Freescale Semiconductor
应用笔记

Document Number: AN5112
Rev 0, 4/2015

使用 Kinetis Flash 只执行访问控制功能

内容

1 简介

部分新的 Kinetis 系列芯片的 Flash 存储器增加了访问控制功能。Flash 访问控制(FAC)是一个可配置的存储器保护方案，允许用户在使用软件库的同时，可以通过编程对该库的访问进行限制。这样飞思卡尔或第三方供应商便可将软件库预编程至芯片中，然后将该芯片发布给可使用此软件库的终端客户。软件存储在芯片内，但终端客户无法从芯片读取代码。

FAC 可用来将片上 Flash 存储器段标记为只执行和/或管理员/特权访问。本文讨论如何在 FLASH 上使用 FAC 功能来创建只执行区域、实现时的一些限制以及如何规避这些限制。

2 Flash 访问控制的工作原理

Flash 访问控制功能为 Flash 模块添加了新的寄存器。8 个 8 位 XACC 寄存器定义了哪些程序 Flash 段为只执行访问，另有 8 个 8 位 SACC 寄存器定义哪些程序 Flash 段为仅特权访问。寄存器中的每一位对应可保护的一个存储器段。默认情况下，所有 XACC 和 SACC 寄存器均为 0xFF。当 XACC 或 SACC 中的位清零后，将对应的 Flash 段标记为只执行访问或仅特权访问。

芯片的 Flash 存储器大小决定了访问受控的段的数量。对于程序 Flash 为不到 128 KB 的芯片，被分为 32 个相同大小的段。对于程序 Flash 大于 128 KB 的芯片，被分为 64 个相同大小的段。可以通过读取 FTFA_FACSS 和 FTFA_FACSN 来确定特定芯片上实现的段的数量和大小。

具有 32 个段的芯片仅使用 FTFA_XACCL[3:0]寄存器（FTFA_XACCHn 的值此时被忽略）。具有 64 个段的芯片使用全部 8 个寄存器—FTFA_XACCH[3:0]和 FTFA_XACCL[3:0]。后文中，8 个 XACC 寄存器均称为 FTFA_XACCn。

Flash 存储器控制器 (FMC) 针对片上 Flash 存储器的每次传输的访问权执行逐周期评估。在每次尝试进行 Flash 传输的寻址阶段，FMC 会检查特权访问寄存器（FTFA_SACCn）和执行访问（FTFA_XACCn）寄存器以决定允许还是拒绝访问。如果对只执行区域进行数据访问，则会以总线错误的方式中止访问。同时，此时读取到的数据为零。

3 将段标记为只执行

FTFA_XACCn 寄存器是只读的。寄存器从程序 Flash (P-Flash) IFR 空间的对应位置获取其数值。XACCA 和 XACCB 位于 P-Flash IFR 中，各占 64bit。复位时，从 IFR 中读取 XACCA 和 XACCB，且 XACCA 和 XACCB 的逻辑 AND 用来加载 FTFA_XACCn 寄存器。Program Once Flash 命令可用来向 P-Flash IFR 中写入 XACCA 和 XACCB 值。有关 XACCA 和 XACCB 的 Program Once 索引，请参考芯片参考手册。

由于 XACCA 和 XACCB 经过 AND 计算，即使在 XACCA 已被编程后，XACCB 仍然可用来为其他段增加保护。段的访问控制可从数据读取和执行（XAn =1）更改为只执行（XAn =0）。两组 XACC IFR 支持两个级别的供应商，可将他们的专用软件库添加至芯片中。

要将某个段标记为只执行，请执行以下步骤:

1. 编程并验证将被标记为只执行区域的库代码。
2. 利用 Program Once Flash 命令将 XACCA 值写入 P-Flash IFR，以便标记库代码所在的段为只执行。
3. 如果还需要保护第二个库，将芯片给到第二个库的所有者，以便编程并验证第二个库。
4. 用 Program Once Flash 命令将 XACCB 值写入 P-Flash IFR，以便标记第二个库所在的段为只执行。
5. 如果没有第二个库，则使用与 XACCA 相同的值对 XACCB 编程。

