STM32G47x 双 Bank 模式下在线升级

关键字：双 Bank， 在线升级

1. 前言

STM32G47x 的 Flash 可以工作在双 bank 模式下，在该模式下对 FLASH 的操作支持 RWW(Read-While-Write)，在 Bank1 中可以对 Bank2 进行操作而不影响当前 Bank1 中的应用程序的运行，反之亦然。本文对双 Bank 模式下程序在线升级进行介绍，指出操作中的注意事项。

2. 双 Bank 工作原理

STM32G47x 系列 MCU 支持 Flash 双 Bank 功能，且芯片出厂默认配置即使能了双 Bank 功能。基于两个独立的 Bank，用户可以选择将应用程序放在任意一个 Bank 中运行，通过设置标志位 BFB2 来决定从哪一个 Bank 启动：

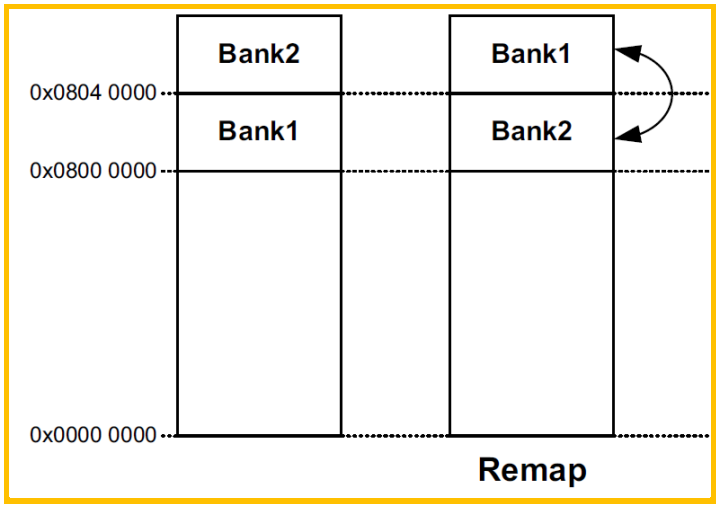
BFB2 = 0，MCU 双 Bank 启动禁能，从 Bank1 启动运行；

BFB2 = 1，MCU 双 Bank 启动使能，从 Bank2 启动运行，若是 Bank2 中无正常程序，则检测 Bank1 中是否有正常程序，若有则运行 Bank1 中的程序，若无则跳转到系统 Bootloader 运行(详见 AN2606 对应说明)。

FB_MODE 反映了 Bank 的地址映射，双 Bank 的地址映射方式如下：

FB_MODE= 0 时，Bank1 的起始地址为 0x08000000，Bank2 的起始地址为 0x08040000；

FB_MODE= 1 时，Bank1 的起始地址为 0x08040000，Bank2 的起始地址为 0x08000000；



在 STM32G47x 系列 MCU 中，MCU 启动时始终是从 0x08000000 这个地址开始运行。通过地址重映射的方式，将不同的 Bank 起始地址指定到 0x08000000，所以当从 Bank2 启动运行时，千万不要认为 MCU 是从 0x08040000 开始运行的，MCU 依旧是从 0x08000000 这个地址开始运行，认清这点对程序在线升级时对 Flash 的擦除与烧写特别重要。

另外基于以上地址重映射操作，在程序设计过程中，用户也不需要对地址空间做偏移处理，完全按照单 Bank 的思路进行设计即可，但程序不能超过单 Bank 大小。

3. 在线升级实现

ST 的软件库中给出了完整单 Bank MCU 的 IAP 升级方案，基于 UART 和 Y-Modem 协议。在双 Bank 模式下，数据传输层面与单 Bank 是完全一样的，所以本文不再对 IAP 的数据传输方式与协议进行描述，具体可以参考 AN4045 与 Y-Modem 协议内容。

