AN4767
应用笔记

优化使用STM32微控制器的闪存双存储区结构 - STM32Cube的软件扩展

引言

双存储区功能是多款STM32微控制器的通用特性。本文档旨在描述如何在客户应用中使用此功能。

本应用笔记中涉及的主要内容是现场升级，通过代码示例（X-CUBE-DBFU）介绍。

尽管本文档仅直接描述了STM32L0系列Cat5设备和STM32L4系列的入门系列和USB OTG设备，但具有两个半独立存储器区的其他STM32MCU也可以共享部分所述属性并采用类似的使用方式。

下列文件可在*www.st.com*获取，可作为参考：

- 参考手册RM0367：“超低功耗STM32L0x3高级的基于ARM®的32位MCU”
- 参考手册RM0376：“超低功耗STM32L0x2高级的基于ARM®的32位MCU”
- 参考手册RM0377：“超低功耗STM32L0x1高级的基于ARM®的32位MCU”
- AN2606应用笔记：“STM32微控制器系统存储器启动模式”
- AN4024应用笔记：“STM32安全固件升级（SFU）”
- AN4657应用笔记：“使用UART进行STM32L0应用内编程”

目录

表格索引

图片索引

1 定义

表1. 缩略语列表

术语	说明
CPU	中央处理单元（MCU的一部分）
EEPROM	电可擦除可编程只读存储器
IAP	应用内编程
MCU	微控制器
NVIC	嵌套向量中断控制器
NVM	非易失性存储器（EEPROM或基于闪存）
RM	参考手册
UART	通用异步收发器
USART	通用同步和异步收发器
VTOR	向量表偏移寄存器

2 双存储区用例

在设计使用双存储区设备的应用程序时，可以选择多种方法来使用程序存储器的后半部分。

2.1 大代码

第一种情况是具有代码的应用程序，该代码在单个程序实例中需要使用两个存储器。

即使在这种情况下，双存储区方法也可能具备优势。还可以考虑在两个存储区分别加载具有不同功能集的两个程序，实现两种工作模式（其中，主/从、录音机/播放器、发送器/接收器）。

2.2 数据存储

第二存储区特别适用于数据存储（例如：数据记录）。

此时，双存储区可用于使能“同时读写”特性，从而避免在将数据编程到另一存储区时CPU失速。 此方法还可以更轻松、更快速地批量擦除存储在第二存储区的数据。

2.3 代码备份

如果不考虑固件升级场景（即设备不可访问），则第二存储区可用于代码备份。第二存储区可以存储相同的代码，并在闪存轻微损坏的情况下用作备份（例如通过辐射和加热）。

相对而言，通过后续代码完整性检查，响应未处理的异常，较易于执行存储区转换。

作为预防措施，可以定期检查两个副本的完整性，并且，在发生故障的情况下，可以用正确的副本重写有故障的副本。

2.4 双存储现场升级

本应用笔记主要讨论现场升级的目的。

除了AN4657所述的应用内编程标准功能之外，还存在多种优势。使用双存储区方法，所有代码均保存在一个项目中，从而得到单一的二进制码。如此一来，这消除对于那些在应用程序编程期间必须依赖加载程序的设备在有限功能的应用程序中需要加载程序的需要。

