MCU 闪存加载器参考手册

文档编号：MCUFLSHLDRRM
第 4 版，2018 年 8 月

目录

章节编号 标题 页码

第 1 章
简介

1.1 简介 7
1.2 术语 7
1.3 功能框图 8
1.4 支持的功能 8
1.5 支持的组件 9

第 2 章
MCU 闪存加载器协议

2.1 简介 11
2.2 无数据阶段的命令 11
2.3 带传入数据阶段的命令 12
2.4 带传出数据阶段的命令 13

第 3 章
闪存加载器数据包类型

3.1 简介 15
3.2 Ping 数据包 15
3.3 Ping 响应数据包 16
3.4 成帧数据包 17
3.5 CRC16 算法 18
3.6 命令数据包 19
3.7 响应数据包 21

第 4 章
MCU 闪存加载器命令 API

4.1 简介 23
4.2 GetProperty 命令 23

4.3 SetProperty 命令 25
4.4 FlashEraseAll 命令 27
4.5 FlashEraseRegion 命令 28
4.6 ReadMemory 命令 29
4.7 WriteMemory 命令 31
4.8 FillMemory 命令 33
4.9 Execute 命令 35
4.10 Call 命令 36
4.11 Reset 命令 37
4.12 FlashProgramOnce/eFuseProgramOnce 命令 38
4.13 FlashReadOnce/eFuseReadOnce 命令 39
4.14 Configure Memory 命令 41
4.15 ReceiveSBFile 命令 41

第 5 章
支持的外设

5.1 简介 43
5.2 UART 外设 43
5.3 USB HID 外设 45
5.3.1 设备描述符 45
5.3.2 端点 46
5.3.3 HID 报告 46

第 6 章
外部存储器支持

6.1 简介 49
6.2 通过 FlexSPI 支持的串行 NOR 闪存 49
6.2.1 FlexSPI NOR 配置块 50
6.2.2 FlexSPI NOR 配置选项块 54
6.2.2.1 FlexSPI NOR 配置块的典型用例 56
6.2.2.2 使用 FlexSPI NOR 配置选项块对串行 NOR 闪存设备进行编程 56

6.3 通过 FlexSPI 支持的串行 NAND 闪存 57
6.3.1 FlexSPI NAND 固件配置块(FCB) 57
6.3.2 FlexSPI NAND 配置块 58
6.3.3 FlexSPI NAND FCB 选项块 60
6.3.4 FlexSPI NAND 配置选项块 61
6.3.5 闪存加载器的用法示例 62
6.4 通过 uSDHC 支持的 SD/eMMC 63
6.4.1 SD 配置块 63
6.4.2 闪存加载器的用法示例 65
6.4.3 eMMC 配置块 66
6.4.4 闪存加载器的用法示例 69

第 7 章
安全实用程序

7.1 简介 71
7.2 映像加密和烧写 71
7.2.1 生成加密映像并编程到闪存的示例 72
7.3 KeyBlob 生成和烧写 73
7.3.1 KeyBlob 73
7.3.2 KeyBlob 选项块 74
7.3.3 生成和烧写 KeyBlob 的示例 75

第 8 章
附录 A：状态和错误码

第 9 章
附录 B：GetProperty 和 SetProperty 命令

第 10 章
修订记录

10.1修订记录 85

第 1 章
简介

1.1 简介

MCU 闪存加载器是一个可配置的闪存烧写实用程序，可通过 MCU 上的串行通讯进行操作。它可以在整个产品生命周期（包括应用程序开发和最终产品制造等）中对 MCU 进行快速轻松编程。MCU 闪存加载器将以高度可配置的二进制或完整源代码形式提供。主机端命令行和 GUI 工具可用于与闪存加载器进行通信。用户可以利用主机工具通过闪存加载器上传和/或下载应用程序代码。

1.2 术语

目标机

运行引导加载程序固件(ROM)的设备。

主机

向目标机发送执行命令的设备。

源

通信序列的发起方。例如，命令或数据包的发送方。

目的地

命令或数据包的接收方。

传入

从主机到目标机。

传出

从目标机到主机。

1.3 功能框图

该功能框图描述了 MCU 闪存加载器的整体结构。

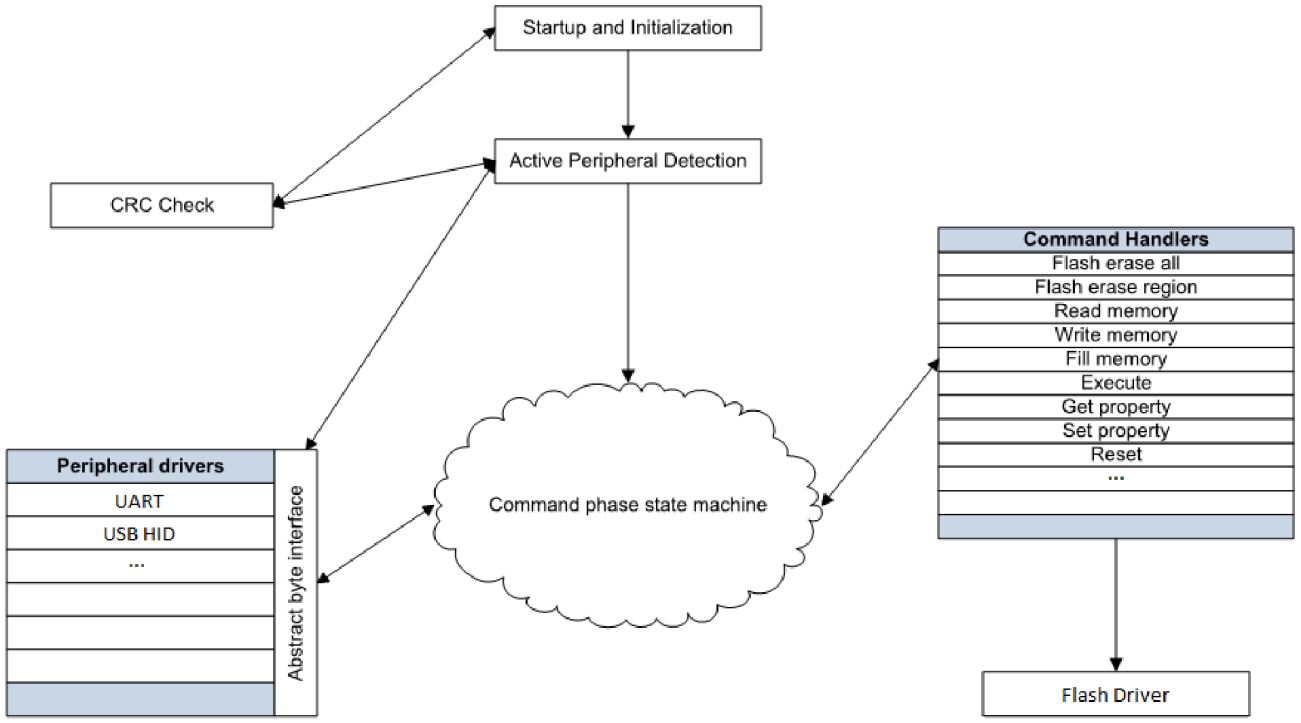

图 1-1. 功能框图

1.4 支持的功能

以下是 MCU 闪存加载器支持的一些功能：

- 支持 UART 和 USB 外设接口。
- 自动检测有效的外设。
- 能够禁用任何外设。

- UART 外设可实现自动波特率。
- 基于数据包的通用协议，适用于所有外设。
- 数据包错误检测和重传。
- 保护闪存加载器在运行时使用的 RAM。
- 提供读取设备属性（如 RAM 大小）的命令。
- 支持串行 QuadSPI 和其他外部存储器。
- 支持加密映像下载。

1.5 支持的组件

适用于闪存加载器固件的组件：

- 启动代码（时钟和 pinmux 等）
- 命令阶段状态机
- 命令处理程序
    - GenericResponse
    - FlashEraseAll
    - FlashEraseRegion
    - ReadMemory
    - ReadMemoryResponse
    - WriteMemory
    - FillMemory
    - GetProperty
    - GetPropertyResponse
    - Execute
    - Call
    - Reset
    - SetProperty
    - FlashProgramOnce/EfuseProgramOnce
    - FlashReadOnce/EfuseReadOnce
    - FlashReadOnceResponse
    - ConfigureMemory
- SB 文件状态机
    - 加密映像支持(AES-128)
- 数据包接口
    - 成帧分包器
    - 命令/数据包处理器
- 存储器接口
    - 抽象层接口

- FlexSPI NOR 存储器接口
- FlexSPI NAND 存储器接口
- SEMC NOR 存储器接口
- SEMC NAND 存储器接口
- SD 卡存储器接口
- eMMC 存储器接口

- 外设驱动程序
  - UART
    - 自动波特率检测器
  - USB 设备
    - USB 控制器驱动器
    - USB 框架
    - USB HID 类

第 2 章
MCU 闪存加载器协议

2.1 简介

本节介绍主机和 MCU 闪存加载器之间数据包传输的通用协议。介绍包括不同事务的数据包传输，例如无数据阶段的命令以及带传入或传出数据阶段的命令。下一节将介绍事务中使用的各种数据包类型。

从主机发出的每条命令都将使用一条响应命令来回复。

命令可能包括可选的数据阶段。

- 如果数据阶段为传入（从主机到 MCU 闪存加载器），则它是原始命令的一部分。
- 如果数据阶段为传出（从 MCU 闪存加载器到主机），则它是响应命令的一部分。

2.2 无数据阶段的命令

注

在这些图中，为响应命令或数据包而发送的 Ack 可以在处理命令/数据包之前、期间或之后的任何时间到达。

无数据阶段的命令

无数据阶段的命令的协议包含：

- 命令数据包（来自主机）
- 通用响应命令数据包（到主机）

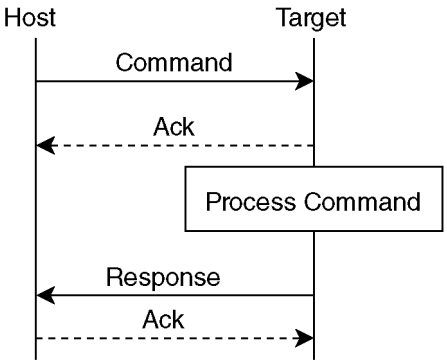

图 2-1 无数据阶段的命令

2.3 带传入数据阶段的命令

带传入数据阶段的命令的协议包含：

- 命令数据包（来自主机）（kCommandFlag_HasDataPhase 置位）
- 通用响应命令数据包（到主机）
- 传入数据包（来自主机）
- 通用响应命令数据包（到主机）

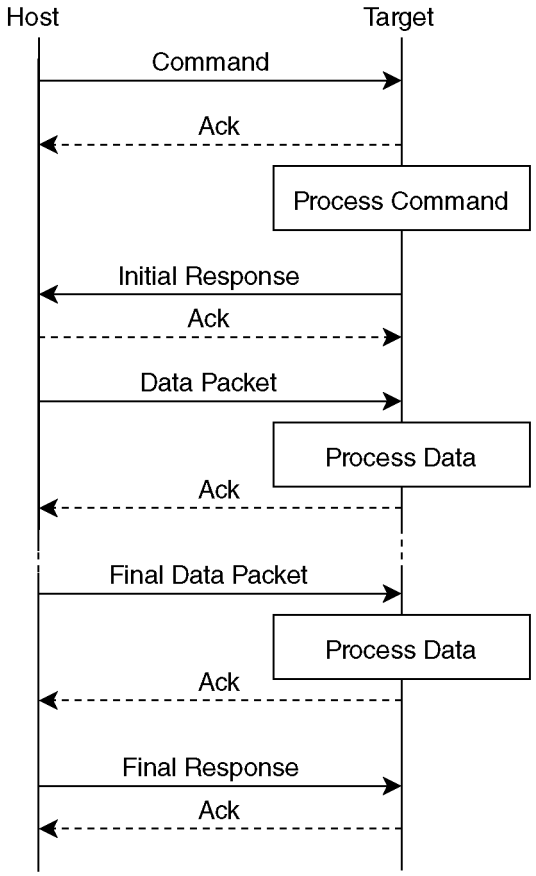

图 2-2. 带传入数据阶段的命令

注

- 在等待对命令的响应时，主机不再发送任何其他数据包。
- 如果在数据阶段开始之前，通用响应数据包的状态不是 kStatus_Success，则数据阶段中止。
- 接收方可以通过提前发送状态为 kStatus_AbortDataPhase 的最终通用响应来中止数据阶段。
- 主机可以通过发送零长度的数据包提前中止数据阶段。
- 在数据阶段之后发送的最终通用响应数据包将包括整个操作的状态。

2.4 带传出数据阶段的命令

带传出数据阶段的命令的协议包含：

- 命令数据包（来自主机）

- ReadMemory 响应命令数据包（到主机）（kCommandFlag_HasDataPhase 置位）
- 传出数据包（到主机）
- 通用响应命令数据包（到主机）

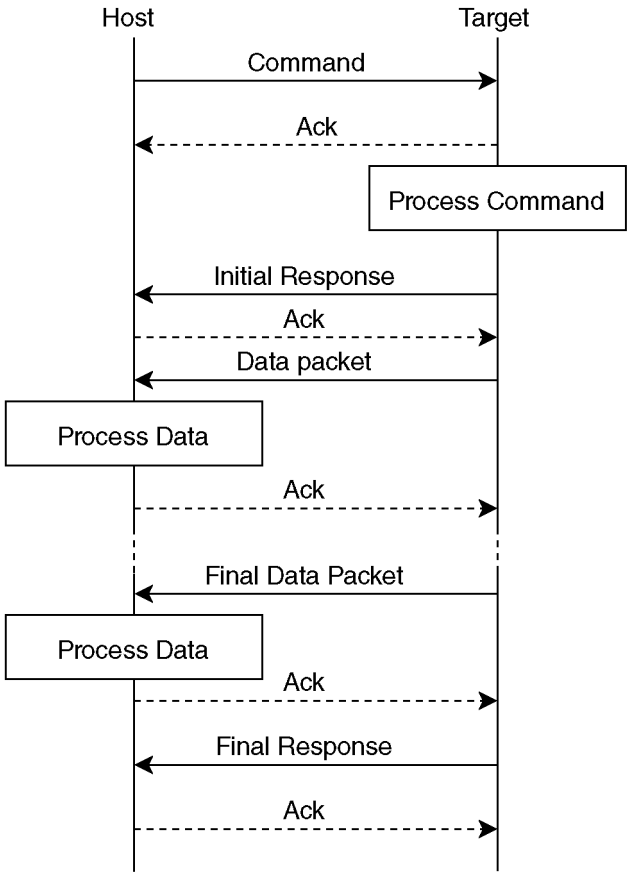

图 2-3. 带传出数据阶段的命令

注

- 对于传出数据阶段序列，数据阶段将视为响应命令的一部分。
- 在主机等待对命令的响应时，主机不再发送任何其他数据包。
- 如果在数据阶段开始之前，ReadMemory 响应命令数据包中没有 kCommandFlag_HasDataPhase 标志，则数据阶段中止。
- 可以通过主机提前发送状态为 kStatus_AbortDataPhase 的最终通用响应来中止数据阶段。发送方通过发送零长度的数据包提前中止数据阶段。
- 在数据阶段之后发送的最终通用响应数据包将包括整个操作的状态。

第 3 章
闪存加载器数据包类型

3.1 简介

MCU 闪存加载器设备以从机模式工作。所有数据通信均由主机发起，该主机可以是 PC 主机，也可以是嵌入式主机。MCU 闪存加载器设备是接收命令或数据包的目标机。主机和目标机之间的所有数据通信均采用分包形式。

注

术语“目标机”是指“MCU 闪存加载器设备”。

使用了 6 种数据包类型：

- Ping 数据包
- Ping 响应数据包
- 成帧数据包
- 命令数据包
- 数据包
- 响应数据包

数据包中的所有字段均按小端模式字节排序。

3.2 Ping 数据包

当目标机期待命令数据包时，可以随时从主机向目标机发送 Ping 数据包。如果选择的外设是 UART，则必须在执行其他任何通信之前发送 Ping 数据包，以便运行自动波特率。对于其他串行外设，这是可选项，但建议这样做以确定串行协议的版本。