G4 系列的开发包中目前还没有专门的 IAP 例程，直接从其他系列移植即可，比如 F3 或是 L4 系列。这些例程中存在两个工程，即 IAP 工程与应用程序工程。在 G47x 双 Bank 模式下，是在当前 Bank 中对另一 Bank 进行操作，且由于 Flash 支持 RWW，所以不需要再将 IAP 与应用程序分开，可以合并到一起。

以 L4 的 IAP 例程为基础，在原 IAP 例程的基础上，为了双 Bank，需要修改的点如下(本文只讨论程序下载，其他暂不讨论)：

- 将要进行擦除烧录的 Flash 的起始地址固定为：0x08040000。
- 添加 Bank 判断函数

```
/**
  * @brief  Gets the bank of a given address
  * @param  Addr: Address of the FLASH Memory
  * @retval The bank of a given address
  */
uint32_t GetBank(uint32_t Addr)
{
  uint32_t bank = 0;

  if (READ_BIT(SYSCFG->MEMRMP, SYSCFG_MEMRMP_FB_MODE) == 0)
  {
    /* No Bank swap */
    if (Addr < (FLASH_BASE + FLASH_BANK_SIZE))
    {
      bank = FLASH_BANK_1;
    }
    else
    {
      bank = FLASH_BANK_2;
    }
  }
  else
  {
    /* Bank swap */
    if (Addr < (FLASH_BASE + FLASH_BANK_SIZE))
    {
      bank = FLASH_BANK_2;
    }
    else
    {
      bank = FLASH_BANK_1;
    }
  }

  return bank;
} « end GetBank »
```

- 添加 Page 判断函数

```
/**
  * @brief  Gets the page of a given address
  * @param  Addr: Address of the FLASH Memory
  * @retval The page of a given address
  */
uint32_t GetPage(uint32_t Addr)
{
  uint32_t page = 0;

  if (Addr < (FLASH_BASE + FLASH_BANK_SIZE))
  {
    /* Bank 1 */
    page = (Addr - FLASH_BASE) / FLASH_PAGE_SIZE;
  }
  else
  {
    /* Bank 2 */
    page = (Addr - (FLASH_BASE + FLASH_BANK_SIZE)) / FLASH_PAGE_SIZE;
  }

  return page;
}
```

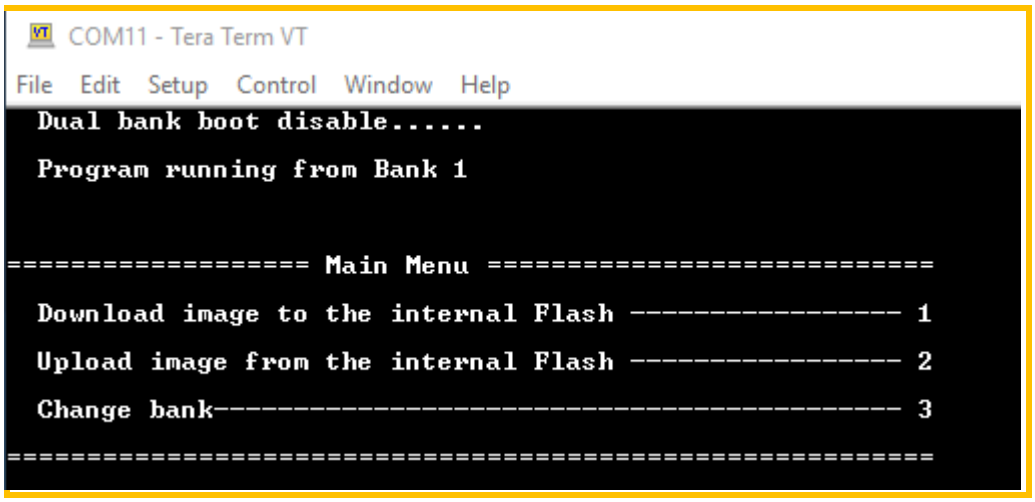
- 修改 FLASH_If_Erase(uint32_t start)函数如下：

```
/**
  * @brief  This function does an erase of all user flash area
  * @param  start: start of user flash area
  * @retval FLASHIF_OK : user flash area successfully erased
  *         FLASHIF_ERASEKO : error occurred
  */
uint32_t FLASH_If_Erase(uint32_t start)
{
  uint32_t NbrOfPages = 0;
  uint32_t PageError = 0;
    uint32_t FLASH_BANK = 0;
  FLASH_EraseInitTypeDef pEraseInit;
  HAL_StatusTypeDef status = HAL_OK;

  /* Unlock the Flash to enable the flash control register access *************/
  HAL_FLASH_Unlock();

  /* Get the sector where start the user flash area */
  NbrOfPages = (FLASH_USER_END_ADDR - start)/FLASH_PAGE_SIZE;


    FLASH_BANK = GetBank(start);


  pEraseInit.TypeErase = FLASH_TYPEERASE_PAGES;
  pEraseInit.Page = GetPage(start);
  pEraseInit.Banks= FLASH_BANK;
  pEraseInit.NbPages = NbrOfPages;
  status = HAL_FLASHEx_Erase(&pEraseInit, &PageError);

  /* Lock the Flash to disable the flash control register access (recommended
     to protect the FLASH memory against possible unwanted operation) *********/
  HAL_FLASH_Lock();

  if (status != HAL_OK)
  {
    /* Error occurred while page erase */
    return FLASHIF_ERASEKO;
  }

  return FLASHIF_OK;
} « end FLASH_If_Erase »
```