传统IAP的典型缺点在于：

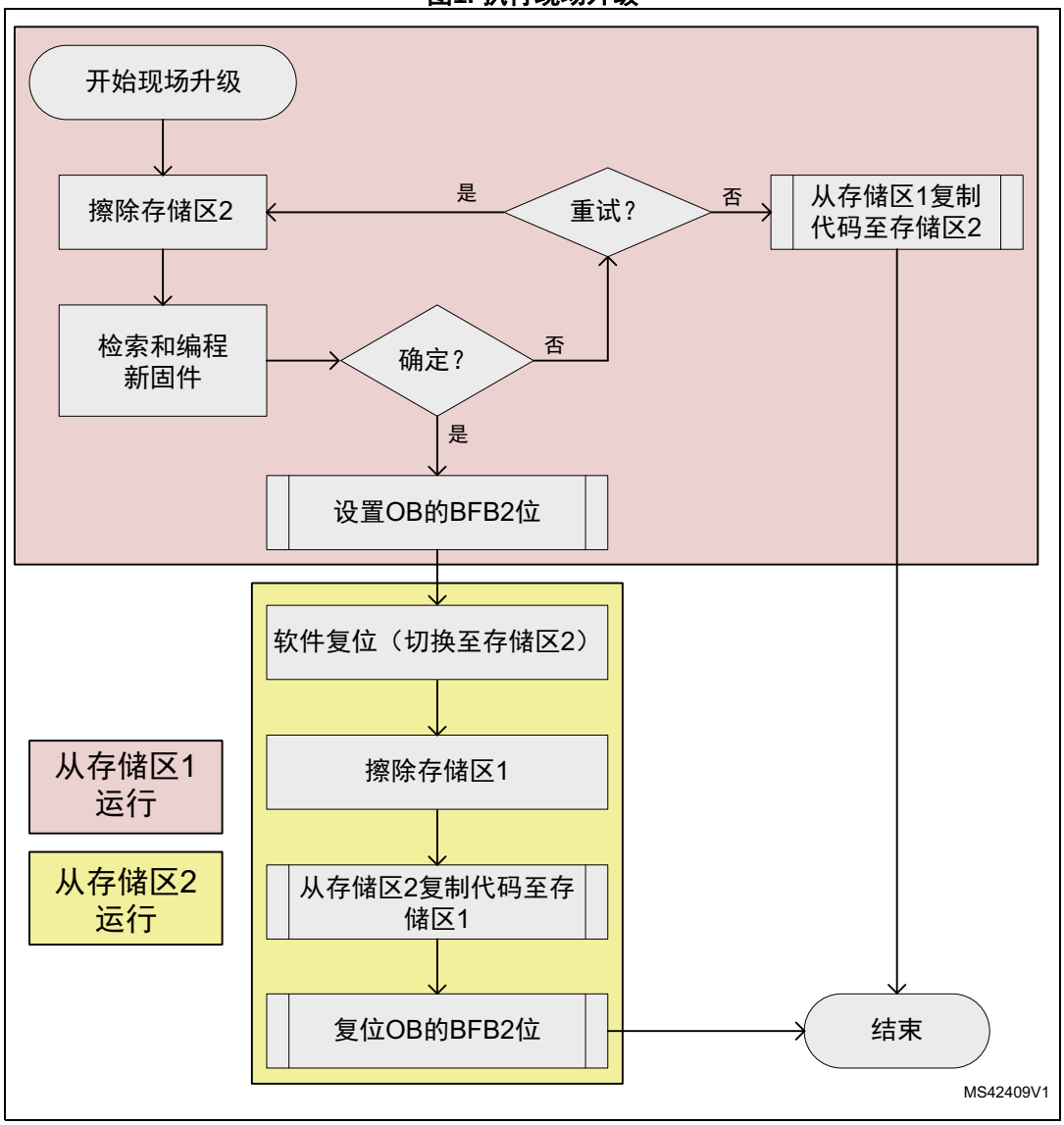
1. 加载程序无法执行所有后台任务
2. 加载程序无法升级或修复
3. 如果在升级过程中发生故障，设备可能保存在加载程序模式
4. 如果发生问题，无需回滚至之前的版本。

通过正确使用双存储区方法，可以应对和解决所有这些问题。

使用双存储区，所有与其他存储区的操作仅仅只是主程序的另一项任务。由于代码地址范围的内部重新映射，在正常操作期间，两个存储区的二进制码均可以保持相同。

执行现场升级的典型场景请参见 *图 1*。

图1. 执行现场升级



当存储区1中不存在代码时，务必保持BFB2标志设置，以确保在意外断电时也可保持安全。接下来，在复位之后，固件可使用多种方法检测到存储区1的代码必须替换，且程序从存储区2运行。务必执行此决定，即，只有一个二进制代码，且在每次启动时执行此代码。

