

## 如何让电子时钟跑起 Bootloader 和 ota 升级

### 一、前期准备

硬件准备：

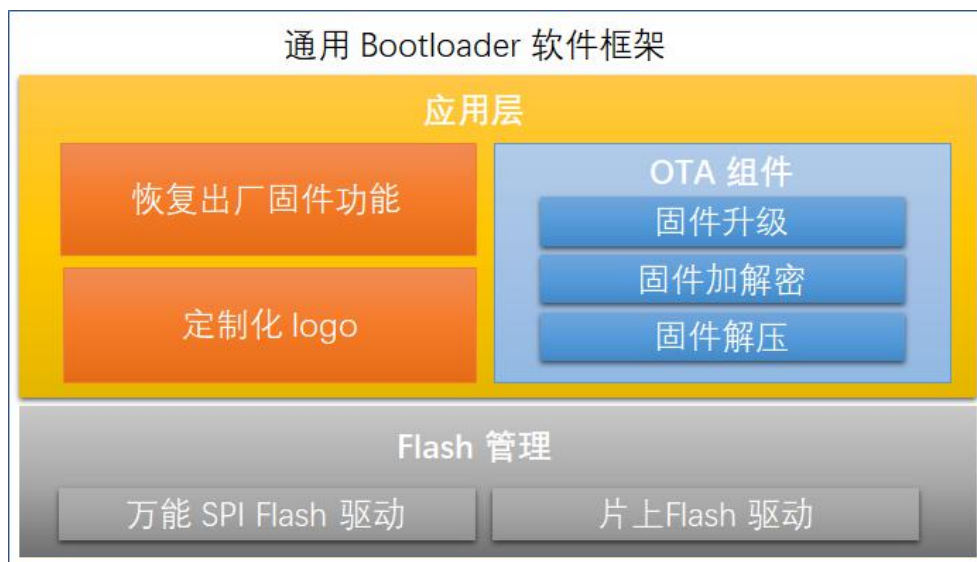
1. 一块 LED 电子数字时钟屏
2. 一根 microusb 线（安卓线）
3. 一个下载器（ST-Link 或者 Jlink 等）

软件准备：

4. MDK5 软件
5. RT-Thread 的 ENV 工具
6. 源代码项目
7. RT-Thread OTA 打包工具
8. WebServer 工具

### 二、主要说明

在本次操作中的 LED 电子数字时钟屏采用的主控芯片是 STM32F401RCT6, FLASH 为 256K, 焊接了一个外置 SPI 接口的 flash 芯片；在添加 bootloader 功能前需要设计分区，对于分区操作有一定的要求说明，暂且不细说，参考 RT-Thread 官网的说明如下，可以对 Boot 有一定的了解。



RT-Thread 通用 Bootloader 有如下特点：

- 以 bin 文件的形式提供，无需修改即可使用
- 资源占用小，ROM 最小只需要 16KB，最大 32KB
- 适用于多系列 STM32 芯片（目前支持 F1 和 F4 系列）
- 支持各种 SPI Flash 存储固件
- 支持固件加解密功能
- 支持多种固件压缩方式
- 支持恢复出厂固件功能
- 以上功能均可自由配置

## 功能说明

Bootloader 的主要功能是更新 app 分区中的固件。

## 分区表介绍

通用 Bootloader 中的分区表包含如下三个分区：

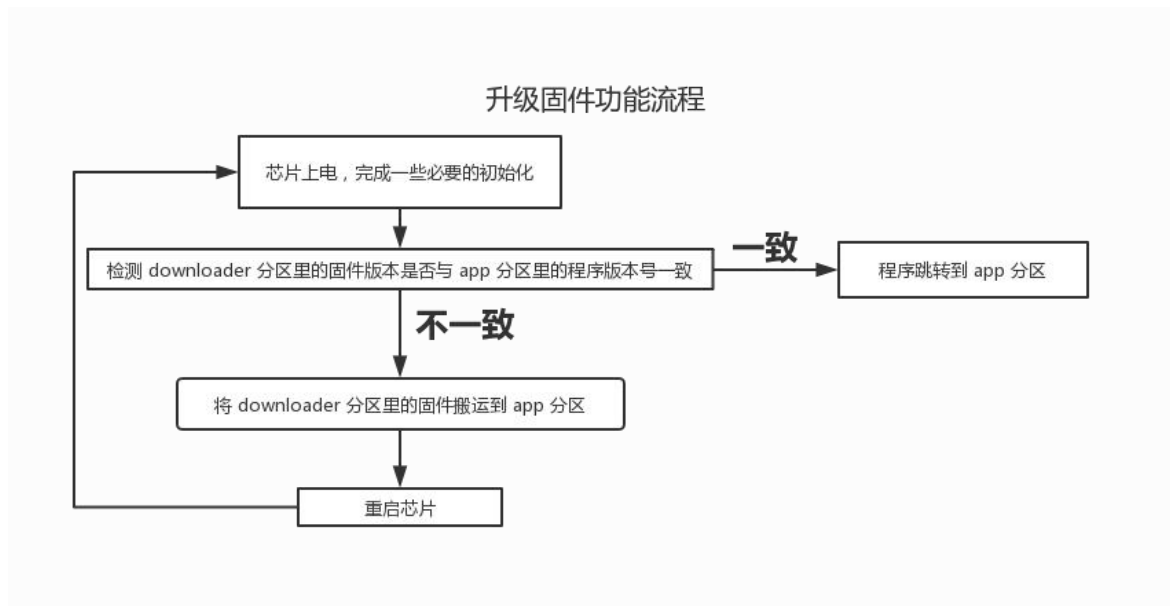
分区名	起始地址	分区大小	分区位置	介绍	
app	自定义	自定义	片内 Flash	存储 app 固件	
download	自定义	自定义	片内 Flash 或者片外 SPI Flash	存储待升级固件	
factory	自定义	自定义	片内 Flash 或者片外 SPI Flash	存储出厂固件	

## 升级固件功能

当系统需要升级固件时，Bootloader 将从 download 分区将固件搬运到 app 分区，主要功能流程如下所示：

1. Bootloader 启动时检查 download 分区和 app 分区中的固件版本。
2. 如果两个固件版本相同，则跳转到 app 分区，Bootloader 运行结束。
3. 固件版本不同则将 download 分区中的固件搬运到 app 分区。
4. 在搬运的过程中 Bootloader 可以对固件进行校验、解密、解压缩等操作。
5. 搬运完毕后，删除 download 分区中存储的固件。
6. 重启系统跳转到 app 分区中的固件运行，Bootloader 运行结束。