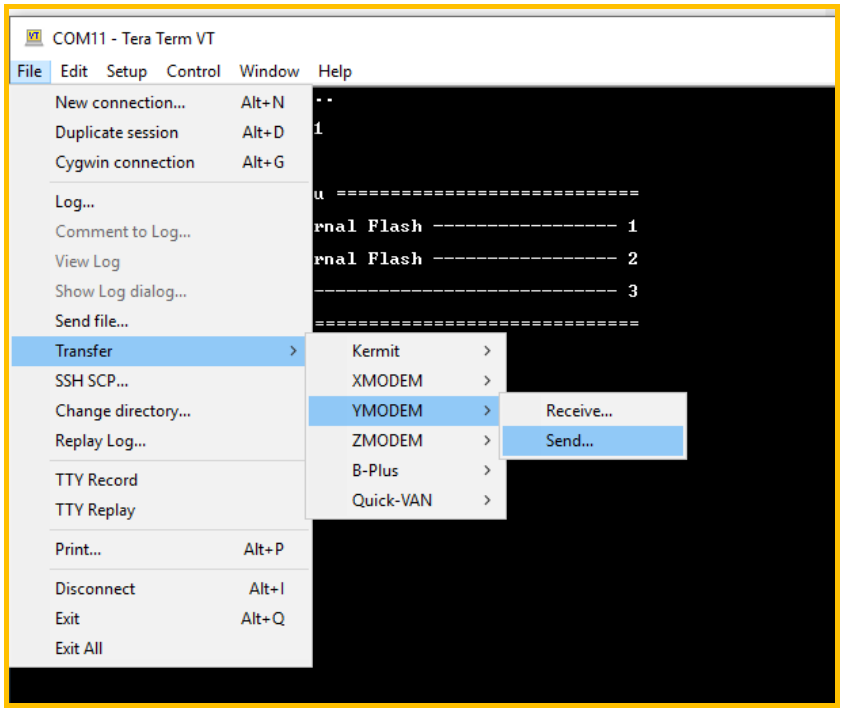
将修改后的代码编译生成 bin 文件，通过 CubeProgrammer 烧录到 0x08000000。