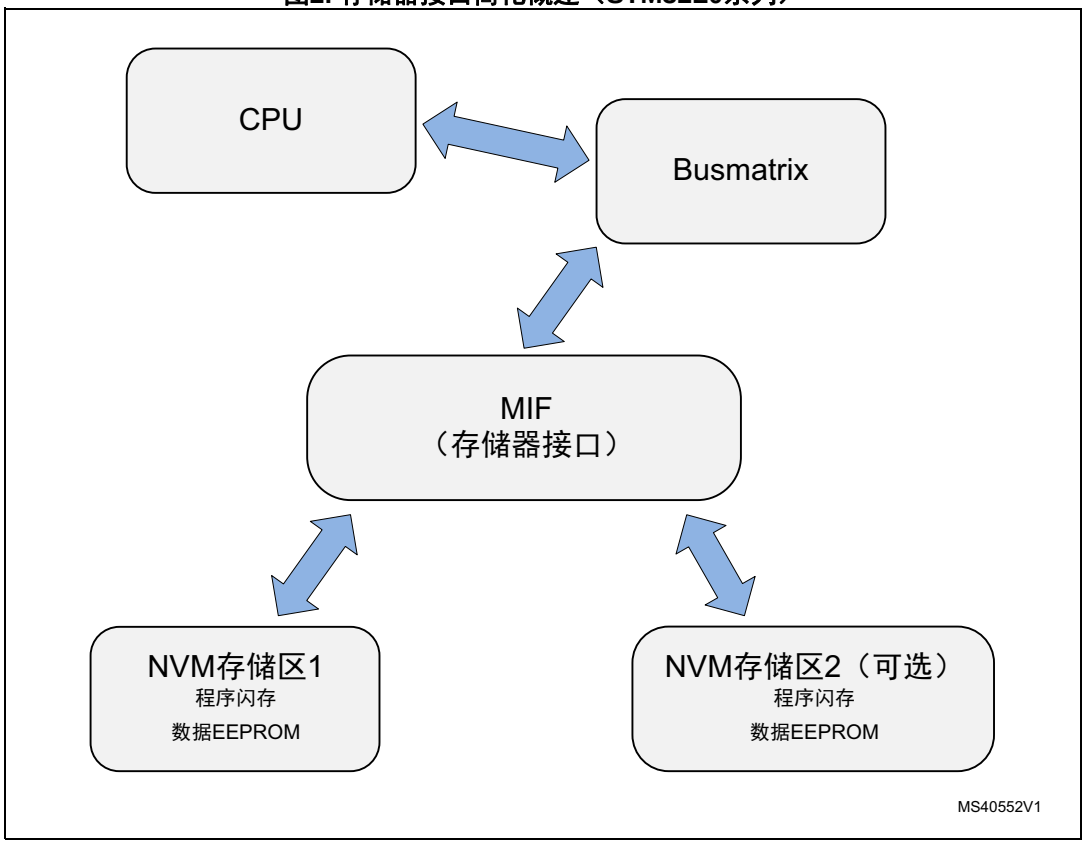
3 存储器实现摘要

STM32L0和STM32L4系列的NVM被分隔成多个区域：

- 用户代码存储器（程序闪存）
- 系统代码存储器
- 选项字节
- 用户数据存储器（数据EEPROM），仅适用于STM32L0系列

适用双存储区特性的存储器（即，在可使用双存储区功能的设备中找到的两个实例）是用户代码存储器和用户数据存储器。

图2. 存储器接口简化概述（STM32L0系列）



存储器接口能够并行读取两个存储区，或者在写入一个存储区时能够读取另一个存储区。但只在一个存储区内无法实现并行能力，因此，将数据存储在另一个存储区的应用程序具有更高的效率。

最有效的EEPROM存储解决方案（适用于STM32L0）是，存储区1的代码使用来自存储区2的数据，而存储区2的代码使用来自存储区1的数据EEPROM（当写入另一个存储区的程序存储器时也是如此）。