Bootloader 工作过程如下图所示：



## 恢复固件功能

当系统中的固件损坏, Bootloader 将从 `factory` 分区将固件搬运到 `app` 分区, 主要功能流程如下所示:

1. Bootloader 启动时检查触发固件恢复的引脚是否为有效电平。
2. 如果有效电平持续超过 10S 则将 `factory` 分区中的固件搬运到 `app` 分区中。
3. 如果有效电平没有持续超过 10S 则继续进行 2.2 小节中介绍的启动步骤。
4. 在搬运的过程中 Bootloader 可以对固件进行校验、解密、解压缩等操作。
5. 搬运完毕后, 保持 `factory` 分区中的固件不变。
6. 重启系统跳转到 `app` 分区中的固件运行, Bootloader 运行结束。

以上资料来源: RT-Thread 官方文档, 详细可以参考:

<https://www.rt-thread.org/document/site/application-note/system/rtboot/an0028-rtboot/>

## 三、操作流程

分区说明: 基于使用的 STM32F401RCT6 的内部 Flash 为 256K, 开源 LED 电子时钟屏的源程序的固件大小为 124k, 在设计分区的时候一般 APP 和 download 区是采用 1:1 的方式, 还有 bootloader 分区一般设计为 16K 到 32K, 显然这样 256K 的内部 FLASH 不能满足要求, 所以需要使用外部 FLASH, 设计分区表为: 内部 flash 的前 32K 为 bootloader 分区, 剩余的内部 FLASH  $(256-32 = 224)$  224K 为 APP 分区, download 分区设在外部 flash 区。

在设计好分区表后, 需要在原项目代码中添加 spi 的驱动, 烧录 boot, 修改分区表, 修改连接地址, 添加 ota\_downloader 的功能组件。

1. 在源程序项目代码中配置添加 spi——flash 驱动。

(1) 在项目代码的目录下，通过 ENV 工具进入命令行，在命令行中输入 menuconfig，如下图 1 所示：

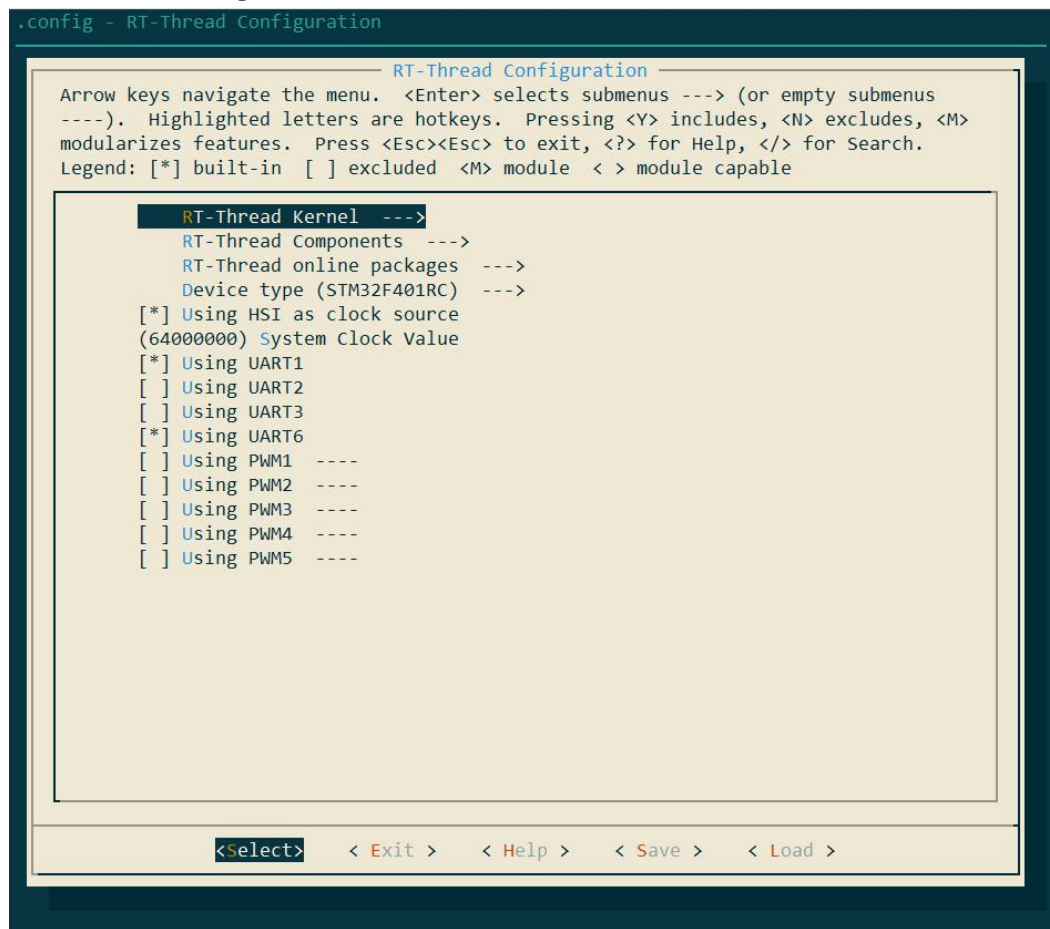
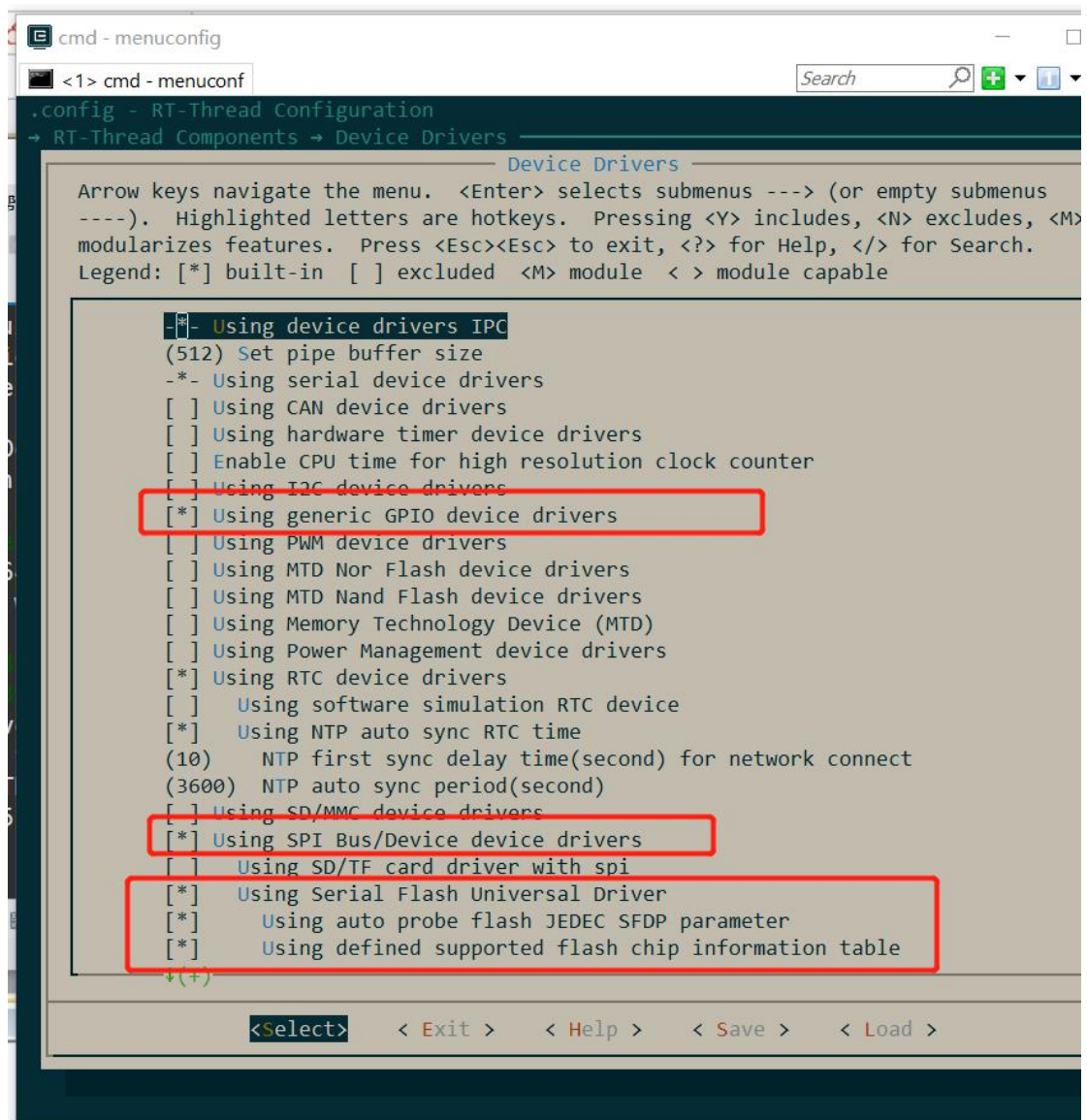