作为对 Ping 数据包的响应，目标机将发送一个 Ping 响应数据包，这将在下一节中讨论。

表 3-1. Ping 数据包格式

字节编号	值	名称
0	0x5A	起始字节
1	0xA6	ping

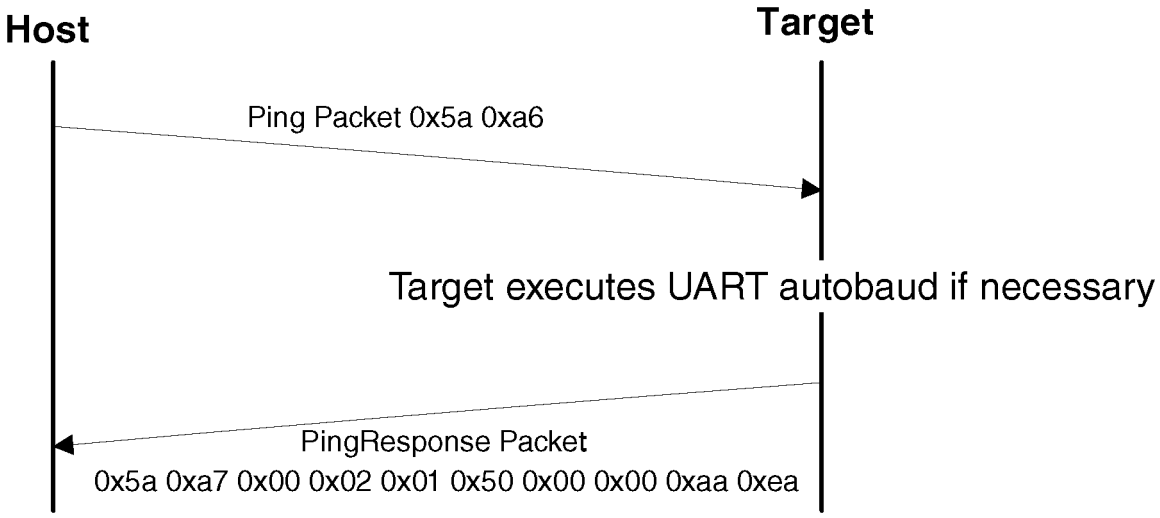

图 3-1. Ping 数据包协议序列

3.3 Ping 响应数据包

目标机在收到 Ping 数据包后将 Ping 响应数据包发送回主机。如果通过 UART 外设进行通信，则目标机使用传入的 Ping 数据包确定波特率，然后再用 Ping 响应数据包进行回复。主机收到 Ping 响应数据包后，将建立连接，然后主机开始向目标机发送命令。

表 3-2. Ping 响应数据包格式

字节编号	值	参数
0	0x5A	起始字节
1	0xA7	Ping 响应代码
2		协议错误修正
3		次要协议
4		主要协议
5		协议名称 = 'P' (0x50)
6		选项低
7		选项高

表格接下页……

表 3-2. Ping 响应数据包格式（续）

字节编号	值	参数
8		CRC16 低
9		CRC16 高

3.4 成帧数据包

成帧数据包用于在未内置此类功能的通信链路中进行流量控制和错误检测。成帧数据包结构位于链路层和命令层之间。它还打包了命令包和数据包。

每个含有单向发送数据的成帧数据包都会导致一个反向同步响应成帧数据包。

本小节描述的成帧数据包适用于串行外设 UART。USB HID 外设不使用成帧数据包，而是使用 USB 协议本身固有的分包功能。

表 3-3. 成帧数据包格式

字节编号	参数	说明
0	起始字节	值——0x5A
1	packetType	
2	length_low	长度是一个 16 位字段，指定整个命令或数据包的大小（单位为字节）。
3	length_high	
4	crc16_low	这是一个 16 位字段。CRC16 值涵盖整个成帧数据包，包括起始字节和命令或数据包，但不包括 CRC 字节。请参见此表后的 CRC16 算法。
5	crc16_high	
6 ...n	命令/数据包有效载荷	

一个仅包含起始字节和数据包类型的特殊成帧数据包用于主机和目标机之间的同步。

表 3-4. 特殊成帧数据包格式

字节编号	值	参数
0	0x5A	起始字节
1	0xAn	数据包

数据包类型字段从下列定义类型中指定数据包的类型：

表 3-5. packetType 字段

packetType	名称	说明
0xA1	kFramingPacketType_Ack	已成功接收上一个数据包；允许发送更多的数据包。
0xA2	kFramingPacketType_Nak	上一个数据包已损坏，必须重新发送。
0xA3	kFramingPacketType_AckAbort	中止数据阶段。
0xA4	kFramingPacketType_Command	成帧数据包中包含命令数据包有效载荷。
0xA5	kFramingPacketType_Data	成帧数据包中包含数据包有效载荷。
0xA6	kFramingPacketType_Ping	发送以验证另一方是否仍在线；也用于 UART 自动波特率。
0xA7	kFramingPacketType_PingResponse	对 Ping 的响应；包含成帧协议的版本号和选项。