注

将段标记为只执行时，必须在配置 XACCA 或 XACCB Program Once 值前编程和验证代码。如果首先对 XACCA 或 XACCB 编程，则下次复位时，对应的段变为只执行，此时若不执行 Erase All Blocks 命令以解锁只执行区域，则无法对该区域编程。

下图示例中，XACCA 和 XACCB 用于保护位于 512 KB Flash 芯片中的 2 个代码库。

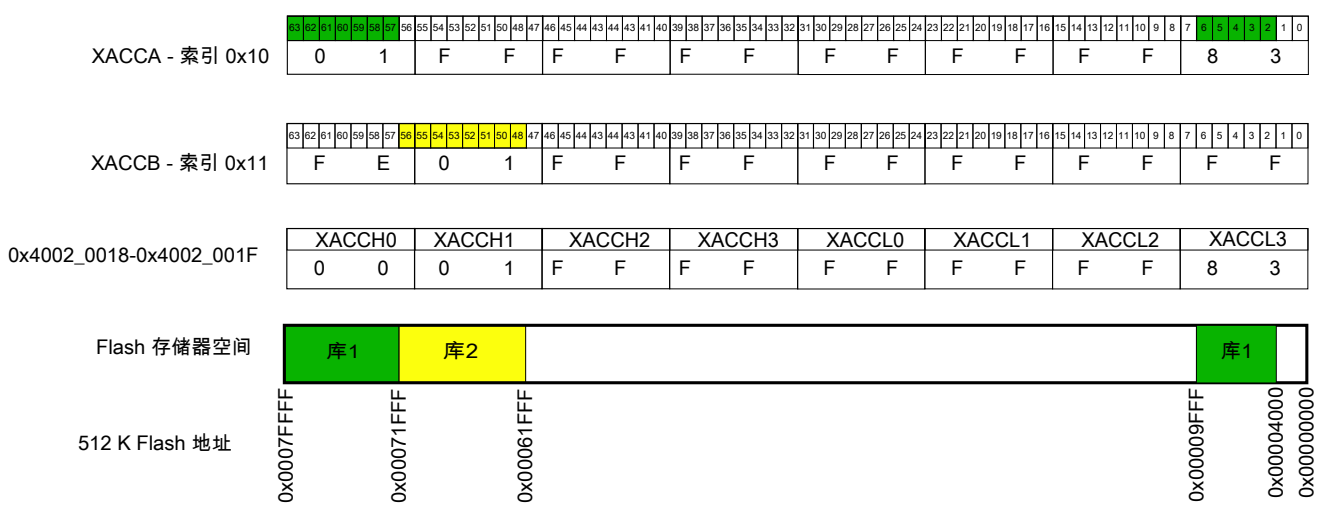

图 1. XACCx 保护示例

4 将区域标记为只执行的其他影响

将某个区域标记为只执行的主要影响有：除了装载相对于 PC 寻址的常数，只执行区域无法通过内核、调试接口或任何其他芯片上主机读取数据。将区域标记为只执行对 FMC 缓存性能有影响，还会影响部分 Flash 命令操作。

4.1 FMC 缓存

一般而言，FMC 缓存的内容在 FMC 寄存器空间中表现为一系列 FMC 标签和数据寄存器（不同芯片的具体的寄存器数量不同）。如果任何段被标记为只执行，则 FMC 缓存对用户隐藏。标签寄存器变为只读且无法写入，并且数据寄存器无法读写。针对标签和数据阵列的写操作将会被忽略，针对数据阵列的读操作返回零。

4.2 Flash 命令影响

FAC 功能与编程至芯片中的软件库一同使用，且在代码标记为只执行之后不再更改。由于这个原因，在 Flash 的只执行区域中作为预编程到芯片的库发布的任何代码都应在发布前进行严格的测试。