图 1 menuconfig 配置界面

进入 RT-Thread Components 目录下的 Device Drivers 下选中 Using generic GPIO device drivers 和 Using SPI Bus/Device device drivers 和 Using Serial Flash Universal Driver 和 Using auto probe flash JEDEC SFDP parameter 和 Using defined support flash chip information tatble，如下图 2 所示：



添加配置后，返回到主界面，修改 CS 的引脚号为 20.



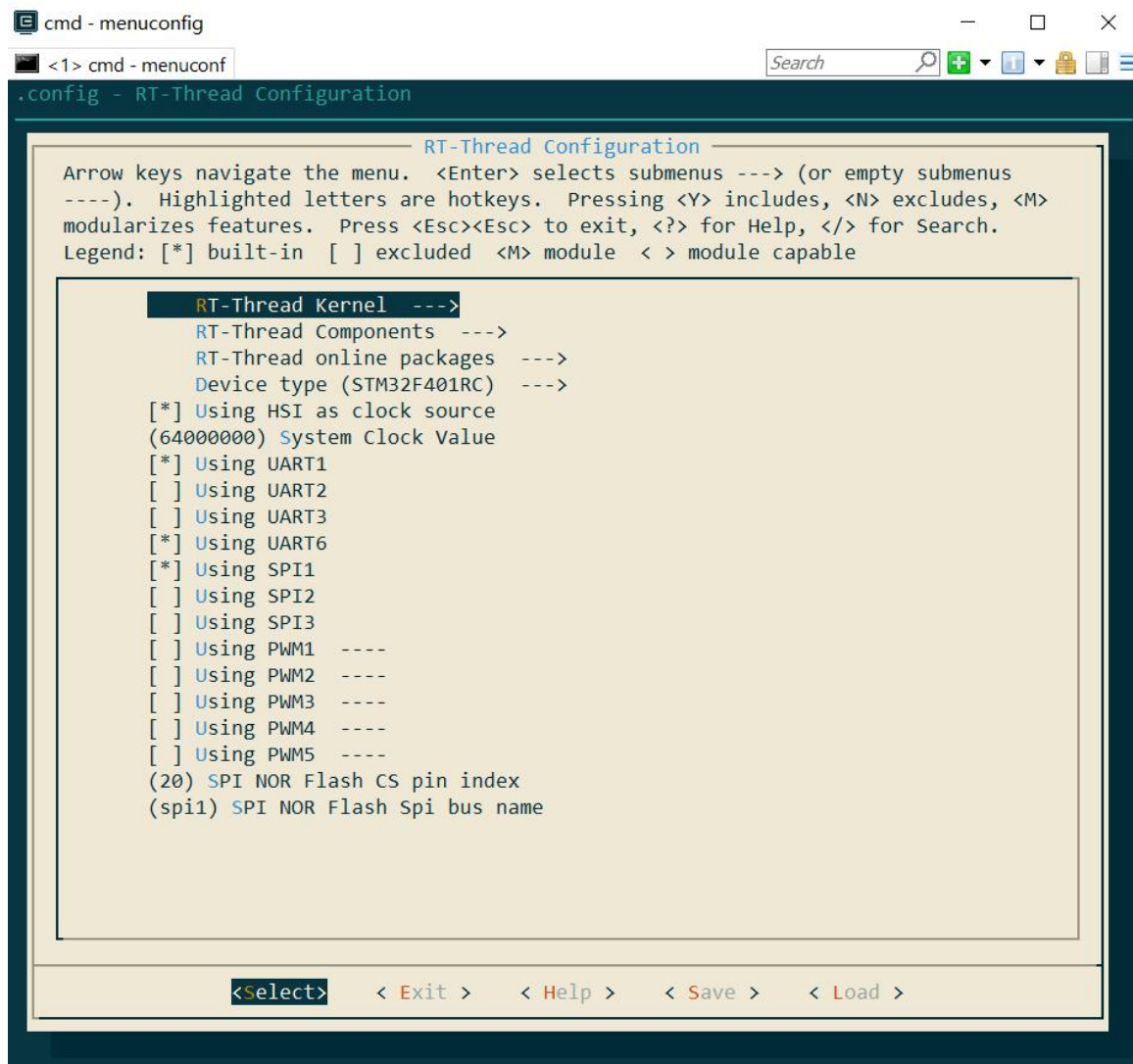


图 2 配置 SPI 驱动和 GPIO 驱动

保存并退出，在 env 命令行下输入 `scons --target=mdk5` 重新编译生成工程。

再用 mdk5 打开编译的项目工程，编译烧录到板子上，在启动信息可以判断外置 flash 已经添加驱动了，如下图 3 所示：

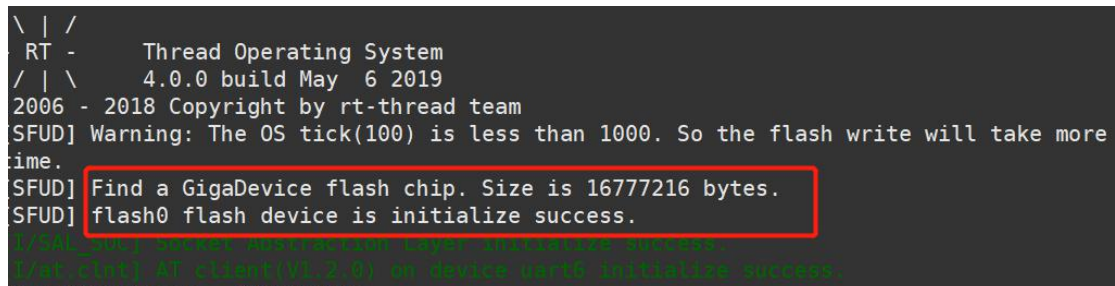


图 3 启动打印 spi flash 芯片信息

(2) 添加 `ota_downloader` 和 `FAL_Packages` 软件包还有 SFUD 的配置。

在 env 工具的命令中，输入 menuconfig 进入配置界面，进入 RT-Thread online packages 目录下的 Iot - internet of things 目录下选中[\*]ota\_downloader，进入 ota\_downloader 目录下选中[\*] Enable OTA downloader debug 和[\*] Enable HTTP/HTTPS OTA（并且配置默认的 URL 为本机的 ip 地址的路径）和[\*] Enable Ymodem OTA，如下图 4 和图 5 所示