3.5 CRC16 算法

本小节介绍 CRC16 算法。

CRC 是对成帧数据包头（不包括 CRC16 字段本身）中的每个字节进行计算，再加上所有有效载荷字节。CRC 算法是 CRC-16 的 XMODEM 变体。

XMODEM 变体的特征如下：

表 3-6. XMODEM 特征

width	16
polynomial	0x1021
init value	0x0000
reflect in	false
reflect out	false

表格接下页……

表 3-6. XMODEM 特征（续）

xor out	0x0000
check result	0x31c3

通过算法运行 ASCII 字符序列“123456789”来计算检查结果。

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
{
    uint32_t crc = 0;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
   {
        uint32_t i;
        uint32_t byte = src[j];
       crc ^= byte << 8;
        for (i = 0; i < 8; ++i)
        {
           uint32_t temp = crc << 1;
            if (crc & 0x8000)
            {
                temp ^= 0x1021;
          }
            crc = temp;
        }
    }
    return crc;
}
```

3.6 命令数据包

命令数据包包含一个 32 位命令头和一个 32 位参数列表。

表 3-7. 命令数据包格式

命令数据包格式（32 字节）										
命令头（4 字节）。				28 字节的参数（最多 7 个参数）						
标签	标志	Rsvd	参数个数	参数 1（32 位）	参数 2（32 位）	参数 3（32 位）	参数 4（32 位）	参数 5（32 位）	参数 6（32 位）	参数 7（32 位）
字节 0	字节 1	字节 2	字节 3	-	-	-	-	-	-	-

表 3-8. 命令头格式

字节编号	命令头字段
0	命令或响应标签
1	标志
2	保留。应为 0x00。
3	ParameterCount

命令头后是数个 32 位参数值，参数个数等于在头中指定的 ParameterCount 字段值。

命令数据包还被目标机用来将响应发送回主机。如第 3.4 节所述，对于所有 UART 传输，成帧数据包中都嵌入了命令数据包和数据包。

表 3-9. 命令标签

命令标签	名称	
0x01	FlashEraseAll	命令标签指定 MCU 闪存加载器支持的一条命令。这里列出了 MCU 闪存加载器的有效命令标签。
0x02	FlashEraseRegion	
0x03	ReadMemory	
0x04	WriteMemory	
0x05	FillMemory	
0x07	GetProperty	
0x08	保留	
0x09	Execute	
0x10	FlashReadResource	
0x11	保留	
0x0A	Call	
0x0B	Reset	
0x0C	SetProperty	
0x0E	eFuseProgram	
0x0F	eFuseRead	
0x10	FlashReadResource	
0x11	ConfigureMemory	

表 3-10. 响应标签

响应标签	名称	
0xA0	GenericResponse	响应标签指定 MCU 闪存加载器（目标机）返回给主机的一个响应。这里列出了有效响应标签。
0xA7	GetPropertyResponse（仅用于对 GetProperty 命令发送响应）	
0xA3	ReadMemoryResponse（仅用于对 ReadMemory 命令发送响应）	
0xAF	FlashReadOnceResponse（仅用于对 FlashReadOnce 命令发送响应）	
0xB0	FlashReadResourceResponse（仅用于对 FlashReadResource 命令发送响应）	

标志：每个命令数据包都包含一个标志字节。仅使用标志字节的位 0。如果标志字节的位 0 设置为 1，则数据包跟随在命令序列中。在数据阶段传输的字节数由参数数组中的特定命令参数确定。

ParameterCount：命令数据包中包含的参数个数。

参数：参数为字长（32 位）。由于默认最大数据包为 32 字节，命令数据包最多可包含 7 个参数。

3.7 响应数据包

响应数据包使用与命令数据包相同的格式（参见第 3.6 节）。响应类型包括：

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse
- FlashReadOnceResponse
- FlashReadResourceResponse

GenericResponse：MCU 闪存加载器处理完一条命令后，闪存加载器会将含有状态和命令标签信息的通用响应发送给主机。通用响应是命令协议序列中的最后一个数据包。通用响应数据包中包含命令包数据（通用响应标签= 0xA0）和参数列表（其定义参见下一节）。标头中的参数计数字段始终设置为 2，用于状态码和命令标签参数。

表 3-11. GenericResponse 参数

字节编号	参数	说明
0 - 3	状态码	状态码是目标机执行命令期间遇到的错误。如果命令成功执行，则返回 kStatus_Success 代码。
4 - 7	命令标签	命令标签参数标识对主机发送的命令的响应。

GetPropertyResponse：GetPropertyResponse 数据包由目标机发送，以响应使用 GetProperty 命令的主机查询。GetPropertyResponse 数据包中包含命令数据包数据，并且命令/响应标签设置为 GetPropertyResponse 标签值(0xA7)。

标头中的参数个数字段设置为大于 1，从而始终包含状态码和一个或多个属性值。

表 3-12. GetPropertyResponse 参数

字节编号	值	参数
0 - 3		状态码
4 - 7		属性值
…		…
		最多可以有 6 个属性值，受限于 32 位命令数据包的大小和属性类型。

ReadMemoryResponse：目标机发送 ReadMemoryResponse 数据包，以响应发送 ReadMemory 命令的主机 ReadMemoryResponse 数据包中包含命令数据包数据，并且命令 / 响应标签设置为 ReadMemoryResponse 标签值 (0xA3)，标志字段设置为 kCommandFlag_HasDataPhase (1)。

参数个数设置为 2，状态码和数据字节数参数如下所示。

表 3-13. ReadMemoryResponse 参数

字节编号	参数	说明
0 - 3	状态码	相关的读存储器命令的状态。
4 - 7	数据字节数	在数据阶段发送的字节数。

FlashReadOnceResponse：目标机发送 ReadMemoryResponse 数据包，以响应发送 FlashReadOnce 命令的主机 FlashReadOnceResponse 数据包中包含命令数据包数据，并且命令/响应标签设置为 FlashReadOnceResponse 标签值(0xAF)，标志字段设置为 0。参数计数设置为 2，加上在 FlashReadOnceCommand 中请求读取的*字数*。

表 3-14. FlashReadOnceResponse 参数

字节编号	值	参数
0 - 3		状态码
4 - 7		要读取的字节数
…		…
		最多可请求读取 20 个字节的数据。

第 4 章
MCU 闪存加载器命令 API

4.1 简介

所有 MCU 闪存加载器命令 API 均遵循由成帧数据包打包的命令数据包格式，如前几小节所述。

有关 MCU 闪存加载器支持的命令列表，请参见表 4-9。

有关 MCU 闪存加载器返回的状态码列表，请参见附录 A。

4.2 GetProperty 命令

GetProperty 命令用来查询闪存加载器的各种相关属性和设置。每个支持的属性都有一个与之关联的唯一 32 位标签。标签占据命令包的第一个参数。目标机将返回一个 GetPropertyResponse 数据包，其中包含在 GetProperty 命令中以该标签标识的属性值。

属性是可以使用 GetProperty 或 SetProperty 命令访问的已定义数据单元。属性可以是只读，也可以是读写。所有读写属性都是 32 位整数，因此可以在命令参数中方便传输。

有关 MCU 闪存加载器支持的属性及其关联的 32 位属性标签列表，请参见附录 B “ GetProperty 和 SetProperty 命令”。

GetProperty 命令所需的唯一参数是 32 位属性标签。

表 4-1. GetProperty 命令的参数

字节编号	命令
0 - 3	属性标签
4 - 7	外部存储器标识符（仅适用于获取外部存储器的属性）

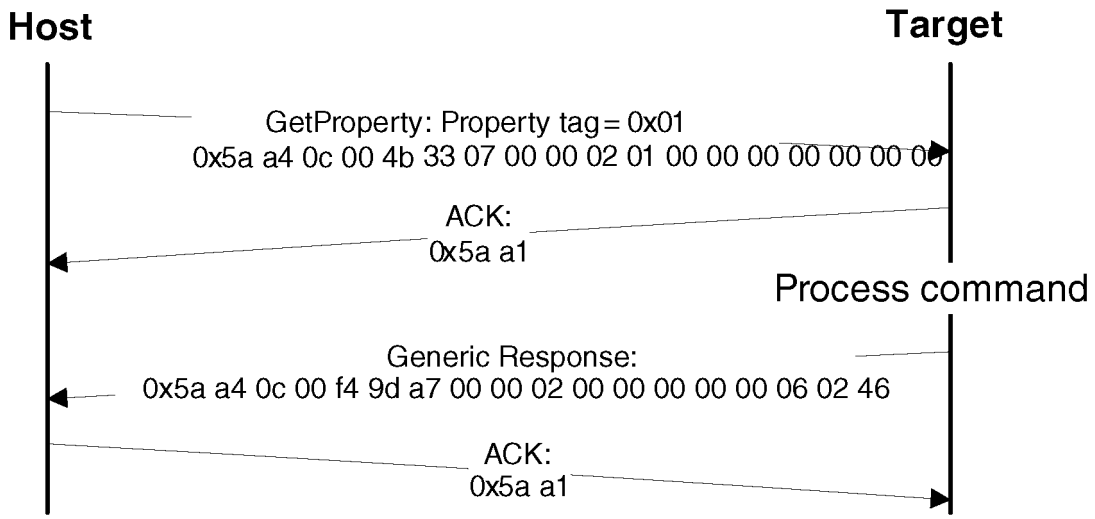

图 4-1. GetProperty 命令的协议序列

表 4-2. GetProperty 数据包格式示例

GetProperty	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4，kFramingPacketType_Command
	长度	0x0C 0x00
	crc16	0x4B 0x33
命令数据包	commandTag	0x07 - GetProperty
	标志	0x00
	保留	0x00
	parameterCount	0x02
	propertyTag	0x00000001 - CurrentVersion
	存储器 ID	0x00000000 -内部闪存

GetProperty 命令没有数据阶段。

响应：作为对 GetProperty 命令的响应，目标机发送一个 GetPropertyResponse 数据包，其响应标签设置为 0xA7。参数个数指示发送用于属性值的参数数量，第一个参数显示状态码 0，后面是属性值。下表显示了一个 GetPropertyResponse 数据包示例。

表 4-3. GetProperty 响应数据包格式示例

GetPropertyResponse	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4，kFramingPacketType_Command

表格接下页……

表 4-3. GetProperty 响应数据包格式示例（续）

GetPropertyResponse	参数	值
	长度	0x0c 0x00（12 字节）
	crc16	0xf4 9d
命令数据包	responseTag	0xA7
	标志	0x00
	保留	0x00
	parameterCount	0x02
	状态	0x00000000
	propertyValue	0x4b020600 - CurrentVersion

4.3 SetProperty 命令

SetProperty 命令用于更改闪存加载器的属性值或选项值。该命令接受与 GetProperty 命令一起使用的相同属性标签。但是，只有某些属性是可写的（参见附录 B）。如果尝试写入只读属性，则将返回错误，指示该属性为只读且无法更改。

SetProperty 命令所需的两个参数是属性标签和要设置的新值。

表 4-4.SetProperty 命令的参数

字节编号	命令
0 - 3	属性标签
4 - 7	属性值

SetProperty 命令

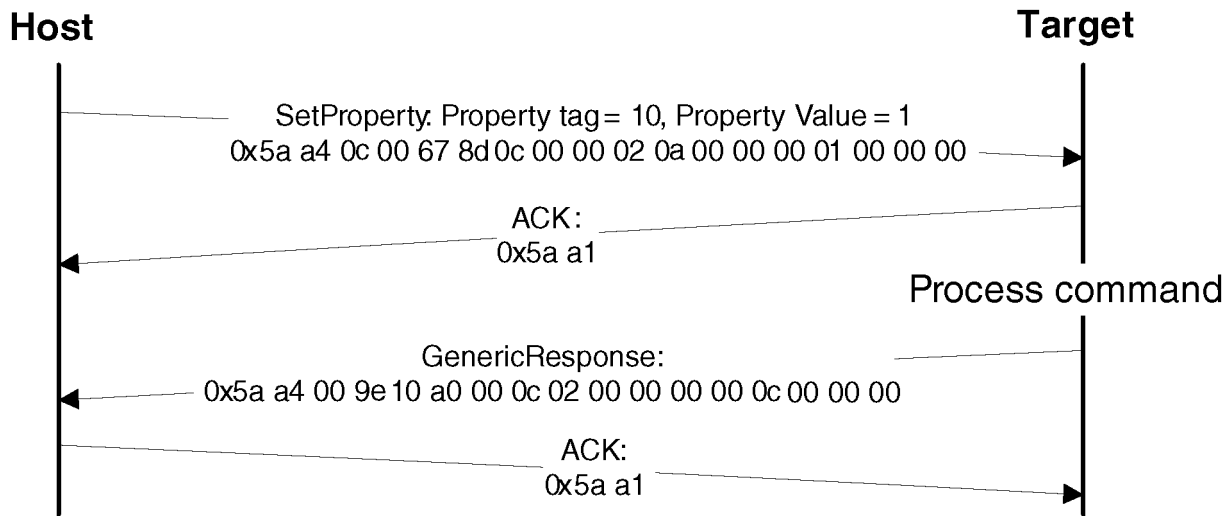

图 4-2. SetProperty 命令的协议序列

表 4-5. SetProperty 数据包格式示例

SetProperty	参数	值
成帧数据包	起始字节	0x5A
	数据包类型	0xA4，kFramingPacketType_Command
	长度	0x0C 0x00
	crc16	0x67 0x8D
命令数据包	commandTag	0x0C - 属性标签为 10 的 SetProperty
	标志	0x00
	保留	0x00
	parameterCount	0x02
	propertyTag	0x0000000A - VerifyWrites
	propertyValue	0x00000001

SetProperty 命令没有数据阶段。

响应：目标机返回一个 GenericResponse 数据包，使用下表中的一个状态码：

表 4-6. SetProperty 响应状态码

状态码
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

4.4 FlashEraseAll 命令

FlashEraseAll 命令执行整个闪存的擦除。如果有任何闪存区域受到保护，则 FlashEraseAll 命令将失败并返回错误状态码。如果通过设置 FTFA_FSEC 寄存器启用了 FlashEraseAll 命令（闪存安全性），则执行该命令将释放闪存安全性。但是，闪存配置字段的 FSEC 字段将被擦除，因此，除非对其进行重新编程，否则在下一次系统复位后将重新启用闪存安全性。FlashEraseAll 命令的命令标签在命令数据包的 commandTag 字段中设置为 0x01。

FlashEraseAll 命令需要一个存储器 ID。如果未指定存储器 ID，将默认选择内部闪存（存储器 ID = 0）。

表 4-7. FlashEraseAll 命令的参数

字节编号	参数	
0-3	存储器 ID	
	0x000	内部闪存
	0x010	内部闪存中的仅执行区域
	0x001	通过 QuadSPI 支持的串行 NOR
	0x008	通过 SEMC 支持的并行 NOR
	0x009	通过 FlexSPI 支持的串行 NOR
	0x100	通过 SEMC 支持的 SLC RAW NAND
	0x101	通过 FlexSPI 支持的串行 NAND
	0x110	通过 SPI 支持的串行 NOR/EEPROM
	0x120	通过 uSDHC 支持的 SD
	0x121	通过 uSDHC 支持的 eMMC

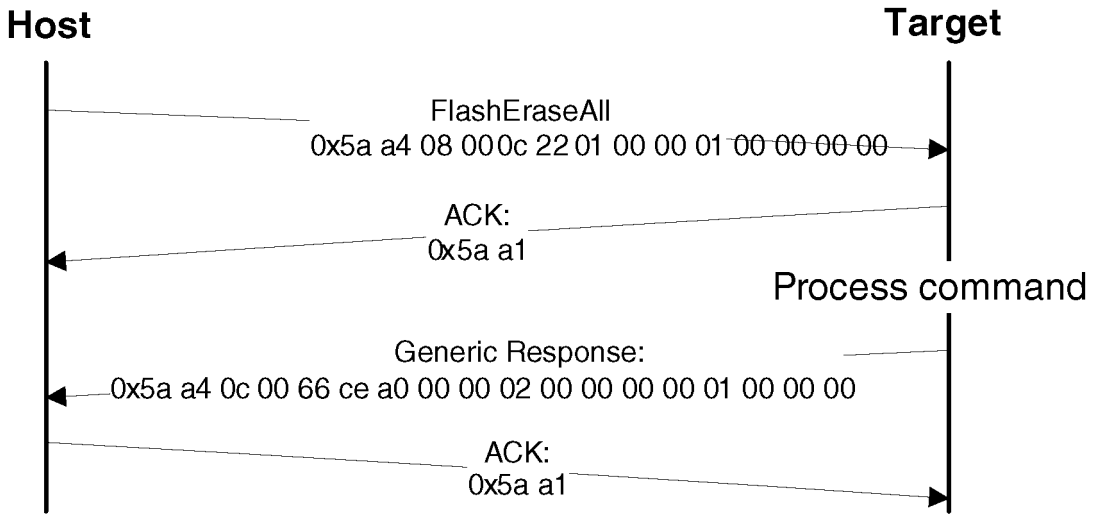

图 4-3. FlashEraseAll 命令的协议序列

表 4-8. FlashEraseAll 数据包格式示例

FlashEraseAll	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4，kFramingPacketType_Command
	长度	0x08 0x00
	crc16	0x0C 0x22
命令数据包	commandTag	0x01 - FlashEraseAll
	标志	0x00
	保留	0x00
	parameterCount	0x01
	存储器 ID	参见上表

FlashEraseAll 命令没有数据阶段。

响应：目标机返回一个 GenericResponse 数据包，其状态码在命令成功执行时设置为 kStatus_Success，否则设置为适当的错误状态码。

4.5 FlashEraseRegion 命令

FlashEraseRegion 命令执行擦除闪存的一个或多个扇区。

FlashEraseRegion 命令所需的两个参数是起始地址和字节数。起始地址和字节数参数必须为 4 字节对齐([1:0] = 00)，否则 FlashEraseRegion 命令失败并返回 kStatus_FlashAlignmentError(101)。如果指定的区域不能容纳闪存空间，则 FlashEraseRegion 命令将失败并返回 kStatus_FlashAddressError(102)。如果指定区域的任何部分受到保护，则 FlashEraseRegion 命令将失败并返回 kStatus_MemoryRangeInvalid(10200)。

表 4-9. FlashEraseRegion 命令的参数

字节编号	参数
0 - 3	起始地址
4 - 7	字节数
8 - 11	存储器 ID

FlashEraseRegion 命令没有数据阶段。

响应：目标机返回一个 GenericResponse 数据包，使用下表中的一个错误状态码：

表 4-10. FlashEraseRegion 响应状态码

状态码
kStatus_Success (0)
kStatus_MemoryRangeInvalid (10200)
kStatus_FlashAlignmentError (101)
kStatus_FlashAddressError (102)
kStatus_FlashAccessError (103)
kStatus_FlashProtectionViolation (104)
kStatus_FlashCommandFailure (105)

4.6 ReadMemory 命令

ReadMemory 命令返回给定地址处指定字节数的内存数据。该命令能够读取 CPU 可访问并且不受安全保护的任何内存区域。

ReadMemory 命令所需的两个参数是起始地址和字节数。存储器 ID 为可选参数。如果未指定存储器 ID，则将选择内部存储器为默认值。

表 4-11. 读取存储器命令参数

Byte	参数	说明
0-3	起始地址	要读取的存储器起始地址
4-7	字节数	读取并返回给调用方的字节数
8-11	存储器 ID	内部或外部存储器标识符