如果只执行区域中的代码需要更新，这在实际应用中是有可能的，需要考虑只执行区域中的 Flash 编程和擦除操作存在限制。

4.2.1 编程命令（编程 longword/phrase/section/check）

如果目标 Flash 位于只执行保护字段内，除非完成 Read 1s All Blocks 或 Erase All Blocks 命令，并返回成功标志（表示该部分完全擦除），编程命令是被禁止的。完成 Read 1s All Blocks 或 Erase All Blocks 命令并返回成功标志后，只执行字段对编程命令开放，包括编程校验命令。这表示，完成某个库的编程并标记为只执行后，更新该只执行空间的唯一途径便是擦除整个芯片。如果试图去对不接受编程命令的保护段进行编程或者编程校验，则会置位 Flash 保护冲突错误标志(FTFA_FSTAT[FPVIOL])。

为了在更新之后重新锁定只执行字段，可复位芯片或再次执行 Read 1s All Blocks 命令。此时，预期 Read 1s All Blocks 会失败（芯片未完全擦除）。这会关闭只执行字段，使其无法编程。

4.2.2 擦除命令（擦除区块/扇区）

如果选定的 Flash 区块或扇区包含只执行保护字段，则不允许 Erase Block 和 Erase Sector 命令，除非完成 Read 1s All Blocks 或 Erase All Blocks 命令，并返回成功标志（表示该部分完全擦除）。完成 Read 1s All Blocks 命令并返回成功标志后，只执行字段便对 Erase Block 和 Erase Sector 命令开放。这意味着不可使用区块和扇区擦除命令来仅仅擦除只执行字段。必须擦除整块芯片，才能修改只执行字段。整块芯片擦除后，可重新编程只执行字段。只执行区域不仅对编程命令开放，还对擦除命令开放。然而，必须擦除芯片才能实现这一点，因而在大多数情况下，如果需要修改只执行字段，则 Erase Block 和 Erase Sector 不起作用。如果试图去对不接受擦除命令的保护段进行块擦除或者扇区擦除，则会置位 Flash 保护冲突错误标志（FTFA_FSTAT[FPVIOL]）。

禁止只读空间的擦除命令和禁止只读空间的编程命令的过程是一样的。若要针对擦除关闭只执行空间，芯片必须复位，或者完成 Read 1s All Blocks 命令且结果为失败（芯片不再完全擦除）。

警告

Erase All Blocks 的命令，始终是允许的，即使标记为只执行的字段也可擦除。终端用户代码应当始终不可在有一个或多个预编程库的芯片上使用 Erase All Blocks 命令。如果使用了 Erase All Blocks 命令，则预编程库代码将会丢失。

5 不可将只执行段改回允许数据和代码访问的原因

段标记为只执行后，无法将其改回允许数据和代码访问。用来载入 XACCn 寄存器的 P-Flash IFR 只能编程一次。XACCA 和 XACCB 编程后，无法修改。没有可用来擦除它们的 Flash 命令，包括 Erase All Blocks 命令或批量擦除。XACCA 和 XACCB 编程后，无法重新编程以增加更多只执行段。尝试使用 Program Once 命令写入已有非 FFFF 值的 IFR 将会返回访问错误（FTFA_FSTAT[ACCERR] = 1）。

6 Flash 访问控制的特殊考虑因素和限制

由于 ARM® Cortex® M4 和 M0+内核无法识别 Flash 访问控制，因此应当考虑和此功能相关的一些重要的限制。为使用 Flash 访问控制的系统设计的软件需要考虑一些特殊因素，以确保正确操作，并保护只执行软件。

6.1 调试

FAC 功能最大的限制是调试接口并非针对该功能而设计。标记为只执行的 Flash 段不可由调试器直接读取。然而，即使代码并非直接可读，调试器依然可用作后门，对代码进行逆向工程。