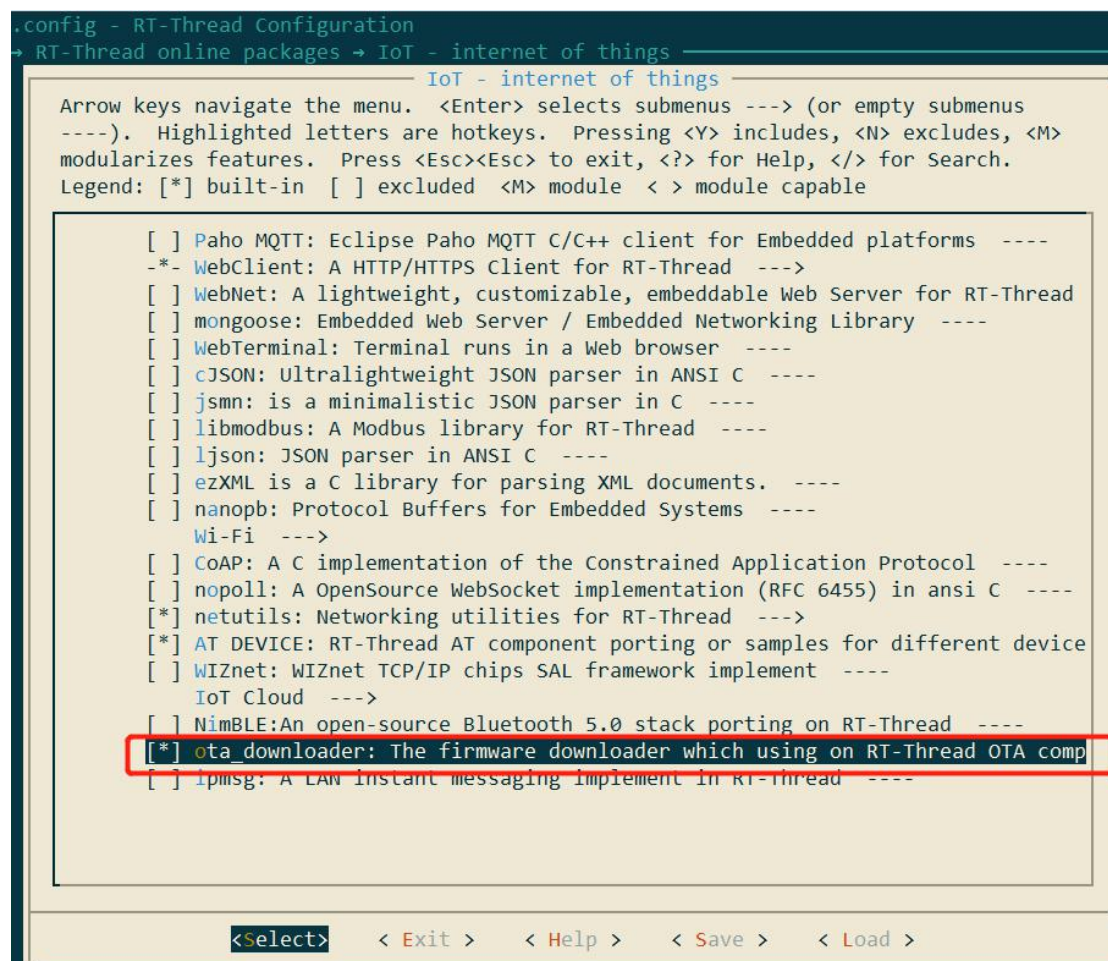


图 4 添加 ota\_downloader 软件包

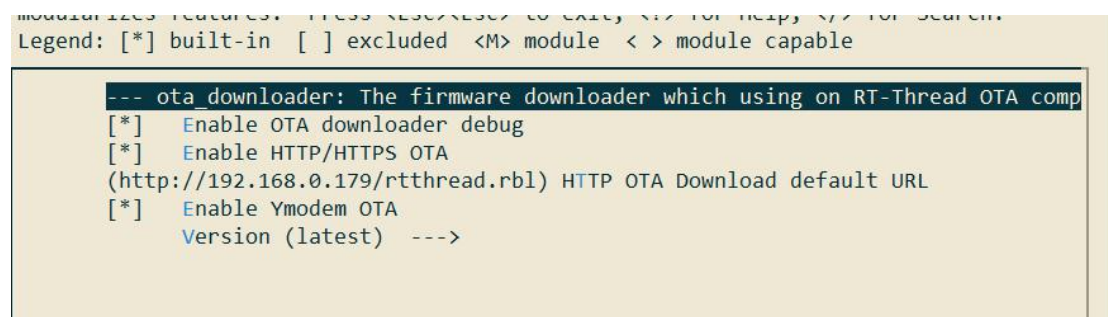


图 5 配置 ota\_downloader 软件包添加 ymodem\_ota 和 http\_ota

通过 ESC 按键返回到 RT-Thread online packages 目录下 system packages 目录下选中 fal: Flash Abstraction layer..., 如下图 6 所示

```
config - RT-Thread Configuration
RT-Thread online packages → system packages ————— system packages —————

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus
----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M>
modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable

[[ ]] Enable GUI Engine -----
[ ] Cairo - Multi-platform 2D graphics library -----
[ ] pixman is a library that provides low-level pixel manipulation -----
[ ] lwext4: an excellent choice of ext2/3/4 filesystem for microcontrollers.
[ ] partition: A simple partition for block device in rt-thread. -----
[*] fal: Flash Abstraction Layer implement. Manage flash device and partition
[ ] SQLite: a self-contained, high-reliability, embedded, full-featured, publ
[ ] RT-Thread Insight: probe tool for RT-Thread -----
[ ] LittlevGL2RTT: The LittlevGl gui lib adapter RT-Thread -----
[ ] CMSIS: Cortex Microcontroller Software Interface Standard from ARM -----
[ ] yaffs: Yet Another Flash File System -----
[ ] Littlefs: A high-integrity embedded file system -----
[ ] thread_pool: A thread pool base on RT-Thread -----
```

图 6 添加 FAL 软件包

进入选中该项 fal 目录下, 选中 FAL partition table config has defined on 'fal\_cfg,h' 和 FAL uses SFUD driver, 并且修改 name 为 flash0 具体配置如下图 7 所示



```
.config - RT-Thread Configuration
[...] packages → fal: Flash Abstraction Layer implement. Manage flash device and partition
fal: Flash Abstraction Layer implement. Manage flash device and partition.
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus
----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M>
modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module < > module capable

-- fal: Flash Abstraction Layer implement. Manage flash device and partition
[*] Enable debug log output
[*] FAL partition table config has defined on 'fal_cfg.h'
[*] FAL uses SFUD drivers
(flash0) The name of the device used by FAL
version (latest) --->
```

图 7 配置 fal 软件包

然后保存并退出。在 ENV 工具的命令行中输入 `pkgs --update` 软件更新的指令，等下载完成后，就可以输入：`scons --target=mdk5` 重新编译生成新的工程。

### （3）添加 fal\_cfg.h 和 fal\_flash\_stm32f4\_port.c 和更新 SFUD 的驱动文件

由于原项目代码版本的问题，需要在新版的 RT-Thread 的源代码中复制两个文件到现工程代码中。

把 fal\_cfg.h 和 fal\_flash\_stm32f4\_port.c 这两个文件复制到现代码项目中的 driver 目录下，并且把这两个文件添加到项目工程中；更新 SFUD 的驱动文件，对比新版 git 库中

rt-thread/components/drivers/spi/spi\_flash\_sfud.c 和 rt-thread/components/drivers/spi/spi\_flash\_sfud.h 的文件，可以直接复制覆盖这两个文件的内容。

（4）烧写 boot.bin 文件，boot.bin 文件可以是烧录附件中的 boot.bin 文件，也可以是 RT-Thread 官网上通过网页生成的 boot.bin，下面主要使用附件中的 boot.bin 文件的使用。