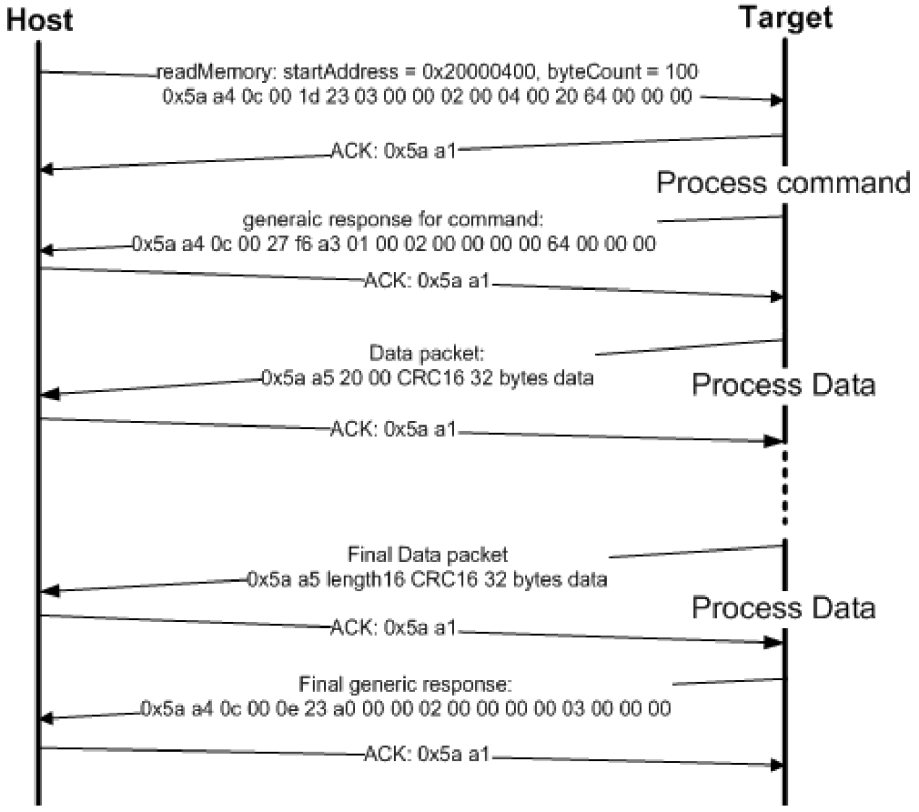

图 4-4. 读取存储器命令序列

表 4-12. ReadMemory 数据包格式示例

ReadMemory	参数	值
成帧数据包	起始字节	0x5A0xA4
	packetType	kFramingPacketType_Command
	长度	0x10 0x00
	crc16	0xF4 0x1B
命令数据包	commandTag	0x03 - readMemory

表格接下页……

表 4-12. ReadMemory 数据包格式示例（续）

ReadMemory	参数	值
	标志	0x00
	保留	0x00
	parameterCount	0x03
	startAddress	0x20000400
	byteCount	0x00000064
	memoryID	0x0

数据阶段：ReadMemory 命令没有数据阶段。因为目标机以从机模式工作，所以主机需要提取数据包，直到主机接收的数据字节数达到 ReadMemory 命令的 byteCount 参数的指定值为止。

响应：目标机返回一个 GenericResponse 数据包以及设置为 kStatus_Success（命令成功执行时）的状态码，或者适当的错误状态码。

4.7 WriteMemory 命令

WriteMemory 命令将在数据阶段提供的数据写入存储器（闪存或 RAM）中指定的字节范围内。但是，如果启用了闪存保护，则写入受保护扇区会失败。

写入闪存时必须格外小心。

- 首先，必须事先使用 FlashEraseAll、FlashEraseRegion 或 FlashEraseAllUnsecure 命令擦除任何已写入的闪存扇区。
- 写入闪存要求起始地址与页面大小对齐。
- 字节数四舍五入为页面大小的倍数，末尾字节以闪存擦除模式(0xff)填充。
- 如果 VerifyWrites 属性设置为 true，则对闪存的写操作还将执行闪存验证程序操作。

写入 RAM 时，无需对齐起始地址，也不需要填充数据。

WriteMemory 命令所需的两个参数为起始地址和字节数。存储器 ID 为可选参数。如果未指定存储器 ID，则将选择内部存储器为默认值。

表 4-13. WriteMemory 命令的参数

字节编号	命令
0 - 3	起始地址
4 - 7	字节数
8 - 11	存储器 ID

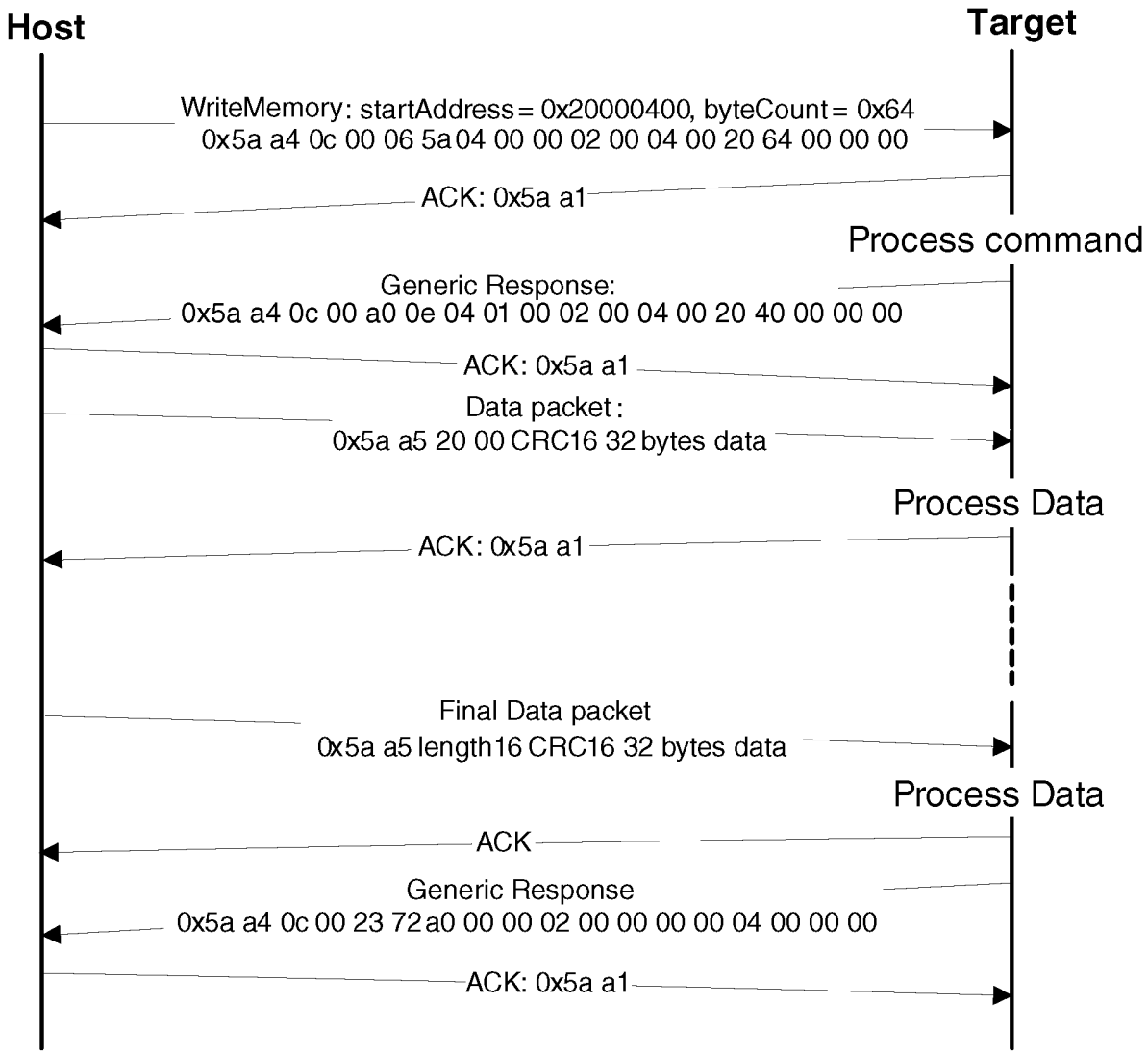

图 4-5.WriteMemory 命令的协议序列

表 4-14.WriteMemory 数据包格式示例

WriteMemory	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4，kFramingPacketType_Command

表格接下页……

表 4-14. WriteMemory 数据包格式示例（续）

WriteMemory	参数	值
	长度	0x10 0x00
	crc16	0x97 0xDD
命令数据包	commandTag	0x04 - writeMemory
	标志	0x01
	保留	0x00
	parameterCount	0x03
	startAddress	0x20000400
	byteCount	0x00000064
	memoryID	0x0

数据阶段：WriteMemory 命令没有数据阶段。主机需要发送数据包，直到目标机接收的数据字节数达到 WriteMemory 命令的 byteCount 参数的指定值为止。

响应：目标机返回一个 GenericResponse 数据包以及设置为 kStatus_Success（命令成功执行时）的状态码，或者适当的错误状态码。

4.8 FillMemory 命令

FillMemory 命令使用数据模式填充存储器中的字节范围。它遵循与 WriteMemory 命令相同的规则。FillMemory 和 WriteMemory 之间的区别在于 FillMemory 命令参数中包含数据模式，并且 FillMemory 命令没有数据阶段，而 WriteMemory 具有数据阶段。

表 4-15. FillMemory 命令的参数

字节编号	命令
0 - 3	要填充的存储器起始地址
4 - 7	要使用模式写入的字节数 • 起始地址应为 32 位对齐。 • 字节数必须能被 4 整除。 **注：** 对于使用 FTFE 闪存的任何器件，起始地址应为 64 位对齐，字节数必须能被 8 整除。
8 - 11	32 位模式

- 要使用字节模式（8 位）填充，必须在 32 位模式中将字节复制 4 次。
- 要使用短整型模式（16 位）填充，必须在 32 位模式中将短整型值复制 2 次。

例如，要用 0xFE 填充字节值，字模式为 0xFEFEFEFE；要填充短整形值 0x5AFE，字模式则为 0x5AFE5AFE。

写入闪存时必须格外小心。

- 首先，必须事先使用 FlashEraseAll、FlashEraseRegion 或 FlashEraseAllUnsecure 命令擦除任何已写入的闪存扇区。
- 写入闪存要求起始地址与页面大小对齐。
- 如果 VerifyWrites 属性设置为 true，则对闪存的写操作还将执行闪存验证程序操作。

写入 RAM 时，无需对齐起始地址。

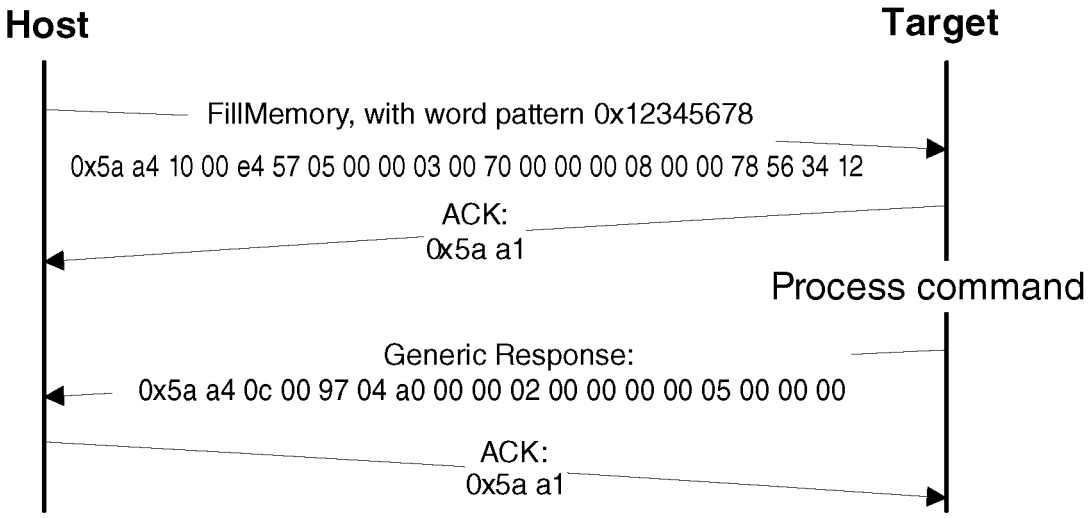

图 4-6. FillMemory 命令的协议序列

表 4-16. FillMemory 数据包格式示例

FillMemory	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4，kFramingPacketType_Command
	长度	0x10 0x00
	crc16	0xE4 0x57
命令数据包	commandTag	0x05 - FillMemory
	标志	0x00
	保留	0x00
	parameterCount	0x03

表格接下页……

表 4-16. FillMemory 数据包格式示例（续）

FillMemory	参数	值
	startAddress	0x00007000
	byteCount	0x00000800
	patternWord	0x12345678

FillMemory 命令没有数据阶段。

响应：成功执行命令后，目标机（MCU 闪存加载器）返回一个 GenericResponse 数据包以及设置为 kStatus_Success 的状态码，或者适当的错误状态码。

4.9 Execute 命令

Execute 命令的结果是，闪存加载器将程序指针设置为所提供的跳转地址处的代码，将 R0 设置为所提供的自变量，并将堆栈指针设置为所提供的堆栈指针地址。在跳转之前，系统将返回到复位状态。

Execute 命令所需的参数为跳转地址、函数参数指针和堆栈指针。如果堆栈指针设置为零，则被调用的代码负责在使用堆栈之前设置处理器堆栈指针。

表 4-17. Execute 命令的参数

字节编号	命令
0 - 3	跳转地址
4 - 7	自变量字
8 - 11	堆栈指针地址

Execute 命令没有数据阶段。

响应：在运行 Execute 命令之前，目标机将验证参数并返回一个 GenericResponse 数据包以及设置为 kStatus_Success 的状态码，或者适当的错误状态码。

4.10 Call 命令

Call 命令执行在该命令所发送地址处写入存储器的函数。该地址必须是可访问闪存（内部或外部）或 RAM 中的有效存储位置。该命令支持传递一个 32 位参数。尽管该命令支持堆栈地址，但此时仍使用当前堆栈指针进行调用。执行该函数后，将在通用响应消息中返回一个 32 位值。

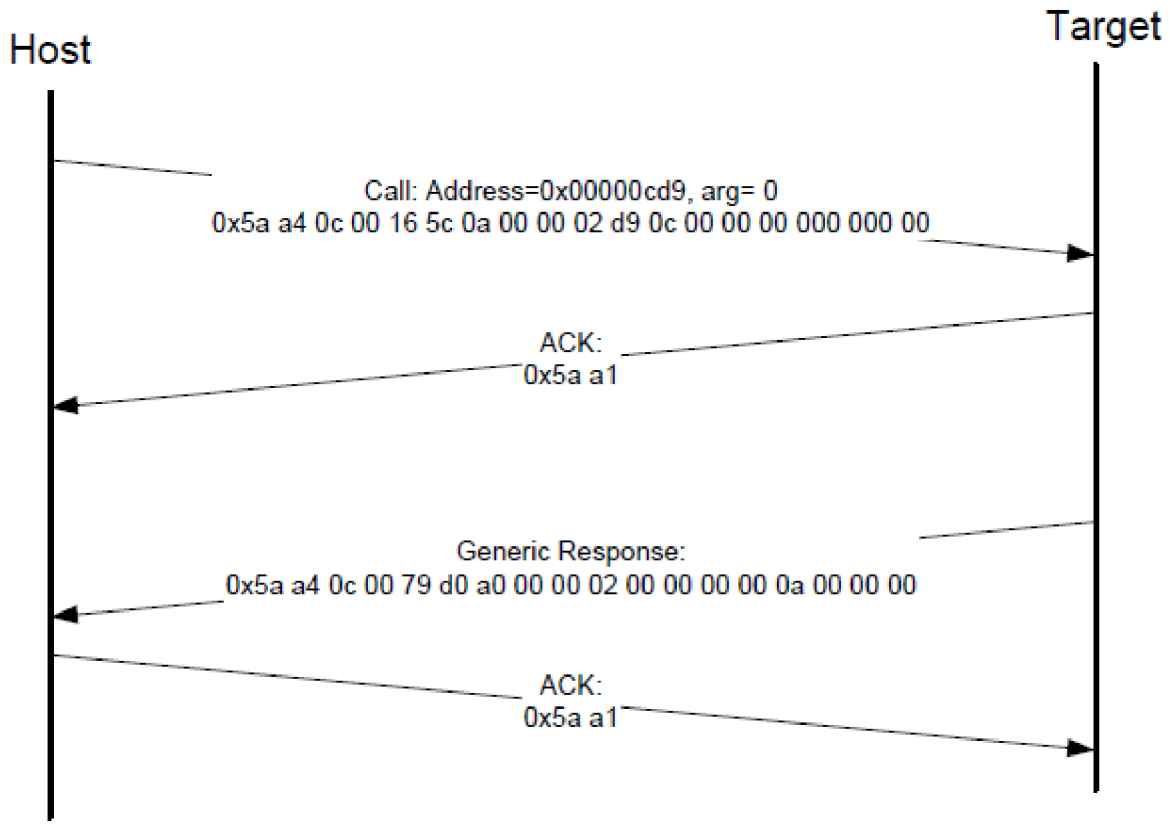

图 4-7. Call 命令的协议序列

表 4-18. Call 命令的参数

字节编号	命令
0 - 3	调用地址
4 - 7	自变量字
8 - 11	堆栈指针

响应：目标机返回一个 GenericResponse 数据包以及设置为所调用函数的返回值或 kStatus_InvalidArgument (105)的状态码。

4.11 Reset 命令

Reset 命令会导致闪存加载器复位芯片。

Reset 命令无需任何参数。

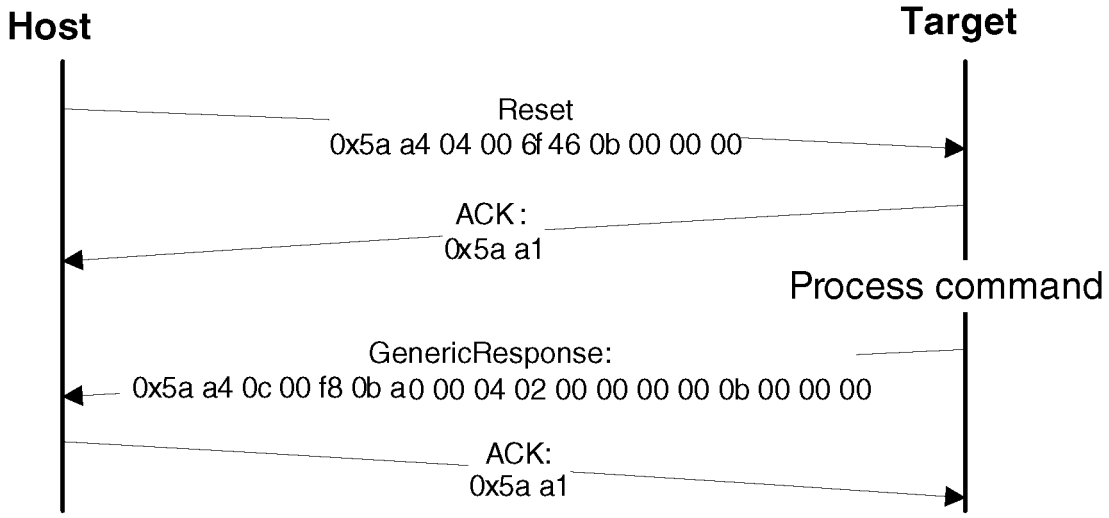

图 4-8. Reset 命令的协议序列

表 4-19. Reset 命令数据包格式示例

Reset	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4，kFramingPacketType_Command
	长度	0x04 0x00
	crc16	0x6F 0x46
命令数据包	commandTag	0x0B - reset
	标志	0x00
	保留	0x00
	parameterCount	0x02

Reset 命令没有数据阶段。

响应：在芯片复位之前，目标机返回一个 GenericResponse 数据包，其状态码设置为 kStatus_Success。

通过闪存加载器成功配置闪存映像后，还可以使用 Reset 命令切换为从闪存启动。发出 reset 命令后，请等待 5 秒钟，以便用户应用能够从闪存开始运行。

4.12 FlashProgramOnce/eFuseProgramOnce 命令

FlashProgramOnce/eFuseProgramOnce 命令将数据（由命令数据包提供）写入烧写一次字段中的指定字节范围。写入烧写一次字段时必须格外小心。

- 烧写一次字段只支持进行一次编程，因此任何对烧写一次字段进行重新编程的尝试将获取一个错误响应。
- 向烧写一次字段写入要求字节数为 4。

FlashProgramOnce 命令使用三个参数：索引、字节数(byteCount)和数据。

表 4-20. FlashProgramOnce 命令的参数

字节编号	命令
0 - 3	烧写一次/eFuse 字段索引
4 - 7	字节数（对于 FlashProgramOnce 必须为 4 或 8；对于 eFuseProgramOnce 必须为 4）
8 - 11	数据
12 - 16	数据

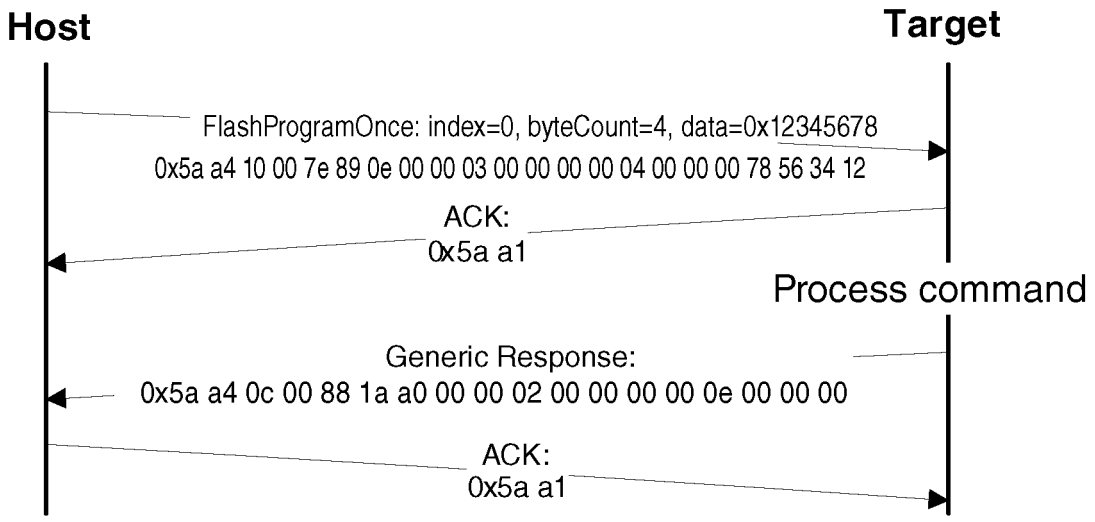

图 4-9. FlashProgramOnce 命令的协议序列

表 4-21. FlashProgramOnce 数据包格式示例

FlashProgramOnce	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4，kFramingPacketType_Command
	长度	0x10 0x00
	crc16	0x7E4 0x89
命令数据包	commandTag	0x0E - FlashProgramOnce
	标志	0
	保留	0
	parameterCount	3
	索引	0x0000_0000
	byteCount	0x0000_0004
	数据	0x1234_5678

响应：成功执行命令后，目标机（MCU 闪存加载器）返回一个 GenericResponse 数据包以及设置为 kStatus_Success 的状态码，或者适当的错误状态码。

4.13 FlashReadOnce/eFuseReadOnce 命令

FlashReadOnce/eFuseReadOnce 命令通过给定的索引和字节数返回烧写一次字段的内容。FlashReadOnce 命令使用两个参数：索引和字节数(byteCount)。

表 4-22. FlashReadOnce 命令的参数

字节编号	参数	说明
0 - 3	索引	（要读取的）烧写一次字段的索引
4 - 7	byteCount	要读取并返回到调用程序的字节数（对于 eFuseReadOnce 必须为 4）

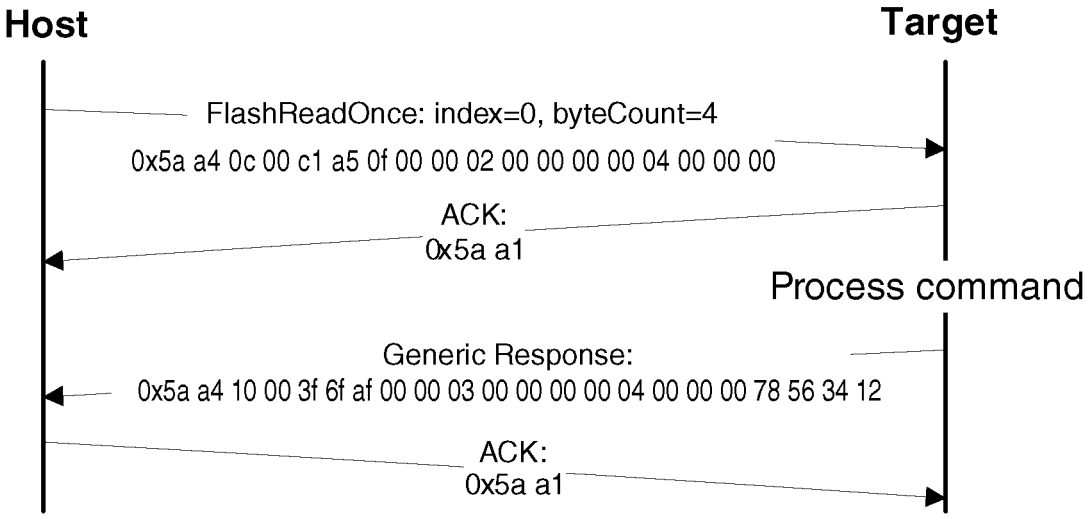

图 4-10. FlashReadOnce 命令的协议序列

表 4-23. FlashReadOnce 数据包格式示例

FlashReadOnce	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4
	长度	0x0C 0x00
	crc	0xC1 0xA5
命令数据包	commandTag	0x0F - FlashReadOnce
	标志	0x00
	保留	0x00
	parameterCount	0x02
	索引	0x0000_0000
	byteCount	0x0000_0004

表 4-24. FlashReadOnce 响应格式示例

FlashReadOnce 响应	参数	值
成帧数据包	起始字节	0x5A
	packetType	0xA4
	长度	0x10 0x00
	crc	0x3F 0x6F
命令数据包	commandTag	0xAF
	标志	0x00
	保留	0x00
	parameterCount	0x03

表格接下页……

表 4-24. FlashReadOnce 响应格式示例（续）

FlashReadOnce 响应	参数	值
	状态	0x0000_0000
	byteCount	0x0000_0004
	数据	0x1234_5678

响应：成功执行命令后，目标机返回一个 FlashReadOnceResponse 数据包，包含设置为 kStatus_Success 的状态码、字节数和从烧写一次字段读取的相应数据，或者返回的数据包状态码设置为适当的错误状态码，并且字节数设置为 0。

4.14 Configure Memory 命令

Configure Memory 命令使用预编程映像来配置外部存储器设备。传递的参数为包含配置数据的存储器 ID 和存储器地址。将配置数据写入 RAM 或闪存位置，然后该命令指示闪存加载器使用此位置的数据来配置外部存储器设备。有关配置数据的详细信息，请参见第 6 章“外部存储器支持”。

表 4-25. Configure Memory 命令的参数

字节编号	命令
0 - 3	存储器 ID
4 - 7	配置块地址

响应：目标机（MCU 闪存加载器）返回一个 GenericResponse 数据包以及设置为 kStatus_Success（命令成功执行时）的状态码，或者适当的错误码。

4.15 ReceiveSBFile 命令

ReceiveSBFile 命令用于向目标机传输 SB 文件。该命令仅指定在数据阶段发送的 SB 文件的大小（单位为字节）。闪存加载器接收到 SB 文件后将对其进行处理。

表 4-26. Receive SB File 命令的参数

字节编号	命令
0 - 3	字节数

数据阶段：Receive SB File 命令没有数据阶段。主机需要发送数据包，直到目标机接收的数据字节数达到 Receive SB File 命令的 byteCount 参数的指定值为止。

响应：目标机返回一个 GenericResponse 数据包以及设置为 kStatus_Success（命令成功执行时）的状态码，或者适当的错误码。

第 5 章
支持的外设

5.1 简介

本小节介绍 MCU 闪存加载器支持的外设。

5.2 UART 外设

MCU 闪存加载器集成了 UART 外设自动波特率检测算法，从而提供了灵活的波特率选择。

自动波特率特性：如果使用 UARTn 连接到闪存加载器，则 UARTn_RX 引脚在检测阶段必须保持高电平且不得浮空，以符合自动波特率检测算法的要求。闪存加载器在 UARTn_RX 上检测到 ping 数据包(0x5A 0xA6)之后，闪存加载器固件将执行自动波特序列。如果波特率检测成功，则闪存加载器将以检测到的波特率发送 ping 数据包响应[(0x5A 0xA7)、协议版本（4 字节）、协议版本选项（2 字节）和 crc16（2 字节）]。然后，MCU 闪存加载器进入循环，等待通过 UART 外设传输的闪存加载器命令。

注

必须以固定的 UART 传输模式（8 位数据，无奇偶校验位和 1 个停止位）连续发送 ping 数据包的数据字节（字节之间的间隔不超过 80 ms）。如果逐一发送 ping 数据包的字节，字节间的延迟超过 80 ms，则自动波特率检测算法可能会计算出错误的波特率。

支持的波特率：波特率与 MCU 内核和系统时钟频率密切相关。支持的典型波特率包括 9600、19200、38400 和 57600。

数据包传输：自动波特率检测成功后，可以通过 UART 外设进行闪存加载器通信。以下流程图显示：

- 主机如何检测来自目标机的 ACK
- 主机如何检测来自目标机的 ping 响应
- 主机如何检测来自目标机的命令响应

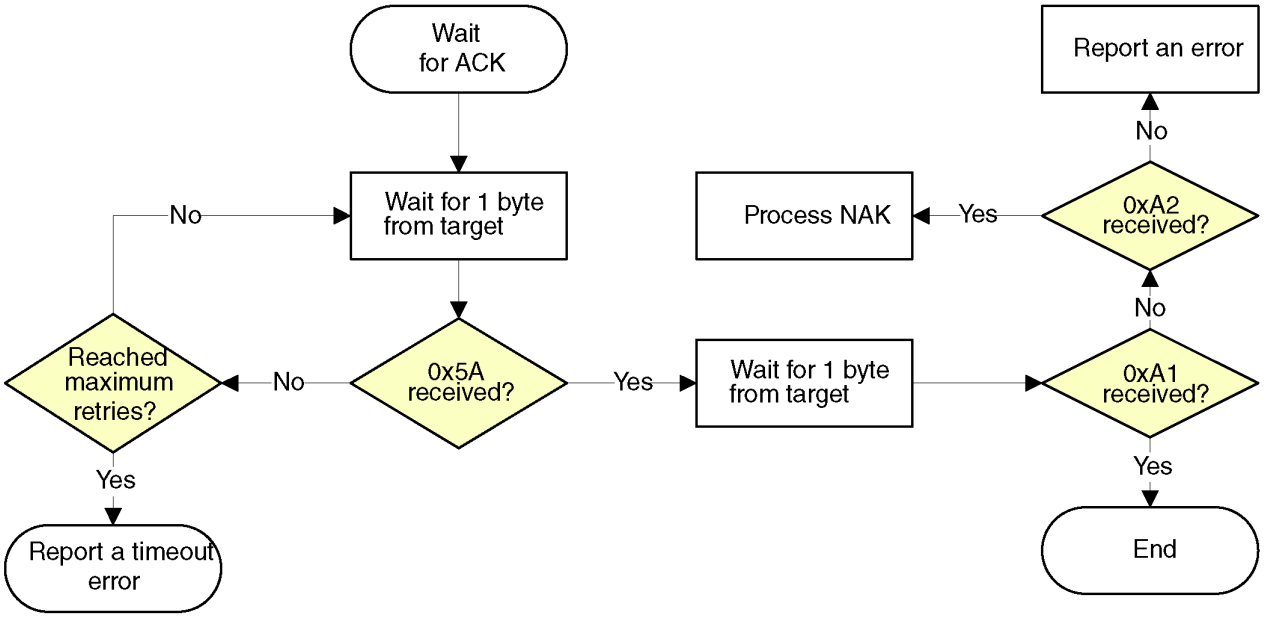

图 5-1. 主机通过 UART 从目标机读取 ACK

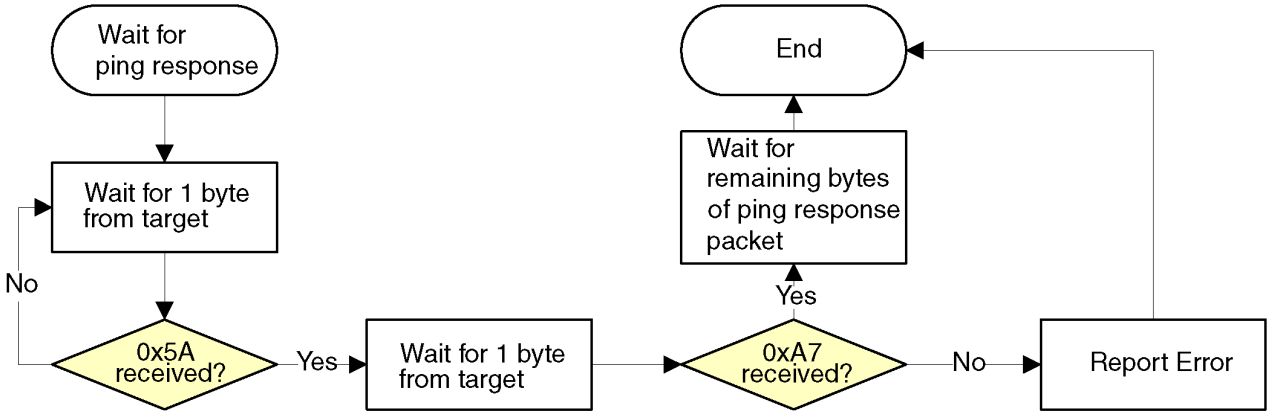

图 5-2. 主机通过 UART 从目标机读取 ping 响应

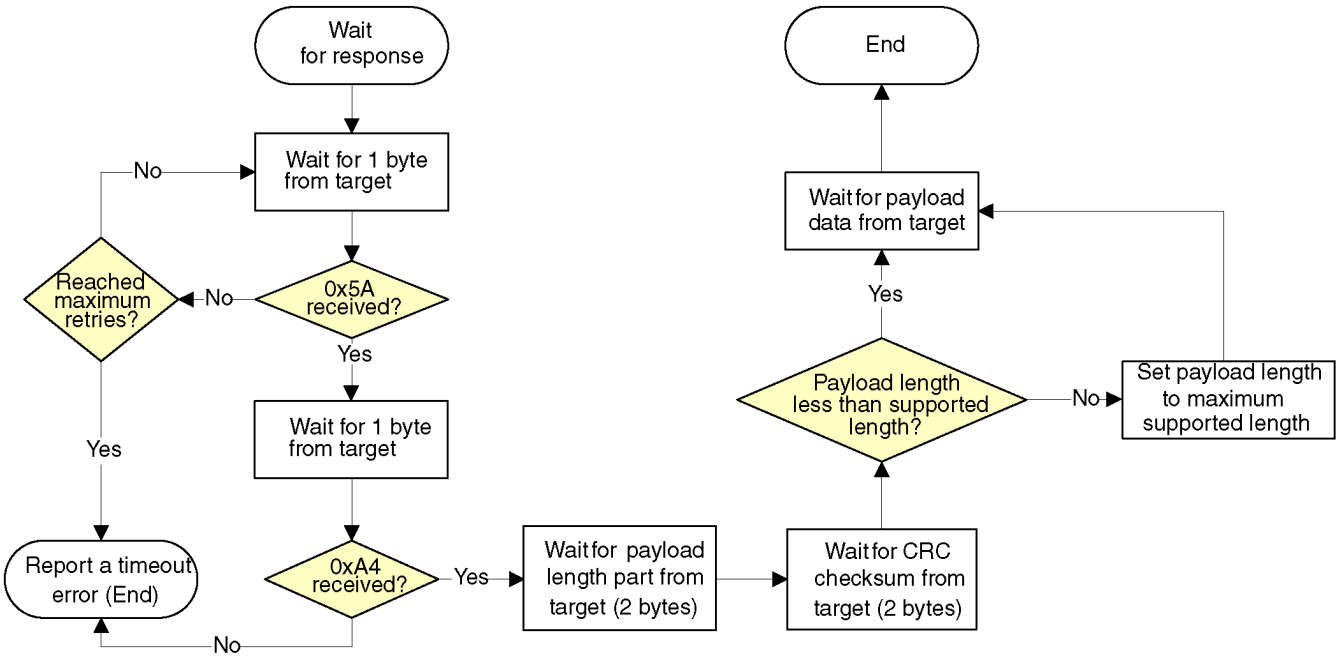

图 5-3. 主机通过 UART 从目标机读取命令响应

5.3 USB HID 外设

MCU 闪存加载器支持通过 USB 外设将数据加载到闪存中。将目标机作为 USB HID 类实现。

USB HID 不使用成帧数据包，而是使用 USB 协议本身固有的分包功能。设备到 NAK Out 传输（直到它们可被接收）的能力提供了所需的流量控制。每个 USB 数据包的内置 CRC 提供所需的错误检测。

5.3.1 设备描述符

MCU 闪存加载器配置的 USB 默认 VID/PID/字符串如下所示：

默认 VID/PID：

- 对于前飞思卡尔器件
  - VID = 0x15A2
  - PID = 0x0073