为了确保只执行代码完全安全且不会被重新构建，飞思卡尔建议:

- 启用 Flash 安全功能，以禁用针对芯片的调试访问（FTFA_FSEC[SEC]）。
- 可选择使用 FTFA_FPROT*n* 以明确保护只执行代码区域。
- 可选择禁用批量擦除功能，以防止通过调试器意外擦除存储在只执行段内的代码(FTFA_FSEC[MEEN])。

6.1.1 Flash 安全和保护功能如何与 FAC 交互

如“Flash 命令影响”章节所述，只执行区域可防止意外擦除和修改。然而，如果终端客户使用 Erase All Blocks 命令，那么只执行区域内的代码依然可能丢失。如果 Flash 的任何区域由 FTFA_FPROT*n* 保护，则 Erase All Blocks 命令将会终止。利用 FTFA_FPROT*n* 可将只执行区域标记为保护，使其更不容易被意外擦除或修改。为了重新编程只执行区域，需要在禁用保护的同时擦除并重新编程含有 FTFA_FPROT*n* 设置的 Flash 配置字段，然后必须复位芯片，使新的保护设置生效。芯片在无区域保护的情况下启动后，Erase All Blocks 命令可用来擦除整个芯片，包括只执行区域。

注

FPROT 位保护的区域大小以及只执行段的大小并非始终相等。

注

飞思卡尔建议不要使用 FTFA_FPROT*n* 保护 Flash 的第一个区域。如果第一个区域被保护，则之后修改保护方案的唯一途径是使用调试器的批量擦除。

调试器运行的批量擦除命令不受 FTFA_FPROT*n* 设置影响。启用安全性时，FTFA_FSEC[MEEN]字段可用来禁用调试器批量擦除。启用安全性并禁用批量擦除可用来防止调试器意外擦除芯片。要通过调试器擦除或编程 MCU，处理器需通过后门密钥访问解除安全限制。

所有这些控制位均位于 Flash 中的位置 0x0400 - 0x040F。如果通过调试端口或启动加载程序访问，可实现扇区擦除和重新编程以设置保护区域。有关更多信息，请参阅使用 *Kinetis* 安全和保护功能（文档 AN4507）。

6.2 只执行代码对其他只执行段可见

ARM 指令集架构高度依赖相对于 PC 的读取操作。必须允许这些看似普通数据读取的操作访问只执行空间。Flash 访问控制逻辑分析所有 Flash 存储器访问，且只允许来自只执行区域的取指以及相对于 PC 的读取操作对只执行区域进行访问。逻辑系统同等对待所有只执行区域；因此，标记为只执行的任何段都可使用 PC 相关访问来访问任何其他只执行段。

对于仅将一组软件库编程到芯片的单次预编程而言，必须编程两级访问控制，以保护预编程代码。这可以通过将与 XACCB 相同的值写入 XACCA 来完成。这样可以防止最终用户将新代码编程到芯片、将其标记为只执行，并使用该代码读取最初的只执行区域的代码。