烧录 boot.bin 文件主要使用 ST-Link Utility 工具。

步骤 1: 打开 STM32 ST-LINK Utility 工具，打开附件中的 bootloader.bin 文件，如下图 8 所示

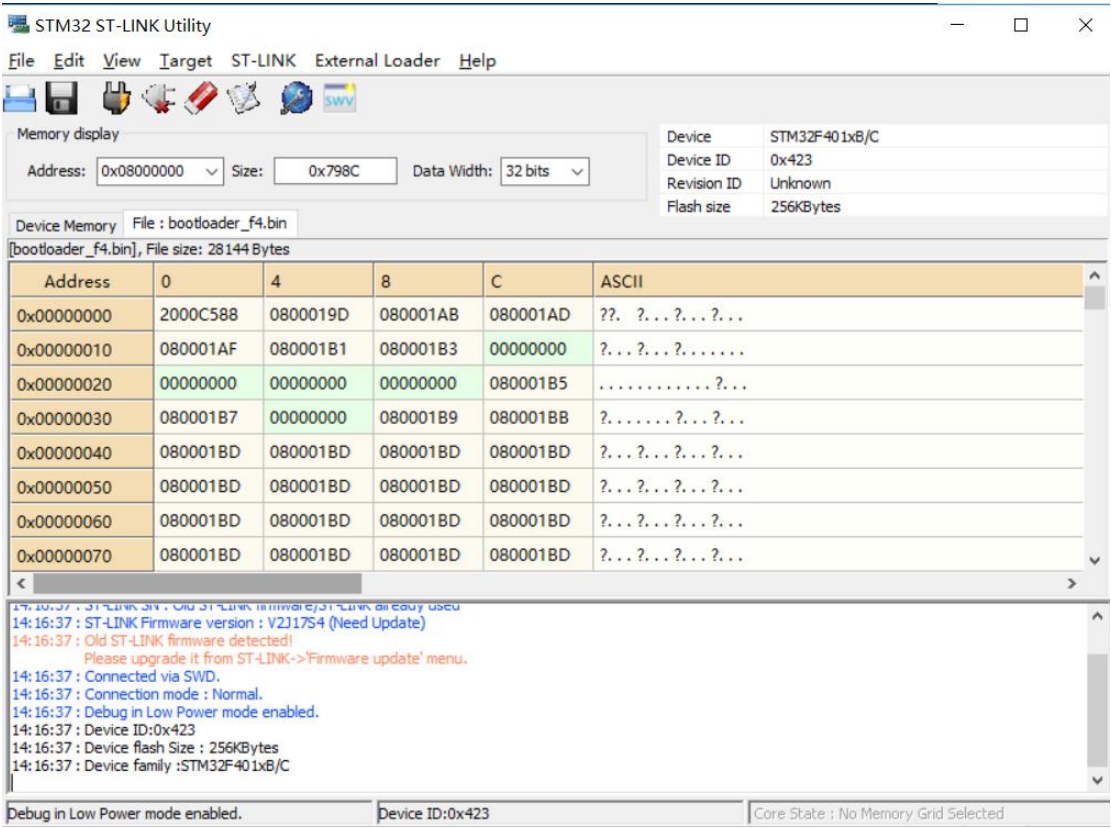


图 8 使用 ST-Link Utility 工具烧录 boot.bin

然后，选择菜单栏上的烧录按钮，设置 start address 为 0x08000000 为默认地址，然后选择 start 按钮，开始烧录，如下图 9 所示。

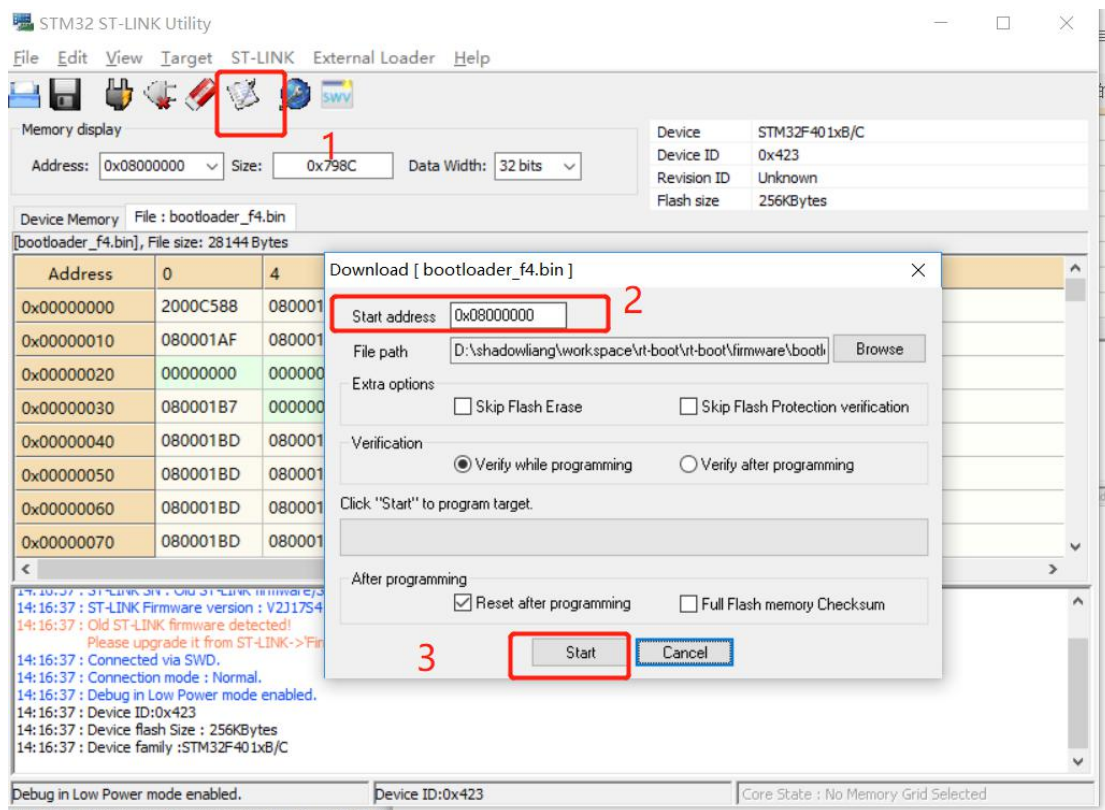


图 9 设置 boot. bin 的起始地址

烧录完成后，在串口中会打印 boot 的信息，如分区表的信息等，如下图所示

```
[SFUD]Find a GigaDevice flash chip. Size is 16777216 bytes.
[SFUD]spi flash device is initialize success.

RT-BOOT
2006 - 2019 Copyright by rt-thread team
0.9.1 build Apr 13 2019

[D/FAL] (fal_flash_init:61) Flash device | onchip_full_16k_part | addr: 0x08000000 | len: 0x00040000 | blk_size: 0x00004000 | initialized
finish.
[D/FAL] (fal_flash_init:61) Flash device | onchip_flash | addr: 0x08020000 | len: 0x000e0000 | blk_size: 0x00020000 | initialized
finish.
[D/FAL] (fal_flash_init:61) Flash device | nor_flash | addr: 0x00000000 | len: 0x01000000 | blk_size: 0x00001000 | initialized
finish.

[D/FAL] ===== RT-Thread Flash Partition Table =====
[D/FAL] | name | Flash_dev | offset | length |
[D/FAL] |-----|-----|-----|-----|
[D/FAL] | app | onchip_full_16k_part | 0x00000000 | 0x00040000 |
[D/FAL] | download | nor_flash | 0x00000000 | 0x00000000 |
[D/FAL] =====
[D/FAL] RT-Thread Flash Abstraction Layer (V0.4.0) initialized success.