- 对于恩智浦器件
  - VID = 0x1FC9
  - PID = 0x007F

默认字符串：

- 对于前飞思卡尔器件
  - Manufacturer [1] = "Freescale Semiconductor Inc.”
  - Product [2] = "Kinetis bootloader"
- 对于恩智浦器件
  - Manufacturer [1] = "NXP Semiconductor Inc."
  - Product [2] = "Kinetis bootloader"

5.3.2 端点

HID 外设使用 3 个端点：

- 控制(0)
- 中断 IN (1)
- 中断 OUT (2)

中断 OUT 端点虽然是 HID 类设备的可选端点，但 MCU 闪存加载器将其用作管道，固件可以在其中 NAK 发送来自 USB 主机的请求。

5.3.3 HID 报告

闪存加载器 USB HID 外设定义并使用了 4 个 HID 报告。报告 ID 决定了在报告中发送数据包的方向和类型。否则，所有报告的内容均相同。

表 5-1. HID 报告

报告 ID	数据包类型	方向
1	命令	OUT
2	数据	OUT
3	命令	IN
4	数据	IN

以下属性适用于所有报告：

表 5-2. 报告属性

用量最小值	1
用量最大值	1
逻辑最小值	0

表格接下页……

表 5-2. 报告属性（续）

逻辑最大值	255
报告大小	8
报告计数	34

每个报告最大为 34 个字节。这是由最小的闪存加载器数据包大小（32 字节）加上 2 字节的报告标头得出的，该报告标头指示在报告中发送的数据包的长度（单位为字节）。

注

未来，最大报告大小可能会增加，以支持传输更大的数据包。
或者，可以添加最大尺寸值更大的其他报告。

在所有报告中发送的实际数据如下所示：

表 5-3. 报告数据

0	报告 ID
1	数据包长度 LSB
2	数据包长度 MSB
3	数据包[0]
4	数据包[1]
5	数据包[2]
	...
N+3-1	数据包[N-1]

此数据包括报告 ID，如果在 HID 报告描述符中定义了多个报告，则需要此 ID。发送和接收的实际数据的最大长度为 35 个字节。数据包长度标头以小端格式写入，并将其设置为报告中发送的数据包大小（单位为字节）。此大小不包括报告 ID 或数据包长度标头本身。在数据阶段，数据包大小为 0 表示来自接收方的数据阶段中止请求。

第 6 章
外部存储器支持

6.1 简介

本小节介绍 MCU 闪存加载器支持的外部存储器设备。要正确使用外部存储器设备，必须使用相应的配置文件启用该设备。闪存加载器无法访问未启用的外部存储设备。MCU 闪存加载器使用存储器标识符启用特定的外部存储设备，如下所示。

表 6-1. 外部存储器设备的存储器 ID

存储器标识符	外部存储器设备
0x01	“QuadSPI 模块上的串行 NOR”
0x08	“SEMC 模块上的并行 NOR”
0x09	“FlexSPI 模块上的串行 NOR”
0x0a	“SPIFI 上的串行 NOR”
0x100	“SEMC 模块上的 SLC raw NAND”
0x101	“FlexSPI 模块上的串行 NAND”
0x110	“LPSPI 模块上的串行 NOR/EEPROM”
0x120	“uSDHC 上的 SD”
0x121	“uSDHC 上的 eMMC”

6.2 通过 FlexSPI 支持的串行 NOR 闪存

MCU 闪存加载器通过 FlexSPI 模块支持读取、写入和擦除外部串行 NOR 闪存设备。在访问串行 NOR 闪存设备之前，必须使用简化的 FlexSPI NOR 配置选项块或完整的 512 字节 FlexSPI NOR 配置块来正确配置 FlexSPI 模块。闪存加载器可以基于简化的闪存配置选项块为市场上大多数串行 NOR 闪存设备生成 512 字节的 FlexSPI NOR 配置块。为了保护外部串行 NOR 闪存的知识产权，如果芯片包含 BEE 模块，则闪存加载器还支持使用 OTPMK/SNVS 密钥进行映像加密和编程。更多信息请参见“安全实用程序”章节。

6.2.1 FlexSPI NOR 配置块

表 6-2. 外部存储器设备的存储器 ID

名称	偏移	大小（字节）	说明
标签	0x000	4	0x42464346，ascii：“FCFB”
版本	0x004	4	0x56010000 [07:00] bugfix [15:08] 次要 [23:16] 主要 = 1 [31:24] ascii“V”
-	0x008	4	保留
readSampleClkSrc	0x00c	1	0 -内部环回 1 -从 DQS 填充环回 3 -闪存提供的 DQS
csHoldTime	0x00d	1	串行闪存 CS 保持时间 建议默认值为 0x03
csSetupTime	0x00e	1	串行闪存 CS 建立时间 建议默认值为 0x03
columnAdressWidth	0x00f	1	3 -用于 HyperFlash/HyperRAM 12/13 -用于串行 NAND，请参见数据手册以查找正确的值 0 -其他设备
deviceModeCfgEnable	0x010	1	设备模式配置启用特性 0 -禁用。 1 -启用。
deviceModeType	0x011	1	指定配置命令的类型 0 -通用命令 1 - Quad 启用 2 - SPI 到 OPI 其他-保留

表格接下页……

表 6-2. 外部存储器设备的存储器 ID（续）

名称	偏移	大小（字节）	说明
waitTimeCfgCommands	0x012	2	所有配置命令的等待时间，单位：100 µs。 0 -使用读取状态命令来确定配置命令的繁忙状态 其他-将配置命令延迟“waitTimeCfgCommads” * 100 µs
deviceModeSeq	0x014	4	设备模式配置的序列参数 [7:0] LUT 序列号 [15:8] 此序列的 LUT 序列索引 [31:16] 保留供日后使用
deviceModeArg	0x018	4	设备模式自变量，仅当 deviceModeCfgEnable = 1 时有效
configCmdEnable	0x01c	1	配置命令启用特性 0 -禁用 1 -启用
configModeType	0x01d	3	配置模式类型，与“ deviceModeType”的定义相同
configCmdSeqs	0x020	12	配置命令序列，支持 4 个单独的配置命令序列
-	0x02c	4	保留
cfgCmdArgs	0x030	12	每个单独的配置命令序列的自变量
-	0x03c	4	保留
controllerMiscOption	0x040	4	Bit0 -启用差分时钟 Bit2 -启用并行模式 Bit3 -启用可按字寻址 Bit4 -启用安全配置频率 Bit5 -启用填充设置覆盖 Bit6 -启用 DDR 模式 其他-保留

表格接下页……

表 6-2. 外部存储器设备的存储器 ID（续）

名称	偏移	大小（字节）	说明
deviceType	0x044	1	1 -串行 NOR 2 -串行 NAND
sflashPadType	0x045	1	1 -单填充 2 -双填充 4 -四填充 8 -八填充 其他-无效值
serialClkFreq	0x046	1	有关特定器件值的更多详细信息，请参见 SoC RM 中的“系统启动”章节。
lutCustomSeqEnable	0x047	1	0 -使用预定义的 LUT 序列索引和编号 1 -使用此块中提供的 LUT 序列参数
保留	0x048	8	保留
sflashA1Size	0x050	4	对于 SPI NOR，需要填写实际大小 对于 SPI NAND，需要填写实际大小* 2
sflashA2Size	0x054	4	对于 SPI NOR，需要填写实际大小 对于 SPI NAND，需要填写实际大小* 2
sflashB1Size	0x058	4	对于 SPI NOR，需要填写实际大小 对于 SPI NAND，需要填写实际大小* 2
sflashB2Size	0x05c	4	对于 SPI NOR，需要填写实际大小 对于 SPI NAND，需要填写实际大小* 2
csPadSettingOverride	0x060	4	如果不支持，则置 0
sclkPadSettingOverride	0x064	4	如果不支持，则置 0
dataPadSettingOverride	0x068	4	如果不支持，则置 0
dqsPadSettingOverride	0x06c	4	如果不支持，则置 0
timeoutInMs	0x070	4	读/写期间的最大等待时间 在 ROM 中不使用
commandInterval	0x074	4	单位：ns 目前，它用于高工作频率下的 SPI NAND。

表格接下页……

表 6-2. 外部存储器设备的存储器 ID（续）

名称	偏移	大小（字节）	说明
dataValidTime	0x078	4	从时钟沿到数据有效沿的时间，单位为 ns 当 FlexSPI 根时钟小于 100MHz 且读取采样时钟源是设备提供的不支持 CK2 的 DQS 信号时，使用此字段 [31:16] DLLB 的数据有效时间，以 0.1ns 为单位 [15:0] DLLA 的数据有效时间，以 0.1ns 为单位
busyOffset	0x07c	2	忙位偏移，有效范围：0-31
busyBitPolarity	0x07e	2	0 -如果设备忙，则忙位为 1 1 -如果设备忙，则忙位为 0
lookupTable	0x080	256	查询表
lutCustomSeq	0x180	48	LUT 自定义序列，请参见下表
-	0x1b0	16	保留
pageSize	0x1c0	4	闪存页面大小
sectorSize	0x1c4	4	闪存扇区大小
ipCmdSerialClkFreq	0x1c8	4	IP 命令时钟频率，与“serialClkFreq”的定义相同
isUniformBlockSize	0x1c9	4	扇区/块大小是否相同
-	0x1ca	2	-
serialNorType	0x1cc	1	串行 NOR 闪存类型： 0 -扩展 SPI 1 - HyperBus 2 - Octal DDR
needExitNoCmdMode	0x1cd	4	保留，置 0
halfClkForNonReadCmd	0x1ce	1	将 SDR 命令的时钟频率除以 2 需要为仅支持 DDR 读取的设备设置，其他命令则为 SDR 命令
needrestorNoCmdMode	0x1cf	1	保留，置 0
blockSize	0x1d0	4	闪存块大小
-	0x1d4	44	保留

注

要为某些特定设备自定义 LUT 序列，用户需要启用“**lutCustomSeqEnable**”并填写下述命令索引指定的相应“**lutCustomSeq**”。

对于串行(SPI) NOR，预定义的 LUT 索引如下：

表 6-3. FlexSPI NOR 的查询表索引预分配

名称	查询表中的索引	说明
Read	0	读命令序列
ReadStatus	1	读取状态命令
ReadStatusXpi	2	OPI 模式下的读取状态命令
WriteEnable	3	写启用命令序列
WriteEnableXpi	4	OPI 模式下的写启用命令
EraseSector	5	擦除扇区命令
EraseBlock	8	擦除块命令
PageProgram	9	页面编程命令
ChipErase	11	擦除整个芯片
ExitNoCmd	15	根据需要退出无命令模式
保留	6，7，10，12，13，14	所有保留的索引都可自由用于其他目的

6.2.2 FlexSPI NOR 配置选项块

FlexSPI NOR 配置选项块以 4 位为一个单位排列。该选项块可扩展，其当前定义如下表所示。

闪存加载器使用大多数符合 JESD216(A/B)标准的闪存设备均支持的读取 SFDP 命令来检测 FNORCB。但是，JESD216A/B 仅定义了 Quad SDR 读取的虚拟周期。为了获取 DDR/DTR 读取模式的虚拟周期，闪存加载器支持通过将测试模式写入外部存储器设备上的偏移量 0x200 来进行自动探测。为了获得最佳时序，对于不支持外部提供的 DQS 填充输入的闪存设备，将 readSampleClkSrc 设置为 1。对于支持外部提供的 DQS 填充输入的闪存设备（例如 HyperFlash），将其设置为 3。FlexSPI_DQS 填充不用于任何其他目的。

表 6-4. FlexSPI NOR 配置选项块

<table>
<tr><th>偏移</th><th>字段</th><th>说明</th></tr>
<tr><td>0</td><td>Option0</td><td>
<table>
<tr><th>标签
[31:28]</th><th>选项大小
[27:24]</th><th>设备检测类型
[23:20]</th><th>查询 CMD 填充
[19:16]</th><th>CMD 填充
[15:12]</th><th>Quad 启用类型
[11:8]</th><th>其他
[11:4]</th><th>最大频率
[3:0]</th></tr>
<tr><td>0x0C</td><td>大小（字节）=（选项大小+ 1）* 4</td><td>0 - QuadSPI SDR
1 - QuadSPI DDR
2 - HyperFLASH 1V8
3 - HyperFLASH 3V
4 - MXIC OPI DDR
6 - Micron OPI DDR
8 - Adesto OPI DDR</td><td>0 - 1
2 - 4
3 - 8</td><td>0 - 1
2 - 4
3 - 8</td><td>1 - QE 位是 StatusReg1 中的位 6
2 - QE 位是 StatusReg2 中的位 1
3 - QE 位是 StatusReg2 中的位 7
4 - QE 位是 StatusReg2 中的位 1，启用命令为 0x31</td><td>3 -在 OPI DDR 模式下交换字节顺序</td><td>有关具体器件的更多详细信息，请参见 SoC RM 中的“系统启动”章节。</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Optionl 可选项</td><td>
<table>
<tr><th>flash_connection
[31:28]</th><th>drive_strength
[27:24]</th><th>保留[23:8]</th><th>虚拟周期
[7:0]</th></tr>
<tr><td>0 - PortA
1 -并行模式</td><td>FlexSPI 填充的驱动强度。有关更多详细信息，请参见 SoC RM 中“IOMUXC”章节。</td><td>保留供日后使用</td><td>0 -使用自动探测虚拟周期
其他-数据手册中提供的虚拟周期</td></tr>
</table>
</td></tr>
</table>

- 标签——固定为 0x0C。
- 选项大小——提供可扩展性以供日后使用，选项块大小等于（选项大小+ 1）* 4 字节。

- 设备检测类型——软件定义的设备类型，用于配置块自动检测。
- 查询命令填充——SFDP 命令的命令填充(1/4/8)。
- CMD 填充——闪存设备的命令填充(1/4/8)对于使用 1-1-4、1-4-4、1-1-8 或 1-8-8 模式的设备，CMD 填充值始终为 0x0。对于仅支持 4-4-4 模式以实现高性能的设备，CMD 填充值为 2。对于仅支持 8-8-8 模式以实现高性能的设备，CMD 填充值为 3。
- Quad 启用类型——指定 Quad 启用序列。仅适用于符合 JESD216 标准的设备。如果设备支持 JESD216A 或更高版本，则忽略此字段。
- 其他——为所选闪存类型指定其他模式。
- 最大频率——指定闪存设备的最大频率
- 虚拟周期——用户提供的 SDR/DDR 读命令的虚拟周期

6.2.2.1 FlexSPI NOR 配置块的典型用例

- QuadSPI NOR - Quad SDR Read：option0 = 0xc0000006 (100 MHz)。
- QuadSPI NOR - Quad DDR Read：option0 = 0xc0100003 (60 MHz)。
- HyperFlash 1V8：option0 = 0xc0233007 (133 MHz)。
- HyperFlash 3V0：option0 = 0xc0333006 (100 MHz)。
- HyperFlash 3V0：option0 = 0xc0333006 (100 MHz)。
- MXIC OPI DDR：对于 SDR 模式和 DDR 模式之间数据顺序一致的设备，option0 = 0xc0403006 (100 MHz)；对于 SDR 模式和 DDR 模式之间数据顺序不一致的设备，option0 = 0xc0403036。
- Micron Octal DDR：option0=0xc0600006 (100 MHz)。
- Micron OPI DDR：option0=0xc0603006 (100 MHz)，SPI->OPI DDR。
- Micron OPI DDR（默认启用 DDR 读取）： option0=0xc0633006(100 MHz)。
- Adesto OPI DDR： option0=0xc0803007(133 MHz)。

6.2.2.2 使用 FlexSPI NOR 配置选项块对串行 NOR 闪存设备进行编程

MCU 闪存加载器支持使用 configure-memory 命令生成完整的 FNORCB。它还支持使用特定选项“0xF000000F”将生成的 FNORCB 烧写到闪存的起始位置。下面引用了一个配置和访问 HyperFlash（假设它是空白的 HyperFlash 设备）的示例。

