manul手册这样写*Return metainformation about installed libraries*） 其实就是`GNU`的那一套玩意儿了
- `autoconf`， `automake`和`libtool`

如果某一项没有被安装利用`brew install <package>`解决就行啦。

想要在OS X上开发STM32最重要的工具当然是编译器了，对于开发ARM芯片当然也有对应的编译器，叫`arm-none-eabi-gcc`， `GNU`提供了一整套的工具链，GNU-ARM-Embedded-Toolchain上面可以下载到整个工具链并且有详细的安装过程，这里就不赘述了。最后别忘了加入到环境变量中

```
export PATH=xx/xxx/gcc-arm-none-eabi/bin:$PATH
```

我为了省事直接写到了`.zshrc`下，其实是个很不好的习惯。

有了上面的工具链，我们就可以愉快的编译代码了，最后只需要再安装一个st-link的命令行工具就行了，在github的README中有详细的安装和debug的过程，这里也不再赘述。只要自己找一个合适自己使用的Makefile文件就行，gcc会根据代码的依赖关系帮你编译成一个`.elf`文件，最后`arm-none-eabi-objcopy`利用这个`.elf`文件生成一个`.hex`或者`.bin`文件，这个要根据自己的需求来，不过根据笔者的经验.bin文件多半是烧写不进去的...囧。