[D/FAL] System initialization success.
[I]RT-Thread OTA package(V0.2.1) initialize success.
[E]The partition APP was not found
[E]Get firmware header occur CRC32(calc.crc: 7b93c5c8 != hdr.info_crc32: ffffffff) error on 'app' partition!
[I]Begin to execute the program on app partition.
[E/FAL] Can't find user firmware on app partition. (Address: 0x08008000, Data: 0xffffffff)
[E/FAL] Bootloader stop.
.....
```

图 10 烧录 boot. bin 后启动

由于 APP 固件的启动地址还没有修改，所以会提示找不到 APP 分区，到此 bootloader 已经成功启动了，下面开始 APP 固件的启动地址进行修改。

### (5) APP 固件分区的启动地址修改

步骤 1: 在 main.c 中添加 FAL 初始化代码和修改中断向量跳转地址, 添加版本打印信息, 具体操作如下图 11 所示。

```
//添加fal的头文件
#include "fal.h"

//添加版本信息
#define APP_VERSION "1.0.0"

//添加中断向量表的跳转地址
/**
 * Function    ota_app_vtor_reconfig
 * Description Set Vector Table base location to the start addr of app(RT_APP_PART_ADDR).
 */
static int ota_app_vtor_reconfig(void)
{
    #define NVIC_VTOR_MASK    0x3FFFFFF80
    /* Set the Vector Table base location by user application firmware definition */
    SCB->VTOR = RT_APP_PART_ADDR & NVIC_VTOR_MASK;

    return 0;
}
INIT_BOARD_EXPORT(ota_app_vtor_reconfig);

//在main函数中添加FAL初始化代码和打印版本信息。
int main(void)
{
    ....
    fal_init();
    rt_kprintf("The current version of APP firmware is %s\n", APP_VERSION);
    ....
}
```