3.1 手动存储区选择

当设备启动至主闪存中的代码时，可以使用SYSCFG寄存器，以动态方式更改存储区的映射和别名。

通过翻转存储区Swap位（L0的SYSCFG_CFGR1寄存器中的UFB标志）或存储区模式（L4的SYSCFG_MEMRMP寄存器中的FB_MODE标志），可以将执行传递给另一个存储区。标志的值还表明了目前正在使用哪个存储区。

如果两个存储区的代码相同，则PC和堆栈值将保持有效，并且在发生映射更改后，可以继续执行程序。

如果两个存储区中的代码不相同（或者至少存储区转换点之前的扇区不相同），则执行很可能会崩溃。

另外，在开始转换之前，务必重置向量表（参见*第 3.3 节*）。

3.2 自动存储区选择

复位时，将评估选项字节中的BFB2标志，以及BOOT0引脚和BOOT1标志的状态。当通过BOOT0值选择启动至主闪存，且BFB2设置完毕时，将调用系统存储器自举程序来评估存储区2的内容。如果存储区2存储器的第一个字包含有效的堆栈指针地址，则系统自举程序假定存储区加载了有意义的代码，接下来，将存储区2另外命名为从0x0800 0000开始的地址范围，而不是存储区1。

注意使用具有双存储区自动选择特性的地址0x0000 0000。尽管BOOT0值指向主闪存选择，但该地址范围仍然是系统存储器的别名。

自动存储区选择过程不会带来任何副作用，仅仅只使启动时间增加几微秒。

3.3 向量表

无论闪存映射如何，为了保持中断处理的能力，必须重置向量表。在本应用笔记随附的示例中，向量表被复制到系统RAM，这是现场升级案例的最安全选项。此时，必须在RAM中分配其尺寸（L0系列高达192 B，L4系列高达288 B，有关任何给定产品的确切尺寸，请参见参考手册），并从原始位置中复制向量表。

接下来，必须将新的中断向量表地址写入系统控制块（SCB）寄存器VTOR。

NVIC并未始终意识到修改后的存储器地址别名，VTOR是引导至向量表存储器位置的最佳方法。默认情况下，链接器位于程序存储器的开始位置。当切换到存储区2时，必须更换VTOR，使其指向存储区2基址，反之亦然；当切换回存储区1时，必须将VTOR重置回存储区1基址。

请注意，VTOR复位值为零，如果BFB2选项处于活动状态，它将默认指向系统存储器。

另一种选择是在现场升级期间禁用中断，但遗憾的是，这种操作会否定双存储区系统的一些重大优势，例如缺乏任何“限制模式”。

4 示例项目

以下描述了示例代码，演示了固件升级用例。此示例与IAP应用程序密切相关，唯一的区别在于只有一个固件，且两个存储区都使用相同的固件，既是加载程序也是应用程序。

4.1 硬件设置

使用RS-232串行线，将USART2端口连接到PC。使用支持YMODEM协议的终端应用程序。Tera Term是Windows HyperTerminal的替代产品，其性能可靠且免费提供。已使用Tera Term版本4.84测试了示例功能。配置通信速度为115200 Bd，8个数据位，无奇偶校验，1个停止位，接着启动开发板。

4.2 加密选项

固件升级文件中包含的IP对于所有者而言可能是宝贵的信息，因此所有者可能会担心代码保密性问题，这一点在情理之中。结合完整性保护功能，即使是对称加密也可以防止引入伪造或欺诈性固件。

多个ST微控制器包括可选的AES硬件加速器外设选项。此时，可以使用STM32L486和STM32L083分别代替评估板的STM32L476和STM32L073，以便使用加密功能。

4.2.1 加密二进制码

此示例中使用的加密方案是在计数器（CTR）模式下使用明文AES。

这种方法的优点是不需要填充，只加密有效负载。虽然这与现场升级案例无关，但值得一提的是，计数器模式还能够避免传播通信错误（仅在通信过程中损坏的字节会在解密消息中被破坏）。