```
blhost -u -- fill-memory 0x2000 0x04 0xc0233007（将选项块写入 SRAM 地址 0x2000）
blhost -u -- configure-memory 0x09 0x2000（使用选项块配置 HyperFLASH）
```

blhost -u -- fill-memory 0x3000 0x04 0xf000000f（将特定选项写入 SRAM 地址 0x3000）
blhost -u -- configure-memory 0x09 0x3000（将 FNORCB 编程为 HyperFLASH 的起始位置）
blhost -u -- write-memory <addr> image.bin

6.3 通过 FlexSPI 支持的串行 NAND 闪存

一些 MCU 支持通过 BootROM 从串行 NAND 闪存设备启动。MCU 闪存加载器用作将启动映像烧写到串行 NAND 中的配套程序。闪存加载器支持生成相应的启动数据结构，例如 BootROM 所需的 FlexSPI NAND 固件配置块(FCB)和发现的坏块表(DBBT)。有关 FlexSPI NAND 启动流程的详细信息，请参见器件参考手册中的“系统启动”章节。本章仅专注于生成 FCB、DBBT，以及使用闪存加载器烧写 FCB、DBBT 和引导映像。

闪存加载器可以使用 FCB 或简化的 FCB 选项块配置串行 NAND 设备。闪存加载器可以基于简化的 FCB 选项块生成完整的 FCB。

6.3.1 FlexSPI NAND 固件配置块(FCB)

FCB 是一个 1024 字节的数据结构，其中包含最适合的 NAND 时序、发现的坏块表(DBBT)搜索区域固件信息（包括起始页地址和页数）的页地址等等。

表 6-5.FlexSPI NAND 固件配置块定义

名称	偏移	大小（字节）	说明
crcChecksum	0x000	4	校验和
fingerprint	0x004	4	0x4E46_4342 ASCII：“NFCB”
版本	0x008	4	0x0000_0001
DBBTSearchStartPage	0x00c	4	坏块表搜索区域的起始页地址
searchStride	0x010	2	DBBT 和 FCB 搜索的搜索步幅 ROM 不使用，最大值在 Fusemap 中定义。有关更多详细信息，请参见 SoC RM 中的“Fusemap”章节。
searchCount	0x012	2	DBBT 和 FCB 上的副本

表格接下页……

表 6-5. FlexSPI NAND 固件配置块定义（续）

名称	偏移	大小（字节）	说明
			ROM 不使用，最大值在 Fusemap 中定义。有关更多详细信息，请参见 SoC RM 中的“Fusemap”章节。
firmwareCopies	0x014	4	固件副本 有效范围为 1-8
保留	0x018	40	保留供日后使用 必须设置为 0
firmwareInfoTable	0x40	64	该表的组成如下（最多 8 个条目）： 字段 | 大小（字节） | 说明 StartPage | 4 | 此固件的起始页 pageCount | 4 | 此固件中的页
保留	0x080	128	保留 必须设置为 0
spiNandConfigBlock	0x100	512	FlexSPI 上的串行 NAND 配置块
保留	0x300	256	保留 必须设置为 0

6.3.2 FlexSPI NAND 配置块

最适合的串行 NAND 参数在 FlexSPI NAND 配置块(FNANDCB)中定义。FNANDCB 是一个 512 字节的数据结构，如下表所示。

表 6-6. FlexSPI NAND 配置块定义

名称	偏移	大小（字节）	说明
memCfg	0x00	480	与 FlexSPI NOR 配置块中前 480 个字节的定义相同

表格接下页……

表 6-6. FlexSPI NAND 配置块定义（续）

名称	偏移	大小（字节）	说明
pageDataSize	0x1c0	480	页面大小（以字节为单位）。通常为 2048 或 4096
pageTotalSize	0x1c4	4	等于列地址的 2^宽度
pagesPerBlock	0x1c8	4	每块页数
bypassReadStatus	0x1cc	1	0 -执行读状态 1 -旁路读状态
bypassEccRead	0x1cd	1	0 -执行 ECC 读 1 -旁路 ECC 读
hasMultiPlanes	0x1ce	1	0 -设备只有 1 个平面 1 -设备有 2 个平面
-	0x1cf	1	保留
eccCheckCustomEnable	0x1d0	1	0 -使用常用的 ECC 校验命令和掩码 1 -使用此配置块中提供的与 ECC 校验相关的掩码
ipCmdSerialClkFreq	0x1d1	1	芯片特定值，设置为 0
readPageTimeUs	0x1d2	2	页读取期间的等待时间，仅在“ bypassReadStatus”置 1 时有效
eccStatusMask	0x1d4	4	ECC 状态掩码，仅在“eccCheckCustomEnable”置 1 时有效
eccFailureMask	0x1d8	4	ECC 校验失败掩码，仅在“eccCheckCustomEnable”置 1 时有效
blocksPerDevice	0x1dc	4	串行 NAND 设备中的块
-	0x1e0	32	保留

注

对于串行(SPI) NAND，预定义的 LUT 索引如下：

表 6-7. FlexSPI 的查询表索引预分配

指令索引	名称	查询表中的索引	说明
0	ReadFromCache	0	从缓存读取

表格接下页……

表 6-7. FlexSPI 的查询表索引预分配（续）

指令索引	名称	查询表中的索引	说明
1	ReadStatus	1	读取状态
2	WriteEnable	3	写启用
3	BlockErase	5	擦除块
4	ProgramLoad	9	程序加载
5	ReadPage	11	读取页面至缓存
6	ReadEccStatus	13	读取 ECC 状态
7	ProgramExecute	14	程序执行
8	ReadFromCacheOdd	4	页面位于奇数平面时从缓存读取
9	ProgramLoadOdd	10	-
-	保留	2，6，7，8，12，15	所有保留的索引都可自由用于其他目的

6.3.3 FlexSPI NAND FCB 选项块

FlexSPI NAND FCB 选项块定义 FCB 所需的主要参数，例如映像信息。详细的配置块定义如下所示。

表 6-8. FlexSPI NAND FCB 选项块

偏移	字段	大小	说明
0	option0	4	偏移 / 字段 / 说明 31:28 / 标签 / 固定为 0x0C 27:24 / searchCount / 有效值：1-4 23:20 / searchStride / 0 - 64 页 1 - 128 页 2 - 256 页 3 - 32 页 注：该字段与熔丝定义一致 19:12 / 保留 / - 11:8 / addressType / 0 -字节地址

表格接下页……

表 6-8. FlexSPI NAND FCB 选项块（续）

<table>
<tr><td></td><td></td><td></td><td><table><tr><td></td><td></td><td>1 -块地址</td></tr><tr><td>7:4</td><td>保留</td><td>-</td></tr><tr><td>3:0</td><td>选项大小</td><td>以长字为单位的选项大小，最小值为 3，最大值为 10</td></tr></table></td></tr>
<tr><td>4</td><td>nandOptionAddr</td><td>4</td><td>上面定义的 NAND 选项的地址</td></tr>
<tr><td>8</td><td>imageInfo</td><td>4-32</td><td>映像信息是以下信息的映射，最大条目大小为 8
<table><tr><th>字段</th><th>大小</th><th>说明</th></tr><tr><td>blockCount</td><td>2</td><td>此映像允许的最大块数</td></tr><tr><td>blockId</td><td>2</td><td>此映像的起始块索引</td></tr></table></td></tr>
</table>

注

- “searchCount”应与 eFUSE 中的配置匹配
- “searchStride”应与 eFUSE 中的配置匹配
- “addressType”指定闪存加载器中擦除、写入和读取操作的起始地址的地址类型
- “选项大小”指定选项块大小的总大小（单位为字长）
- “nandOptionAddr”指定存储 FlexSPI NAND 配置选项的地址
- “imageInfo”是一个数组，其中包含启动期间使用的每个映像信息。例如，0x00040002 表示块 ID 为 4，最大允许块数为 2

6.3.4 FlexSPI NAND 配置选项块

目前，市场上所有串行 NAND 设备都支持相同的命令。不同之处在于 NAND 大小和页面大小等。该选项块主要是针对这些差异，详细的块定义如下所示：

表 6-9. FlexSPI NAND 配置选项块

<table>
<tr><th>偏移</th><th>字段</th><th>说明</th></tr>
<tr><td>0</td><td>option0</td><td>
<table>
<tr><th>标签
[31:28]</th><th>选项大小
[27:24]</th><th>保留
[23:20]</th><th>闪存大小
[19:16]</th><th>具有多个平面
[15:12]</th><th>每块页数
[11:8]</th><th>页面大小 (Kbytes)
[7:4]</th><th>最大频率
[3:0]</th></tr>
<tr><td>0x0c</td><td>大小（字节）=（选项大小+ 1）* 4</td><td>0</td><td>0 - 512Mb
1 - 1Gb
2 - 2Gb
4 - 4Gb</td><td>0 - 1 个平面
1 - 2 个平面</td><td>0 - 64
1 - 128
2 - 256
3 - 32</td><td>2 - 2KB
4 - 4KB</td><td>NAND
频率：器件特定</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>option1</td><td>此字段为可选字段，如果 option0 中的选项大小大于 0，则此字段有效
<table>
<tr><th>保留[31:8]</th><th>制造商 ID [7:0]</th></tr>
<tr><td>保留供日后使用</td><td>串行 NAND 器件数据手册中提供的实际制造商 ID
例如，分配给 Micron 的制造 ID 为 0x2C</td></tr>
</table>
</td></tr>
</table>

6.3.5 闪存加载器的用法示例

闪存加载器可以基于指定的 FlexSPI NAND FCB 选项块生成 FCB 和 DBBT。

假设 FCB 参数为：

- FCB 和 DBBT 副本为 2。
- 固件副本为 2。
- 固件 0 从块 4 开始，最大块数为 2。
- 固件 1 从块 8 开始，最大块数为 2。

假设串行 NAND 参数为：

- 闪存大小：1 Gb
- 平面编号：1
- 每块页数：64

- 页面大小：2 KB
- 最大频率：80 MHz

以下是生成 FlexSPI NAND 配置选项块的示例步骤。

将 FlexSPI NAND 配置选项块写入 SRAM：