图 11

步骤 2: 修改 APP 启动地址为 0x08008000, 如下图 12 所示。

主要修改fal\_cfg.h和stm32\_rom.sct的内容。修改fal\_cfg.h的内容:

```
#define RT_APP_PART_ADDR 0x08008000
```

修改stm32\_rom.sct的内容:

```
; *****
; *** Scatter-Loading Description File generated by uVision ***
; *****

LR_IROM1 0x08008000 0x00100000 { ; load region size_region
    ER_IROM1 0x08008000 0x00100000 { ; load address = execution address
        *.o (RESET, +First)
        *(InRoot$$Sections)
        .ANY (+R0)
    }
    RW_IRAM1 0x20000000 0x00020000 { ; RW data
        .ANY (+RW +ZI)
    }
}
```

图 12

步骤 3 添加屏蔽中断向量跳转到默认地址的语句, 如下图 13 所示



主要在system\_stm32f4xx.c文件下修改。

```
/* Configure the Vector Table location add offset address -----*/
#ifdef VECT_TAB_SRAM
// SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#else
// SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH */
#endif
```

到这一步可以正常从内部flash中分区的bootloader启动app.

图 13

(6) 分区表 fal\_cfg.h 的修改  
步骤参考，如下图 14 所示：

主要修改fal\_cfg.h文件,参考原来的添加一个nor\_flash0设备。

```
extern struct fal_flash_dev nor_flash0;

/* flash device table */
#define FAL_FLASH_DEV_TABLE
{
    &stm32_onchip_flash_16k,
    &stm32_onchip_flash_64k,
    &stm32_onchip_flash_128k,
    &nor_flash0,
}
```

还有修改分区表的名称

```
/* partition table */
#define FAL_PART_TABLE
{
    {FAL_PART_MAGIC_WROD, "download", "flash0", 0, (224 * 1024), 0}, \
    {FAL_PART_MAGIC_WROD, "app", "onchip_flash_128k", 0, (128 * 1024), 0}, \
}
```

注意分区表的名称要和bootloader分区表的一致。

图 14

到此为止，bootloader 和 ota 升级功能已经添加完成，重新编译工程，下载到板子上即可看到程序正常运行。板子启动打印的信息如下图所示，验证了 bootloader 已经正常工作，而且成功跳转到了 app 固件分区中，如下图 15 所示。



```

  _ _ _ _ _
 | | | | |
 | | | | |
  _ _ _ _ _
2006 - 2019 Copyright by rt-thread team
          0.9.1 build Apr 13 2019
[D/FAL] (fal_flash_init:61) Flash device | onchip_full_16k_part | addr: 0x08000000 | len: 0x00040000 | blk_size: 0x00004000 | initialized
finish.
[D/FAL] (fal_flash_init:61) Flash device | onchip_flash | addr: 0x08020000 | len: 0x000e0000 | blk_size: 0x00020000 | initialized
finish.
[D/FAL] (fal_flash_init:61) Flash device | nor_flash | addr: 0x00000000 | len: 0x01000000 | blk_size: 0x00001000 | initialized
finish.
[D/FAL] ===== FAL partition table =====
[D/FAL] | name      | flash_dev | offset | length |
[D/FAL] |-----|-----|-----|-----|
[D/FAL] | app       | onchip_full_16k_part | 0x00000000 | 0x00040000 |
[D/FAL] | download  | nor_flash | 0x00000000 | 0x00010000 |
[D/FAL] |-----|-----|-----|-----|
[D/FAL] RT-Thread Flash Abstraction Layer (V0.3.0) initialize success.
[D/FAL] System initialization successfully.
[I]RT-Thread OTA package(V0.2.1) initialize success.
[E]Get firmware header occur CRC32(calc.crc: 7b93c5c8 != hdr.info_crc32: ffffffff) error on 'download' partition!
[E]Get OTA download partition firmware header failed!
[E]Get firmware header occur CRC32(calc.crc: 7b93c5c8 != hdr.info_crc32: ffffffff) error on 'app' partition!
[I]Begin to execute the program on app partition.
[D/FAL] Find user firmware on app partition 0x00000000 successfully.
[D/FAL] Bootloader jumps to user firmware now.

\ | /
- RT -   Thread Operating System
/ | \    4.0.0 build May 6 2019
2006 - 2018 Copyright by rt-thread team
[SFUD] Warning: The OS tick(100) is less than 1000. So the flash write will take more time.
[SFUD] Find a GigaDevice flash chip. Size is 16777216 bytes.
[SFUD] flash0 flash device is initialize success.
[D/SU-80C] Socket Abstraction Layer initialize success.
[D/rt-thread] AT client(V0.3.0) on device 4405 initialize success.
SystemCoreClock = 64000000Hz

```

图 15

而且在程序的命令行中也看到了 ymodem\_ota 和 http\_ota 的指令，如下图所示。

```

RT-Thread shell commands:
led                - led test
pwm                - pwm test
at_net_init        - initialize AT network
at_ping            - AT ping network host
fal                - FAL (Flash Abstraction Layer) operate.
ntp_sync           - Update time by NTP(Network Time Protocol): ntp_sync [host_name]
http_ota           - Use HTTP to download the firmware
ymodem_ota         - Use Y-MODEM to download the firmware
wget               - Get file by URI: wget <URI> <Filename>.
list_fd            - list file descriptor
date               - get date and time or set [year month day hour min sec]
sf                 - SPI Flash operate.
version            - show RT-Thread version information
list_thread        - list thread
list_sem           - list semaphore in system
list_event         - list event in system
list_mutex         - list mutex in system
list_mailbox       - list mail box in system
list_msgqueue      - list message queue in system
list_memheap       - list memory heap in system
list_mempool       - list memory pool in system
list_timer         - list timer in system
list_device        - list device in system
help               - RT-Thread shell help.
ls                 - List information about the FILES.
cp                 - Copy SOURCE to DEST.
mv                 - Rename SOURCE to DEST.
cat                - Concatenate FILE(s)
rm                 - Remove(unlink) the FILE(s).
cd                 - Change the shell working directory.
pwd                - Print the name of the current working directory.
mkdir              - Create the DIRECTORY.
mkfs               - format disk with file system
df                 - disk free
echo               - echo string to file
ps                 - List threads in the system.