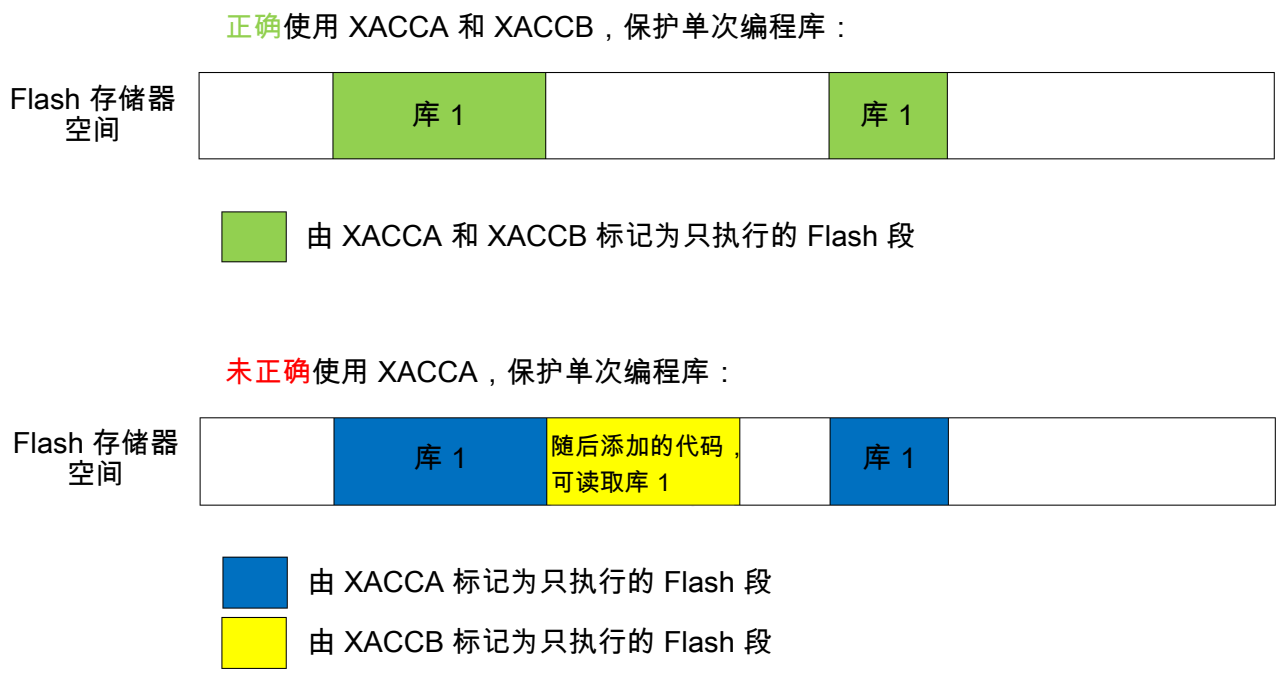

图 2. 保护单个库

如果要使用 2 个预编程库，则编程第二个库时，可看到第一个库。例如，公司 A 将一个库编程到芯片，并对 XACCA 编程将段标记为只执行，然后将此芯片送至公司 B。公司 B 向芯片编程第二个软件库，同时写入 XACCB 将第二个库的段标记为只执行。公司 B 必须是一个受信任的实体，因为公司 B 可以访问公司 A 的软件库。公司之间应当已经建立合作关系，以决定由谁来保护芯片，并提供系统级功能。合作时，必须签署 NDA 和法律协议，以保护软件库的利益。

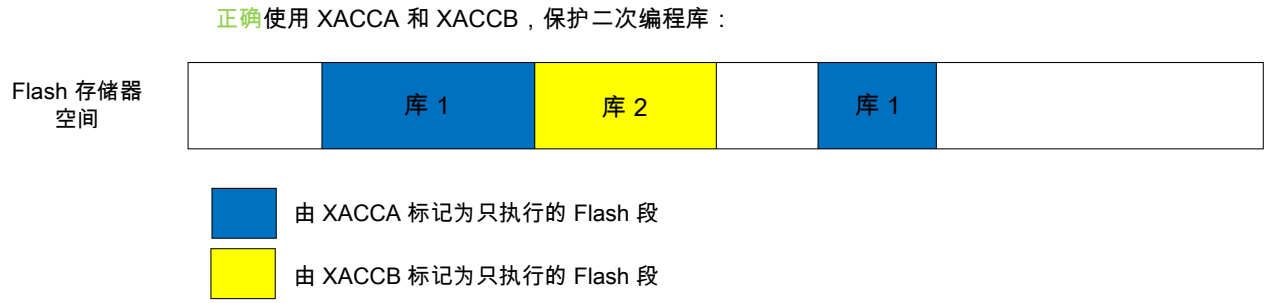

图 3. 保护两个库

6.3 ARM Cortex-M4 内核上的只执行代码入口

检测 ARM Cortex-M4 内核上的相对于 PC 的读取操作并没有确定的方法。为了保护只执行空间，非只执行空间与只执行空间之间的转换有延迟，随后状态信息才会显示内核处于只执行空间。在延迟期间，不允许从只执行空间进行相对 PC 的读取操作。如果此时从只执行段进行相对 PC 的读取操作，则会触发访问错误（如同从代码和数据 Flash 区域读取只执行段）。