对 `Makefile` 熟悉之后看这个会好一些，我也是从头到尾折腾了将近一个礼拜，F407这块开发板很好搞，现成的资料有很多，但是我这块淘宝买来的F103就很坑，店家给的教程连 `bootloader` 部分的代码都是店家自己写的...好在ST出的芯片所有的东西包括源代码，手册什么的都可以在ST官网下载得到。下面给出我自修改的一个 `F407` 的 `Makefile` 文件以及我自己文件树。为了方便阅读并与文件树对应，这里的Makefile未使用正则表达式。Makefile的注释应该比较清楚，动手自己改一个会对Makefile的结构更加理解...

```
# Original Author    :   Malkavian(His github repo is :
https://github.com/Malkavian/tuts.git)
# Modified           :   Eric(higuoxing@outlook.com)
# I modified his makefile to make stm32 easy to be programmed under
Linux or OS X
# Moreover, I could learn more about stm32 programming from this simple
makefile

# Makefile for STM32F4Discovery
# My File tree...
# STM32F4DiscoveryBlankProject
#    └──STM32F4-Discovery_FW_V1.1.0     #注：这个是官网直接下载的标准库，不同芯片
#    └──blank
#        ├── Makefile
#        ├── main.c
#        ├── stm32_flash.ld
#        ├── stm32f4xx_conf.h
#        └── system_stm32f4xx.c

# Name of your project
PROJECT_NAME = BLANK

# Project root path
PROJECT_ROOT_PATH = ..
STM_STD_LIB_PATH = $(PROJECT_ROOT_PATH)/STM32F4-Discovery_FW_V1.1.0

# Third-party libraries should be add here
# Example: TM_STM_LIB_PATH = $(PROJECT_ROOT_PATH)/TM_STM32F4xxLIBRARIES

# STM header files path should be registered here
# Standard libraries inc file path
STM_INC_PATH  = $(STM_STD_LIB_PATH)/Utitlties/STM32F4-Discovery
STM_INC_PATH += $(STM_STD_LIB_PATH)/Libraries/CMSIS/Include
STM_INC_PATH += $(STM_STD_LIB_PATH)/Libraries/CMSIS/ST/STM32F4xx/Include
STM_INC_PATH +=
$(STM_STD_LIB_PATH)/Libraries/STM32F4xx_StdPeriph_Driver/inc
STM_INC_PATH += .

# TM libraries inc file path
STM_INC_PATH += $(TM_STM_LIB_PATH)
```

```
# STM source files path should be registered here
STM_SRC_PATH  =
$(STM_STD_LIB_PATH)/Libraries/STM32F4xx_StdPeriph_Driver/src
STM_SRC_PATH += $(TM_STM_LIB_PATH)

vpath %.c $(STM_SRC_PATH)

#main source file
SRCS          = main.c
SRCS         += system_stm32f4xx.c
SRCS         +=
$(STM_STD_LIB_PATH)/Libraries/CMSIS/ST/STM32F4xx/Source/Templates/TrueST
UDIO/startup_stm32f4xx.s

# dependencies must be declared here
# EXAMPLE     :   SRCS += stm32f4xx_gpio.c
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# ======================STD_LIB=====================================
# SRCS         += stm32f4xx_adc.c
# SRCS         += stm32f4xx_can.c
# SRCS         += stm32f4xx_crc.c
# SRCS         += stm32f4xx_cryp_aes.c
# SRCS         += stm32f4xx_cryp_des.c
# SRCS         += stm32f4xx_cryp_tdes.c
# SRCS         += stm32f4xx_cryp.c
# SRCS         += stm32f4xx_dac.c
# SRCS         += stm32f4xx_dbgmcu.c
# SRCS         += stm32f4xx_dcmi.c
# SRCS         += stm32f4xx_dma.c
# SRCS         += stm32f4xx_exti.c
# SRCS         += stm32f4xx_flash.c
# SRCS         += stm32f4xx_fsmc.c
# SRCS         += stm32f4xx_gpio.c
# SRCS         += stm32f4xx_hash_md5.c
# SRCS         += stm32f4xx_hash_sha1
# SRCS         += stm32f4xx_hash.c
# SRCS         += stm32f4xx_i2c.c
# SRCS         += stm32f4xx_iwdg.c
# SRCS         += stm32f4xx_pwr.c
# SRCS         += stm32f4xx_rcc.c
# SRCS         += stm32f4xx_rng.c
# SRCS         += stm32f4xx_rtc.c
# SRCS         += stm32f4xx_sdio.c
# SRCS         += stm32f4xx_spi.c
# SRCS         += stm32f4xx_syscfg.c
# SRCS         += stm32f4xx_tim.c
# SRCS         += stm32f4xx_usart.c
# SRCS         += stm32f4xx_wwdg.c
```

```
# ==================third-party-lib================================
# Example: SRCS += xxx.c

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#########SHOULD NOT BE CHANGED UNLESS PROFESSIONAL###################

#####################################################################
#                         TOOLCHAINS CONFIG                         #
#####################################################################
#Toolchain has been exported
# The tool we use
CC      = arm-none-eabi-gcc
OBJCOPY = arm-none-eabi-objcopy
GDB     = arm-none-eabi-gdb

## Preprocessor options

# directories to be searched for header files
INCLUDE = $(addprefix -I,$(STM_INC_PATH))

# #defines needed when working with the STM library
DEFS    = -DUSE_STDPERIPH_DRIVER
# if you use the following option, you must implement the function
#    assert_failed(uint8_t* file, uint32_t line)
# because it is conditionally used in the library
# DEFS   += -DUSE_FULL_ASSERT

## Compiler options
CFLAGS  = -ggdb
# please do not optimize anything because we are debugging
CFLAGS += -O0
CFLAGS += -Wall -Wextra -Warray-bounds
CFLAGS += -mlittle-endian -mthumb -mcpu=cortex-m4 -mthumb-interwork
CFLAGS += -mfloat-abi=hard -mfpu=fpv4-sp-d16

## Linker options
# tell ld which linker file to use
# (this file is in the current directory)
LFLAGS  = -Tstm32_flash.ld


#####################################################################
#                           SETUP TARGETS                           #
#####################################################################

.PHONY: $(PROJECT_NAME)
$(PROJECT_NAME): $(PROJECT_NAME).elf
```

```
$(PROJECT_NAME).elf: $(SRCS)
    $(CC) $(INCLUDE) $(DEFS) $(CFLAGS) $(LFLAGS) $^ -o $@
    $(OBJCOPY) -O ihex $(PROJECT_NAME).elf   $(PROJECT_NAME).hex
    $(OBJCOPY) -O binary $(PROJECT_NAME).elf $(PROJECT_NAME).bin

.PHONY: clean
clean:
    rm -f *.o $(PROJECT_NAME).elf $(PROJECT_NAME).hex
$(PROJECT_NAME).bin

# Flash the STM32F4
flash:
    st-flash write $(PROJECT_NAME).bin 0x8000000

.PHONY: debug
debug:
# before you start gdb, you must start st-util
    $(GDB) $(PROJECT_NAME).elf
```