```

图 16

#### 四、功能验证

##### 1. ymodem\_ota 升级功能验证。

操作步骤:

(1) 采用在 项目代码目录 下面的 packages\ota\_downloader-latest\tools\ota\_packager 文件夹下 双击运行 rt\_ota\_packaging\_tool.exe, 如下图 17 所示

名称	修改日期	类型	大小
fastlz.dll	2019/5/6 11:26	应用程序扩展	12 KB
quicklz150_32_3.dll	2019/5/6 11:26	应用程序扩展	34 KB
rt_ota_packaging_tool.exe	2019/5/6 11:26	应用程序	119 KB
rt_ota_packaging_tool.exe.config	2019/5/6 11:26	Configuration 源...	1 KB

图 17

在软件中的配置如下图 18 所示，点击选择固件，指定项目工程代码编译生成的 rtthread-stm32f4xx.bin 文件，采用不加密不压缩的方法，固件分区名为” app”，固件版本为数字序号以区分不同的版本，然后点击开始打包，即可在固件目录下生成一个 rbl 后缀的文件。

**RT-Thread OTA 固件打包器**

**选择固件** D:\shadowliang\workspace\stm32f4xx-HAL-led\LED\_Matri

**保存路径** D:\shadowliang\workspace\stm32f4xx-HAL-led\LED\_Matri

**压缩算法** 不压缩

**加密算法** 不加密

**加密密钥**

**加密 IV**

**固件分区名** app **固件版本** 1525

**结果：**

HASH\_CODE RAW\_SIZE :

HDR\_CRC32 : PKG\_SIZE :

BODY\_CRC32: TIMESTAMP :

**开始打包**

COPYRIGHT (C) 2012-2018, Shanghai Real-Thread Technology Co., Ltd Ver: 1.0.7

图 18

打包生成的 rbl 文件，如下图 19 所示。




 rtconfig.pyc	2018/12/27 15:37	Compiled Pytho...	4 KB
 rtthread-stm32f4xx.bin	2019/5/6 14:48	BIN 文件	129 KB
 rtthread-stm32f4xx.rbl	2019/5/6 15:02	RBL 文件	129 KB
 SConscript	2018/10/18 18:37	文件	1 KB

图 19

(2) 在板子的串口终端中输入：ymodem\_ota 命令，如下图 20 所示

```
msh />y,  
msh />y  
ymodem_ota  
msh />ymodem_ota  
Default save firmware on download partition.  
Warning: Ymodem has started! This operator will not recovery.  
Please select the ota firmware file and use Ymodem to send.  
C
```

图 20

然后，利用串口终端的 Ymodem 传输工具发送，方法为在 Xshell 串口终端中，鼠标右键选择传输，再选择 YMODEM，再选择 YMODEM 发送，指定发送的 rbl 文件即可。

接下来就会自动进入下载程序到板子上进行升级，部分过程图如下图 21 所示。

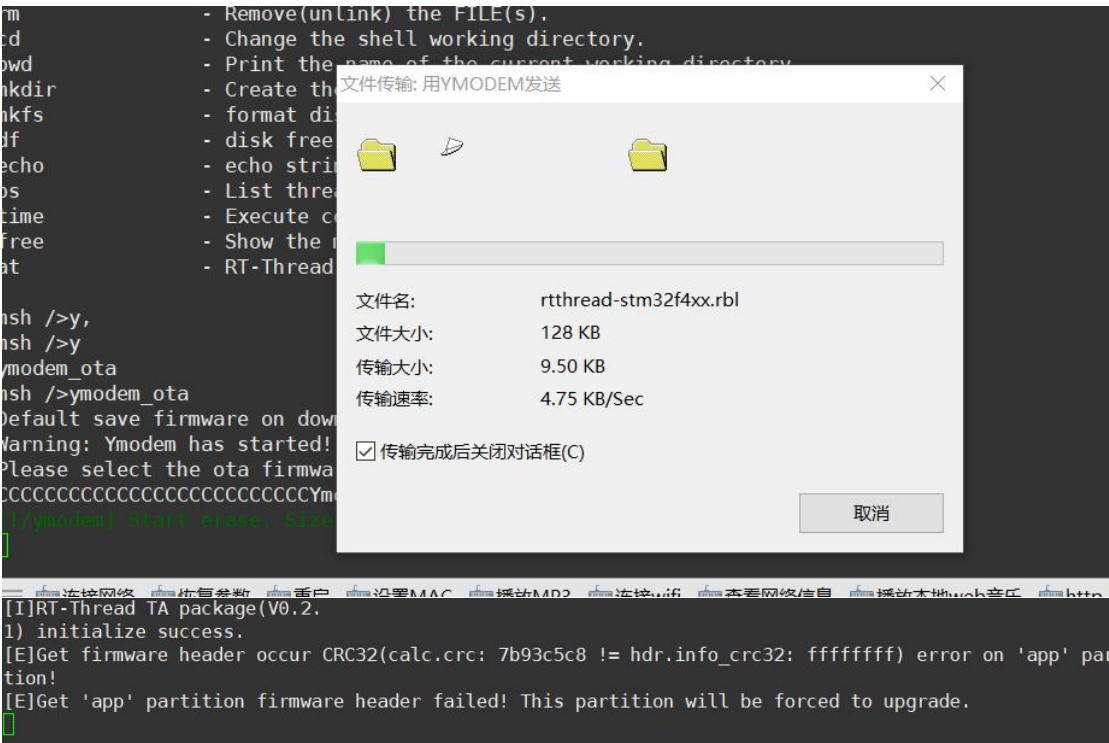










图 22

在板子的串口终端中的命令行操作，输入 `http_ota url` 指定 `rb1` 的地址 的命令, 就会进入下载个更新 `app` 固件，如下图 23 所示，



图 23

固件下载完成后，开始更新 `app` 固件，如下图 24 所示。

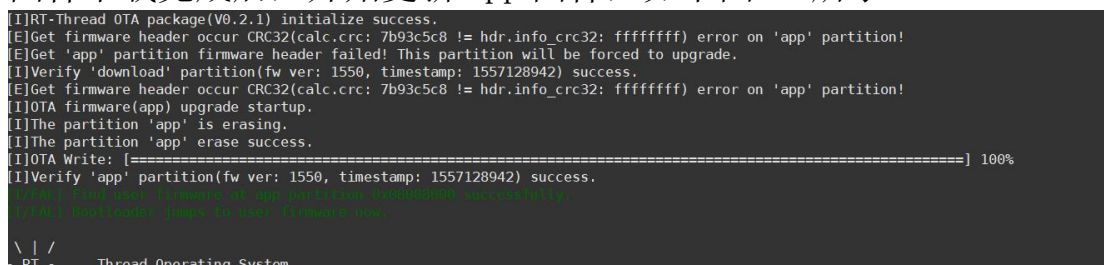


图 24

到此，让电子时钟跑起 `BootLoader` 和支持 `ota` 升级的功能就完成了。

下一步，计划在电子时钟上尝试功能扩张添加获取天气的功能显示。。。。