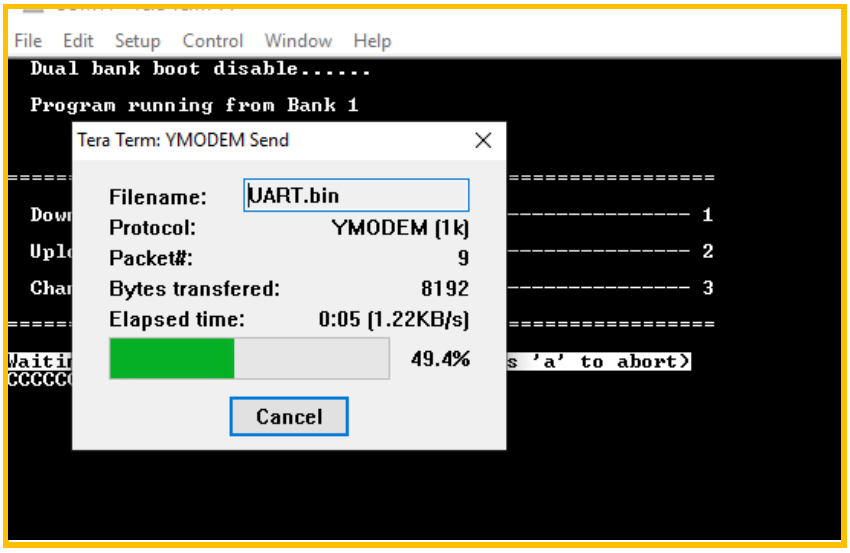
4. 结果验证

MCU 运行后的通过终端串口工具(Tera Term)打印调试信息，烧录完程序后，第一次上电的信息如下，BFB2 禁能，程序中 Bank1 中启动。在终端中输入 1 可以启动程序下载，输入 3 触发 Bank 切换，“2”忽略。



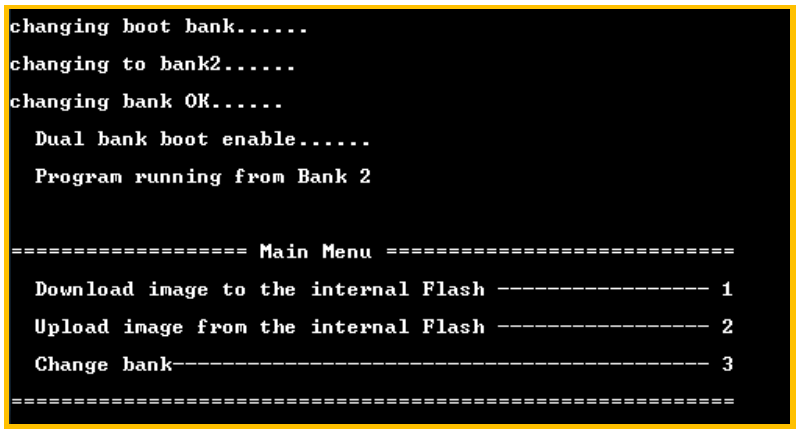
此时 MCU 将进入等待数据传输，并显示“Waiting for the file to be sent ... (press 'a' to abort)”，用户在 Tera Term 通过以下的菜单选择目标 bin 文件，并启动传输。





传输完成后，将显示烧录成功的信息。

此时可以输入 3 进行 Bank 切换，切换成功后将显示结果如下，切换到 Bank2 中运行。



5. 小结

本文简单介绍了双 Bank 的工作原理与双 Bank 进行在线 IAP 时应该注意的问题和需要添加的代码，并对最终运行的结果进行了验证。

参考文献

【如有， 请注明；否则，请注明：无】

文件编号	文件标题	版本号	发布日期
1	RM0440 Reference manual	REV6	

文档中所用到的工具及版本

Keil：version 5.33

Tera Term： version 4.94

版本历史

日期	版本	变更
2022 年 02 月 08 日	1.0	首版发布

重要通知 – 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其 各自所有者的财产。

本文档是 ST 中国本地团队的技术性文章，旨在交流与分享，并期望借此给予客户产品应用上足够的帮助或提醒。若文中内容存有局限或与 ST 官网资料不一致，请以实际应用验证结果和 ST 官网最新发布的内容为准。您拥有完全自主权是否采纳本文档（包括代码，电路图等）信息，我们也不承担因使用或采纳本文档内容而导致的任何风险。

本文档中的信息取代本文档所有早期版本中提供的信息。