```
blhost -u -- fill-memory 0x2030 0x4 0xc0010025
```

将 FlexSPI NAND FCB 选项块写入 SRAM：

```
blhost -u -- fill-memory 0x2000 0x4 0xc2000104
blhost -u -- fill-memory 0x2004 0x4 0x2030 // nandOptionAddr = 0x2030
blhost -u -- fill-memory 0x2008 0x4 0x00040002 // blockId = 4，blockCount = 2
blhost -u -- fill-memory 0x200c 0x4 0x00080002 // blockId = 8，blockCount = 2
```

使用 FCB 选项和 NAND 选项配置串行 NAND：

```
blhost -u -- configure-memory 0x101 0x2000
```

擦除和烧写映像

```
blhost -u -- flash-erase-region 0x4 0x2 0x101 //从块 4 开始擦除 2 个块
blhost -u -- write-memory 0x4 image.bin 0x101 //将 image.bin 烧写到块 4
blhost -u -- flash-erase-region 0x8 0x2 0x101 //从块 8 开始擦除 2 个块
blhost -u -- write-memory 0x8 image.bin 0x101 //将 image.bin 烧写到块 8
```

6.4 通过 uSDHC 支持的 SD/eMMC

一些 MCU 支持通过 BootROM 从 SD/eMMC 设备启动。MCU 闪存加载器支持将启动映像烧录到 SD/eMMC 设备中。本小节介绍通过闪存加载器使用 SD/eMMC 的方法。

6.4.1 SD 配置块

必须先初始化 SD 卡，然后闪存加载器才能访问 SD 存储器。SD 配置块是闪存加载器用于初始化卡的若干必要 SD 配置的组合。

表 6-10 列出了 SD 配置块中所有位的详细说明。

表 6-10. SD 配置块定义

字索引	位字段	名称	说明
Word0	[31:28]	标签	SD 配置块标签，用于标记该块是否有效 0xD：有效块 其他：无效
	[27:26]	RSV	0x0
	[25:24]	PWR_DOWN_TIME	SD 卡上电之前的 SD 掉电延迟时间 仅在 PWR_CYCLE_ENABLE 启用时有效 0：20 ms 1：10 ms 2：5 ms 3：2.5 ms
	23	PWR_POLARITY	SD 电源控制极性 仅在 PWR_CYCLE_ENABLE 启用时有效 0：当 uSDHC.RST 置为低电平时掉电 1：当 uSDHC.RST 置为高电平时掉电
	[22:21]	RSV	0x0
	20	PWR_UP_TIME	SD 等待稳压器输出稳定的上电延迟时间 仅在 PWR_CYCLE_ENABLE 启用时有效 0：5 ms 1：2.5 ms
	19	PWR_CYCLE_ENABLE	在初始化过程开始之前执行电源循环[1] 0：禁止（适合非 UHSI 时序[2]） 启用（适合 UHSI 时序） 1：启用
	[18:15]	RSV	0x0
	[14:12]	TIMING_INTERFACE	SD 速度时序选择。 0：正常/SDR12 1：高/SDR25

表格接下页……

表 6-10. SD 配置块定义（续）

			2：SDR50 3：SDR104 4：DDR50（尚不支持） 5-7：保留
	[11:9]	RSV	0x0
	8	BUS_WIDTH	SD 总线带宽选择。 0：1 位 4 位（适用于 UHSI 时序） 1：4 位
	[7:0]	RSV	0x0
Wordl	[31:0]	RSV	0x0

注

闪存加载器切换 uSDHC.RST 引脚以执行电源循环过程。这需要板级硬件支持。如果硬件不支持 SD 电源控制，则电源循环过程无法完全复位 SD 卡。

注

UHSI 时序包括 SDR50、SDR104 和 DDR50。

6.4.2 闪存加载器的用法示例

本小节以 SDR25 时序和 4 位总线宽度为例。为确保在初始化之前复位 SD 卡，建议启用电源循环。选择电源循环的默认设置。

SD 配置块的十六进制为 0xD0082100。

- 将配置块写入 MCU 的内部 RAM。

  blhost -u -- fill-memory 0x20000000 0x4 0xD0082100

  选择 RAM 地址 0x20000000 作为示例。用户可以选择任何可用的 RAM 位置。用户还可以选择位于 XIP 外部存储器的地址，例如 Flex SPI NOR 闪存。

- 使用 configure-memory 命令执行初始化过程。

  blhost -u -- configure-memory 0x120 0x20000000

0x120 是 eMMC 卡设备的存储器 ID。如果已成功初始化 eMMC 卡，则将收到“成功”消息，并且 SD 存储器可供闪存加载器访问。如果发生错误，有关调试信息请参见*第 8 章附录 A “状态和错误码”*。

- SD 卡初始化后，用户可以使用“获取属性 25”(get the property 25)命令来检查 SD 卡的容量。

```
blhost -u -- get-property 25 0x120
```

- 要烧写启动映像，用户需要先擦除 SD 卡存储器，然后才能烧写映像。

```
blhost -u -- flash-erase-region 0x0 0x1000 0x120
blhost -u -- write-memory 0x400 C:\Image\bootImage.bin 0x120
```

flash-erase-region 命令行中的 0x0 和写存储器命令行中的 0x400 是 SD 存储器的字节偏移量，而不是扇区偏移量。这意味着将擦除从 SD 存储器起始地址开始的 4 K 字节，然后将启动映像 *C:\Image\bootImage.bin* 写入从 SD 第二个块开始的空间。

- 要检查启动映像是否烧写成功，用户可以将数据读出。

```
blhost -u -- read-memory 0x400 0x1000 0x120
```

在大多数情况下，用户无需读出数据即可验证启动映像是否成功写入。闪存加载器可对此提供保证。

6.4.3 eMMC 配置块

与 SD 卡类似，在访问 eMMC 之前也必须对其进行初始化。eMMC 配置块用于告诉闪存加载器如何初始化 eMMC 设备。要使用 BootROM 提供的快速启动功能，还必须预先配置 eMMC。快速启动配置也包含在 eMMC 配置块中。

表 8-11 列出了 eMMC 配置块中所有位的详细说明。

表 6-11. eMMC 配置块定义

字索引	位字段	名称	说明
Word0	[31:28]	标签	eMMC 配置块标签，用于标记该块是否有效

表格接下页……

表 6-11. eMMC 配置块定义（续）

			0xC：有效块 其他：无效
	27	RSV	0x0
	[26:24]	PARTITION_ACCESS	选择闪存加载器要向其中写入映像或数据的 eMMC 分区 0：用户数据区 1：启动分区 1 2：启动分区 2 3：RPMB 4：通用分区 1 5：通用分区 2 6：通用分区 3 7：通用分区 4
	23	RSV	0x0
	[22:20]	BOOT_PARTITION_ENABLE	选择用于快速启动的启动分区 仅在 BOOT_CONFIG_ENABLE 置位时有效 0：未启用 1：启动分区 1 2：启动分区 2 3-6：保留 7：用户数据区
	[19:18]	RSV	0x0
	[17:16]	BOOT_BUS_WIDTH	选择用于快速启动的总线带宽 0：x1(SDR)， x4(DDR) 1：x4(SDR,DDR) 2：x8(SDR,DDR) 3：保留
	[15:12]	TIMING_INTERFACE	选择闪存加载器访问 eMMC 存储器时的总线时序 0：正常 1：HS 2：HS200（尚不支持） 3：HS400（尚不支持） 4-15：保留

表格接下页……

表 6-11. eMMC 配置块定义（续）

	[11:8]	BUS_WIDTH	选择闪存加载器访问 eMMC 存储器时的总线带宽 0：x1 SDR 1：x4 SDR 2：x8 SDR 3-4：保留 5：x4 DDR 6：x8 DDR 7-15：保留
	[7:6]	RSV	0x0
	[5:4]	BOOT_MODE	0：正常 1：HS 2：DDR 3：保留
	3	RESET_BOOT_BUS_CONDITIONS	退出快速启动后配置 eMMC 行为 0：复位为 x1，SDR，正常 1：保留启动配置
	2	BOOT_ACK	在快速启动时配置 eMMC ACK 行为 0：NO ACK 1：ACK
	1	RSV	0x0
	0	BOOT_CONFIG_ENABLE	确定是否将快速启动配置写入 eMMC [2] 0：忽略启动配置 1：将启动配置写入设备
Word1	[31:26]	RSV	0x0
	[25:24]	PWR_DOWN_TIME	eMMC 卡上电之前的 eMMC 掉电延迟时间 仅在 PWR_CYCLE_ENABLE 启用时有效 0：20 ms 1：10 ms 2：5 ms 3：2.5 ms
	23	PWR_POLARITY	eMMC 电源控制极性。

表格接下页……

表 6-11. eMMC 配置块定义（续）

			仅在 PWR_CYCLE_ENABLE 启用时有效 0：当 uSDHC.RST 置为低电平时掉电 1：当 uSDHC.RST 置为高电平时掉电
	[22:21]	RSV	0x0
	20	PWR_UP_TIME	eMMC 等待稳压器输出稳定的上电延迟时间 仅在 PWR_CYCLE_ENABLE 启用时有效 0：5 ms 1：2.5 ms
	19	PWR_CYCLE_ENABLE	在 SD 初始化过程开始之前执行电源循环 0：禁用 1：启用
	18	1V8_ENABLE	选择是否设置 uSDHC.VSELECT 引脚 0：不设置 vselect 引脚 1：将 vselect 引脚设置为高电平
	[17:0]	RSV	0x0

注

快速启动配置包括 BOOT_PARTITION_ENABLE、BOOT_BUS_WIDTH、BOOT_MODE、RESET_BOOT_BUS_CONDITIONS 和 BOOT ACK。

6.4.4 闪存加载器的用法示例

本小节以 8 位 DDR 模式为例，说明如何将启动映像写入用户数据区。写入启动映像后，用户希望启动 ROM 能够通过快速启动来启动映像，以减少启动时间。快速启动也使用相同的模式（8 位 DDR 模式）。启用 ACK 以支持快速启动。

eMMC 配置块的十六进制为 0xC0721625，0x00000000

- 将配置块写入 MCU 的内部 RAM。

```
blhost -u -- fill-memory 0x20000000 0x4 0xC0721625
blhost -u -- fill-memory 0x20000004 0x4 0x00000000
```

选择 RAM 地址 0x20000000 作为示例。用户可以选择任何可用的 RAM 位置。用户还可以选择位于 XIP 外部存储器的地址，例如 Flex SPI NOR 闪存。

- 使用 configure-memory 命令执行初始化过程。

```
blhost -u -- configure-memory 0x121 0x20000000
```

0x121 是 eMMC 卡设备的存储器 ID。如果已成功初始化 eMMC 卡，则将收到“成功”消息。如果发生错误，有关调试信息请参见*第 8 章“附录 A：状态和错误码”*。

- 执行了第 2 步之后，即可访问 eMMC。用户可以使用“获取属性 25”(get the property 25)命令来检查 eMMC 卡的容量。 blhost -u -- get-property 25 0x121

- 要烧写启动映像，用户需要先擦除 eMMC 卡存储器，然后才能烧写映像。

```
blhost -u -- flash-erase-region 0x0 0x2000 0x121
blhost -u -- write-memory 0x400 C:\Image\bootImage.bin 0x121
```

命令行中 eMMC 存储器的地址是字节地址，而不是扇区地址。这意味着将擦除从 eMMC 存储器起始地址开始的 8 K 字节，然后将启动映像 *C:\Image\bootImage.bin* 写入 eMMC 第一个块。

- 要检查启动映像是否烧写成功，用户可以将数据读出。

```
blhost -u -- read-memory 0x200 0x2000 0x121
```

在大多数情况下，用户无需读出数据即可验证启动映像是否成功写入。当用户获得写入存储器命令的“成功”状态时，闪存加载器可对此提供保证。

如果用户要切换到 eMMC 设备的其他分区，则需要重新配置 eMMC 设备两次。

- 选择启动分区 1，总线带宽和速度时序保持不变。如果用户不打算更新，则无需快速启动配置。

```
blhost -u -- fill-memory 0x20000000 0x4 0xC1001600
blhost -u -- configure-memory 0x121 0x20000000
blhost -u -- flash-erase-region 0x0 0x1000 0x121
blhost -u -- write-memory 0x400 C:\Image\bootPartitionOneImage.bin 0x121
```

第 7 章
安全实用程序

7.1 简介

MCU 闪存加载器支持某些安全实用程序，用于轻松生成与安全性相关的块。请注意，必须首先对闪存加载器本身进行签名才能正确启用安全实用程序。

7.2 映像加密和烧写

对于具有 BEE 模块的设备，它使用两个唯一的加密密钥，分别支持两个加密区域。每个加密区域最多可以支持 3 个 FAC 细分区域。有关映像解密及其所需数据结构的详细信息，请参见 SoC Rm 的“系统启动”章节。本小节将重点介绍如何使用闪存加载器生成和烧写加密映像。闪存加载器基于简化的 PRDB 选项块生成加密映像，具体定义如下所述。

注

闪存加载器仅支持使用 OTPMK/SNVS 密钥对第一个加密区域进行映像加密和烧写。

表 7-1. PRDB 选项块

<table>
<tr><th>偏移</th><th>字段</th><th>大小（字节）</th><th>说明</th></tr>
<tr><td>0</td><td>选项</td><td>4</td><td>
<table>
<tr><th>标签 [31:28]</th><th>密钥源 [27:24]</th><th>模式 [23:20]</th><th>FAC 区域数 [19:16]</th><th>区域 1 保护模式 [15:12]</th><th>区域 2 保护模式 [11:8]</th><th>区域 3 保护模式 [7:4]</th><th>锁定选项 [3:0]</th></tr>
<tr><td>0xE</td><td>0 - OTPMK/SNVS</td><td>1 - AES CTR</td><td>1/2/3</td><td>0/1/</td><td>0/1</td><td>0/1</td><td>0 = 不锁定</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>FAC 区域信息</td><td>8-24</td><td>
<table>
<tr><th>偏移</th><th>字段</th><th>说明</th></tr>
<tr><td>0</td><td>起始</td><td>FAC 区域开始</td></tr>
<tr><td>4</td><td>大小</td><td>FAC 区域大小</td></tr>
</table>
</td></tr>
</table>

注

- 标签固定为 0x0E。
- 密钥源可以是 OTPMK/SNVS [255:128]。
- 模式：建议使用 AES-CTR 模式。
- FAC 区域数：允许的最大 FAC 区域数为 3（由加密区域 0 和加密区域 1 共享）。
- 区域 n 保护模式：0 - 无保护，1 - 禁用调试。
- 锁定选项：必须为 0。

7.2.1 生成加密映像并编程到闪存的示例

以 HyperFlash 为例，假设加密信息为：

- 密钥源：OTPMK/SNVS [255:128]。
- FAC 区域数：2.
- 区域保护模式：1.

创建 PRDB 选项块的步骤如下所述。

使用 FlexSPI NOR 配置选项块配置 HyperFlash：