此外，无需开发专用加密实用程序，任何公开可用的加密库都能够生成计数器模式暂存器，并在暂存器和数据之间进行XOR。

如果用户不确定如何在计算机上准备输入，则必须遵循以下步骤：

1. 下载开源库 Crypto++® (www.cryptopp.com)
2. 安装GCC免费工具，由于不支持MinGW，最好是安装Cygwin。
3. 打开Cygwin.bat，使用 *make* 命令来构建Cryptopp库。
4. 使用 */cryptest.exe v*（v选项：验证套件）来测试库
5. 如果所有测试都通过，将显示消息“所有测试通过！”：
6. 生成二进制文件并将其放在Cryptopp库下
7. 使用参数调用cryptest.exe
   cryptest.exe ae <HexKey> <HexIV> Project.bin Firmware.aes
   项目中包含一个批文件，旨在消除对密钥格式的任何疑问。

cryptest.exe 将生成一个 *.aes* 文件，后者将用于具有AES外设的设备（STM32L083/STM32L486）。

4.2.2 配置解密

在项目中添加或取消注释*#define ENCRYPT*。这仅适用于具有AES外设的设备（在本例中为STM32L083/STM32L486）。

将占位符密钥和初始化向量替换为 *main.c* 中的任何其他值，即可定制此项目。

4.2.3 简单解决方案限制

本示例中使用的基本对称加密解决方案存在一些局限性。

它对现场的所有设备使用相同的密钥。单密钥的广泛使用会导致加密系统更易于受到攻击。它不仅让攻击者有更多机会获取密钥，而且一旦密钥泄露，所有设备都处于公开状态。使用代码的某些属性（例如版本）导出密钥，可以在一定程度上减少此影响。

此外，身份验证仅依赖于知道密钥的这一事实，这并不能完全减少提供非正版代码的可能性。

事实上，代码的前几个字节（由初始堆栈指针和向量表组成）大部分是可预测的（“已知的明文攻击”），这可能导致会削弱加密属性。在加密之前向“明文”添加随机序列（“salt”），在解密后忽略“salt”，即可轻松解决这一弱点。

4.2.4 其他加密选项

保护代码不仅涉及加密算法，而且涉及复杂的加密系统，包括在所有产品生命阶段（从设计和开发，到制造和维修，再到停用和回收）实施的安全要求。

还必须认真考虑身份验证角色和完整性保护。

意法半导体能够提供顶级嵌入式安全解决方案。

有关详细信息，请参见AN4024或联系最近的ST销售办事处。

4.3 示例操作

采用终端软件提供的用户界面易于操作。示例固件首先显示标题和菜单。按PC键盘上的数字即可选择选项。

1. 将文件下载到另一个存储区

   选择第一个菜单选项，即表示开始将二进制文件下载到另一个存储区中。擦除和重写另一个存储区先前的内容。文件大小受存储器大小的限制。选择要下载的文件，解密并写入该文件。使用YMODEM协议继续进行通信。

2. 擦除另一个存储区的内容

   此选项使其他存储区的内容无效。

3. 重写另一个存储区的内容

   来自活动存储区的代码被复制到相对的存储器存储区。

4. 检查另一个存储区的完整性

   此选项启动对其他存储区内容的检查。

   对于L0系列，示例固件将代码CRC完整性值存储在EEPROM数据存储器中。对于L4设备，检查内容仅限于简单存在性检查。

5. 切换存储区

   如果在另一个存储区似乎存在功能代码，则重写向量表并切换存储区（如 *第 3.1 节：手动存储区选择*所述）。

6. 切换系统存储区选择

   对选项字节进行编程，以修改BFB2位的值（从存储区2启动，如*第 3.2 节：自动存储区选择*所述）。

5 结论

如果使用得当，双存储区特性能够提供适用于广泛应用的众多优点。

用户可以通过选择具有双存储区存储器的设备来发挥这些优点，这可能需要使用成本更高且封装更大的产品。

6 版本历史

表2. 版本历史

日期	版本	变化说明
2016年9月29日	1	初始版本。
2016年10月31日	2	更新了文档标题。

表3. 中文版本历史

日期	版本	变化说明
2018年11月21日	1	中文初始版本。

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、更正、增强、修改和改进的权利，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。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。