为了避免不必要的访问错误，应当控制只执行代码的入口点：

- 可从非仅执行代码中调用的只执行函数不应在前 6 个指令中包含相对于 PC 的读取操作。实现该目标的最简单方法是在只执行函数起点加入 6 个 NOP 指令。必须确保编译器不会优化掉这些 NOP 指令。
- 仅从其他只执行函数（未暴露于库 API 的函数）调用的只执行函数不需要特殊的入口序列。
- 中断产生之后，应该跳转到一段特殊的只执行代码。如果对所有中断使用同一个默认处理程序，则默认处理程序应以 6 个 NOP 指令起始（或其他不从只执行空间执行相对于 PC 的读取的指令）。安全进入只执行空间后，处理程序可跳转至相关中断的特定处理代码。当特定中断服务程序返回默认处理程序时，应执行另外 6 个 NOP（或其他非相对于 PC 的读取指令）。此时，默认处理程序满足只执行空间的入口要求，因而可正常返回。这样，如果中断代码位于只执行空间内，中断服务程序应遵循与正常只执行函数入口点相同的入口点规则。

注

ARM Cortex-M0+架构允许检测相对于 PC 的载入的确定方法。对于带有 ARM Cortex-M0+内核和 Flash 访问控制功能的 Kinetis 芯片，不需要控制只执行代码的入口点，因此本节信息不适用于 Cortex-M0+处理器。

6.4 ARM Cortex-M4 芯片上的向量表

在 ARM Cortex-M4 内核的芯片上，如果默认处理程序用于所有中断向量，则 CPU 的向量表可保留在 Flash 存储器中，且运行时内无需修改以添加客户中断。另外，也可以使用独立的处理程序跳转表，以便跟踪默认处理程序需要调用的特定中断服务程序。实现方式可以不同，但大多数情况下，Flash 中会存储跳转表的默认版本。启动时，该表可复制到 RAM，以便在应用程序运行时可注册其他中断服务程序。

注

为了让内核从向量表中获取向量，Flash 的第一个段不应标记为只执行。

6.5 Flash 编程

如果禁用了调试器访问，则终端客户无法使用 EzPort 或调试器来将自己的代码编程至芯片上。相反，需要提供 Flash 编程的功能并将其预编程至芯片中（Flash 编程功能也可位于只执行区域内）。可使用 Kinetis 启动加载程序，让客户将其自己的代码编程至芯片中。有关 Kinetis 启动加载程序的更多信息，请参阅 www.freescale.com/kboot 。

Flash 编程/启动加载程序代码应配置成默认执行(向量表中的初始 PC 值应指向 Flash 编程/启动加载程序例程)。这在终端客户将自己的代码编程至从未使用过的新芯片时需要用到。建议将 Flash 编程/启动加载程序代码保留在芯片上(不让终端客户覆盖 Flash 编程软件)。这意味着启动加载程序代码默认运行。启动加载程序可以使用超时、对引脚进行采样，或者使用其他机制确定何时尝试跳转至用户代码。为了让启动加载程序跳转至用户代码，在终端用户的应用中，启动引导程序里面需要配置一个事先定义好的用户代码的地址。

7 参考资料

有关更多信息，请参阅以下文档:

- 您专用的 Kinetis 系列芯片参考手册：www.freescale.com/kinetis
- Kinetis 启动加载程序：www.freescale.com/kboot
- 使用 *Kinetis* 安全和保护功能（文档 AN4507）

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN5112
Revision 0, 4/2015