```
blhost -u -- fill-memory 0x2000 0x04 0xc0233007 //(133MHz)
blhost -u -- configure-memory 0x9 0x2000
blhost -u -- fill-memory 0x3000 0x04 0xf000000f
blhost -u -- configure-memory 0x09 0x3000
```

使用 PRDB 选项块准备 PRDB0 信息：

```
blhost -u -- fill-memory 0x4000 0x04 0xe0121100
blhost -u -- configure-memory 0x09 0x4000
//烧写 HyperFLASH
blhost -u -- write-memory <addr> image.bin
```

7.3 KeyBlob 生成和烧写

7.3.1 KeyBlob

KeyBlob 是一种将 DEK 打包的数据结构，以使用 AES-CCM 算法解密映像。整个 KeyBlob 数据结构如下所示。

表 7-2. KeyBlob 数据结构

字段	大小（字节）	说明		
标头	4	**偏移**	**字段**	**说明**
		0	标签	固定值：0x81
		1-2	长度	KeyBlob 块的长度，16 位大端顺序
		3	参数	KeyBlob 版本，设置为 0x42 或 0x43
AEAD	4	**偏移**	**字段**	**说明**
		0	模式	固定为 0x66，CCM 模式
		1	算法	固定为 0x55，加密算法：AES
		2	mac_bytes	固定为 16
		3	aad_bytes	固定为 0

表格接下页……

表 7-2. KeyBlob 数据结构（续）

字段	大小（字节）	说明
EBK	16/32	Blob 密钥用于 DEK 加密，它是 TRNG 引擎生成的随机数 Blob 密钥通过由安全引擎（例如 SNVS 或 CAAM）派生的密钥加密为 EBK
EDEK	16/24/32	DEK 用于启动映像加密，它由 BK 使用 AES-CCM 模式的 AES 算法通过 BK 加密为 EDEK。
MAC	16	在 DEK 加密期间生成 MAC

7.3.2 KeyBlob 选项块

MCU 闪存加载器使用称为 KeyBlob 选项块的简化选项块来支持 KeyBlob 的生成和烧写。

表 7-3. KeyBlob 数据结构

偏移	字段	大小	说明
0	选项	4	偏移 / 字段 / 说明： 31:28 / 标签 / 固定为 0x0B 27:24 / 类型 / 0 -更新，用于更新 keyblob 上下文 1 -烧写，用于通知存储器驱动程序将 Keyblob 烧写到目标位置 23:20 / 大小 / keyblob_info 大小 如果 type = 0，则必须等于 3；如果 type = 1，则忽略 19:8 / 保留 / - 7:4 / dek_size / DEK 大小 0 - 128 位 1 - 192 位 2 - 256 位

表格接下页……

表 7-3. KeyBlob 数据结构（续）

<table>
<tr><th>偏移</th><th>字段</th><th>大小</th><th>说明</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>偏移</th><th>字段</th><th>说明</th></tr>
<tr><td></td><td></td><td>如果 type = 0，则有效；如果 type = 1，则忽略</td></tr>
<tr><td>3:0</td><td>image_index</td><td>启动映像索引
例如，firmwareTable 的索引
如果 type = 1，则有效；如果 type = 0，则忽略</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>dek_addr</td><td>4</td><td>存放 DEK 的存储器起始地址</td></tr>
<tr><td>2</td><td>keyblob_offset</td><td>4</td><td>所选映像中的 Keyblob 相对偏移
例如，包含 IVT、加密的应用程序、CSF、密钥 blob 的签名映像
IVT 的偏移为 0x400
加密映像的偏移为 0x2000
CSF 的偏移为 0xA000
KeyBlob 的偏移为 0xB000
注： 对于 NAND 设备，必须页对齐 keyblob_offset</td></tr>
</table>

7.3.3 生成和烧写 KeyBlob 的示例

生成 KeyBlob

//将 DEK 写入 RAM

blhost -u -- write-memory 0x2100 dek.bin

//构造 KeyBlob 选项

blhost -u -- fill-memory 0x2080 4 0xb0300000 // tag = 0x0b, type = 0, size = 3, dek_size = 0 (128bits)
blhost -u -- fill-memory 0x2084 4 0x2100 // dek_addr = 0x2100
blhost -u -- fill-memory 02088 4 0x80000 // keyblob_offset = 0x80000，keyblob 位于应用程序映像中的偏移 0x80000 处

//更新 KeyBlob 信息

KeyBlob 生成和烧写

blhost -u -- configure-memory 0x101 0x2080 //更新 KeyBlob 信息（存储器 id：0x101 - FlexSPI NAND）

//烧写 KeyBlob

blhost -u -- fill-memory 0x2080 0xb1000000　　// tag = 0x0b，type = 1，image_index = 0
blhost -u -- configure-memory 0x101 0x2080 //生成 KeyBlob 并将其烧写至
所选映像<image_idex>存储器区域中的偏移<keyblob_offset>处

第 8 章
附录 A：状态和错误码

状态和错误码按组件分组。定义错误的每个组件都有一个组编号。下述表达式用于构造状态码值。

status_code =（组别号 * 100）+ 代码

下表列出了组件的组编号。

表 8-1. 组件组编号

组	组件
0	一般错误
100	引导加载程序
101	SB 加载程序
102	存储器接口
103	属性存储
104	CRC 校验器
105	分包器
106	可靠的更新

下表列出了所有错误和状态码。

表 8-2. 错误和状态码

名称	值	说明
kStatus_Success	0	操作成功，无错误
kStatus_Fail	1	操作失败，发生一般错误
kStatus_ReadOnly	2	无法更改属性，因为它是只读的
kStatus_OutOfRange	3	请求值超出范围
kStatus_InvalidArgument	4	请求命令的参数未定义
kStatus_Timeout	5	发生超时
kStatus_NoTransferInProgress	6	当前传输状态为空闲
kStatus_SDMMC_NotSupportYet	1800	不支持此功能
kStatus_SDMMC_TransferFailed	1801	与设备通信失败

表格接下页……

表 8-2. 错误和状态码（续）

名称	值	说明
kStatus_SDMMC_SetCardBlockSizeFailed	1802	设置块大小失败
kStatus_SDMMC_HostNotSupport	1803	主机不支持此功能
kStatus_SDMMC_CardNotSupport	1804	该卡不支持此功能
kStatus_SDMMC_AllSendCidFailed	1805	发送 CID 失败
kStatus_SDMMC_SendRelativeAddressFailed	1806	发送相对地址失败
kStatus_SDMMC_SendCsdFailed	1807	发送 CSD 失败
kStatus_SDMMC_SelectCardFailed	1808	选择卡失败
kStatus_SDMMC_SendScrFailed	1809	发送 SCR 失败
kStatus_SDMMC_SetDataBusWidthFailed	1810	设置总线带宽失败
kStatus_SDMMC_GoIdleFailed	1811	进入空闲状态失败
kStatus_SDMMC_HandShakeOperationConditionFailed	1812	发送操作条件失败
kStatus_SDMMC_SendApplicationCommandFailed	1813	发送应用程序命令失败
kStatus_SDMMC_SwitchFailed	1814	切换命令失败
kStatus_SDMMC_StopTransmissionFailed	1815	停止传输失败
kStatus_SDMMC_WaitWriteCompleteFailed	1816	等待写入完成失败
kStatus_SDMMC_SetBlockCountFailed	1817	设置块数失败
kStatus_SDMMC_SetRelativeAddressFailed	1818	设置相对地址失败
kStatus_SDMMC_SwitchBusTimingFailed	1819	高速切换失败
kStatus_SDMMC_SendExtendedCsdFailed	1820	发送 EXT_CSD 失败
kStatus_SDMMC_ConfigureBootFailed	1821	配置启动失败
kStatus_SDMMC_ConfigureExtendedCsdFailed	1822	配置 EXT_CSD 失败
kStatus_SDMMC_EnableHighCapacityEraseFailed	1823	启用大容量擦除失败
kStatus_SDMMC_SendTestPatternFailed	1824	发送测试模式失败
kStatus_SDMMC_ReceiveTestPatternFailed	1825	接收测试模式失败
kStatus_SDMMC_InvalidVoltage	1829	无效电压
kStatus_SDMMC_TuningFail	1833	调整失败
kStatus_SDMMC_SwitchVoltageFail	1834	切换电压失败
kStatus_SDMMC_SetPowerClassFail	1837	设置电源类别失败
kStatus_UnknownCommand	10000	请求命令值未定义

表格接下页……

表 8-2. 错误和状态码（续）

名称	值	说明
kStatus_SecurityViolation	10001	由于启用了闪存安全性，因此不允许使用该命令
kStatus_AbortDataPhase	10002	提前中止数据阶段
kStatus_Ping	10003	内部：在命令阶段收到 ping
kStatus_NoResponse	10004	该命令无响应
kStatus_NoResponseExpected	10005	无预期的命令响应
kStatusRomLdrSectionOverrun	10100	ROM SB 加载程序扇区超限
kStatusRomLdrSignature	10101	ROM SB 加载程序签名错误
kStatusRomLdrSectionLength	10102	ROM SB 加载程序扇区长度错误
kStatusRomLdrUnencryptedOnly	10103	ROM SB 加载程序不支持纯文本映像
kStatusRomLdrEOFReached	10104	ROM SB 加载程序已达 EOF
kStatusRomLdrChecksum	10105	ROM SB 加载程序校验和错误
kStatusRomLdrCrc32Error	10106	ROM SB 加载程序 CRC32 错误
kStatusRomLdrUnknownCommand	10107	ROM SB 加载程序未知命令
kStatusRomLdrIdNotFound	10108	找不到 ROM SB 加载程序 ID
kStatusRomLdrDataUnderrun	10109	ROM SB 加载程序数据不足
kStatusRomLdrJumpReturned	10110	发生 ROM SB 加载程序从跳转命令返回
kStatusRomLdrCallFailed	10111	ROM SB 加载程序调用命令失败
kStatusRomLdrKeyNotFound	10112	找不到 ROM SB 加载程序密钥
kStatusRomLdrSecureOnly	10113	仅保护 ROM SB 加载程序的安全状态
kStatusRomLdrResetReturned	10114	发生 ROM SB 加载程序从复位返回
kStatusMemoryRangeInvalid	10200	存储器地址范围与受保护区域冲突
kStatusMemoryReadFailed	10201	无法读取存储器地址范围
kStatusMemoryWriteFailed	10202	无法写入存储器地址范围
StatusMemoryCumulativeWrite	10203	无法写入未擦除的存储器地址范围
kStatusMemoryAppOverlapWithExecuteOnlyRegion	10204	存储器地址范围包含一个受保护的仅执行区域
kStatusMemoryNotConfigured	10205	无法访问未配置的外部存储器
kStatusMemoryAlignmentError	10206	地址对齐错误
kStatusMemoryVerifyFailed	10207	验证写入操作失败
kStatusMemoryWriteProtected	10208	存储器地址范围包含一个受保护的存储器区域
kStatus_UnknownProperty	10300	请求属性值未定义
kStatus_ReadOnlyProperty	10301	无法写入请求属性值
kStatus_InvalidPropertyValue	10302	指定的属性值无效
kStatus_AppCrcCheckPassed	10400	CRC 校验通过
kStatus_AppCrcCheckFailed	10401	CRC 校验失败
kStatus_AppCrcCheckInactive	10402	未启用 CRC 校验器
kStatus_AppCrcCheckInvalid	10403	无效 CRC 校验器
kStatus_AppCrcCheckOutOfRange	10404	CRC 校验有效，但地址超出范围
kStatus_NoPingResponse	10500	分包器未收到对 ping 数据包的任何响应

表格接下页……

表 8-2. 错误和状态码（续）

名称	值	说明
kStatus_InvalidPacketType	10501	数据包类型无效
kStatus_InvalidCRC	10502	数据包中 CRC 无效
kStatus_NoCommandResponse	10503	未接收到命令响应
kStatus_ReliableUpdateSuccess	10600	成功完成可靠的更新过程
kStatus_ReliableUpdateFail	10601	可靠的更新过程失败
kStatus_ReliableUpdateInacive	10602	可靠的更新功能无效
kStatus_ReliableUpdateBackupApplicationInvalid	10603	备用应用程序映像无效
kStatus_ReliableUpdateStillInMainApplication	10604	下次启动仍将使用主应用程序映像
kStatus_ReliableUpdateSwapSystemNotReady	10605	默认情况下无法交换闪存，因为交换系统尚未就绪
kStatus_ReliableUpdateBackupBootloaderNotReady	10606	由于没有有效的备份引导加载程序映像，因此无法交换闪存
kStatus_ReliableUpdateSwapIndicatorAddressInvalid	10607	由于提供的交换指示器无效，因此无法交换闪存

第 9 章
附录 B：GetProperty 和 SetProperty 命令

属性是可以使用GetProperty或SetProperty命令访问的已定义数据单元。属性可以是只读，也可以是读写。所有读写属性都是 32 位整数，因此可以在命令参数中轻松加载。并非所有属性在所有平台上都可用。如果属性不可用，GetProperty 和 SetProperty 将返回kStatus_UnknownProperty。

GetProperty 和 SetProperty 命令使用下表所示的标签值，以查询有关闪存加载器的信息。

表 9-1. GetProperty 和 SetProperty 标签值

名称	可写	标签值	大小	说明
CurrentVersion	否	0x01	4	闪存加载器当前版本
AvailablePeripherals	否	0x02	4	该芯片支持的外设集
FlashStartAddress	否	0x03	4	闪存烧写的起始地址
FlashSizeInBytes	否	0x04	4	闪存烧写的大小（以字节为单位）。
FlashSectorSize	否	0x05	4	一个闪存烧写扇区的大小（以字节为单位），这是最小擦除大小
FlashBlockCount	否	0x06	4	闪存阵列中的块数
AvailableCommands	否	0x07	4	闪存加载器支持的命令集
CRCCheckStatus	否	0x08	4	应用程序 CRC 校验的状态
保留	不适用	0x09	不适用	

表格接下页……

表 9-1. GetProperty 和 SetProperty 标签值（续）

名称	可写	标签值	大小	说明
VerifyWrites	是	0x0a	4	控制引导加载程序是否验证对闪存的写入。默认情况下启用 VerifyWrites 功能。 0 - 未验证 1 - 启用验证
MaxPacketSize	否	0x0b	4	当前活动的外设接口支持的最大数据包大小
ReservedRegions	否	0x0c	n	闪存加载器保留的存储器区域列表。返回格式为数值对(<区域起始地址>,<区域结束地址>) • 如果未设置 HasDataPhase 标志，则响应数据包参数计数指示对数 • 如果设置了 HasDataPhase 标志，则第二个参数是数据阶段中的字节数
RAMStartAddress	否	0x0e	4	RAM 的起始地址
RAMSizeInBytes	否	0x0f	4	RAM 的大小（以字节为单位）
SystemDeviceId	否	0x10	4	Kinetis 系统器件标识寄存器的值
FlashSecurityState	否	0x11	4	指示是否启用闪存安全性 0 - 禁用闪存安全性 1 - 启用闪存安全性
UniqueDeviceId	否	0x12	n	唯一器件标识符，唯一标识寄存器的值（K 系列器件为 16，KL 系列器件为 12）

表格接下页……

表 9-1. GetProperty 和 SetProperty 标签值（续）

名称	可写	标签值	大小	说明
FlashFacSupport	否	0x13	4	FAC（闪存访问控制）支持标志 0 - 不支持 FAC 1 - 支持 FAC
FlashAccessSegmentSize	否	0x14	4	1 个闪存扇区的大小（以字节为单位）
FlashAccessSegmentCount	否	0x15	4	FAC 扇区数（闪存模型内的闪存访问扇区数）
FlashReadMargin	是	0x16	4	闪存擦除和烧写验证命令的容限级别设置 0=正常 1=用户 2=工厂
QspiInitStatus	否	0x17	4	QSPI 或 OTFAD 初始化过程的结果 405 - QSPI 未初始化 0 - QSPI 已初始化
TargetVersion	否	0x18	4	目标机版本号
ExternalMemoryAttributes	否	0x19	24	指定的存储器 ID（0 = 内部闪存，1 = QuadSpi0）支持的属性列表，请参见本小节中有关返回值的说明 ExternalMemoryAttributes Property

第 10 章
修订记录

10.1 修订记录

下表显示了本文档的修订记录。

表 10-1. 修订记录

修订版本号	日期	重要变化
0	2016 年 4 月	Kinetis 引导加载程序版本 v2.0.0
1	2017 年 10 月	更新了 i.MX RT 系列器件的闪存加载器应用程序
2	2018 年 1 月	更新了仅符合 JESD216 标准的 QuadSPI NOR 闪存设备的闪存加载器应用程序
3	2018 年 5 月	MCU 引导加载程序版本 v2.5.0
4	2018 年 8 月	RT1060 更新

如何联系我们：

主页：

nxp.com

网络支持：

nxp.com/support

文档编号：MCUFLSHLDRRM
第 4 版，2018 年 8 月