MS60F3xxx

**Bootloader(IAP)**应用 说明

## 概述

IAP 是用户自己的程序在运行过程中对程序 Flash 部分区域进行烧写，目的是为了在产品发布后可以方便地通过预留的通信口对产品中的固件程序进行更新升级。

通常在用户需要实现 IAP 功能时，即用户程序运行中作自身的更新操作，需要在设计固件程序时编写两个项目代码，第一个项目程序（Bootloader 区）不执行用户应用功能，而只是通过 I2C 通信管道接收数据，对第二部分（APP 区）用户应用代码更新；第二个项目代码（APP区）才是真正的用户应用功能代码。这两部分项目代码都同时烧录在程序 Flash 区中，当芯片上电后，首先是第一个项目代码开始运行，它作如下操作：

* 1. 检查是否需要对第二部分代码进行更新。
  2. 如果不需要更新则转到下面第 4 步操作。
  3. 执行更新操作。
  4. 跳转到第二部分代码执行。

第一部分代码必须通过 SWD 烧入（出厂前已烧入，用户无需自行烧入）；第二部分代码可以使用第一部分代码 Bootloader 功能烧入，也可以和第一部分代码一起烧入（用户需提供 APP 程序区的 hex 文件）， 以后需要程序更新是再通过第一部分 Bootloader 代码更新。

第一部分代码处于程序 Flash 的开始 4k 字节区域，而第二部分代码紧随其后。由于我们的 Bootloader 程序不使用中断处理，用户程序不需要做中断向量表映射等操作.

## Bootloader 数据传送格式

主机一帧数据传送格式（具体到不同的指令会有细微变化）：

0x7C： 数据包头标志

0x7E： 数据包尾标志

数据长度: 数据长度可以为 0~0xFF。（注意，真实数据长度为”数据长度+1”，例如，数据长度为 0xFF，实际长度为 256）。

数据： 根据数据长度填入相应的数据。

校验码： 校验码校验范围从包头标志后一个字节到 Checksum 前一个字节（计算时去掉本身以及包头包尾）。校验方法：数据做累加，最后得到的一个字节的数据即为校验码

从机一帧数据传送格式（具体到不用的指令会有细微变化）：

0x7C+“指令”+”指令反码”+”执行结果”+”数据” +”数据” +”数据” +”数据”+”校验码”+0x7E

执行结果：0x55:执行成功；0xAA：执行失败数据： 从机固定发送 4 个字节的信息

校验码： 校验码校验范围从包头标志后一个字节到 Checksum 前一个字节（计算时去掉本身以及包头包尾）。校验方法：数据做累加，最后得到的一个字节的数据即为校验码

“数据”格式：

1. 读取 Bootloader 程序版本号以及 APP 程序区数据 Checksum(指令码 0x01)

命令作用：1.可以用来判断主机和从机是否连接成功

* + 1. 可以读出芯片Bootloader 版本号
    2. 可以读出当前APP 程序数据 Checksum

主机格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令反码 | 校验码 | 包尾 |
| 0x7C | 0x01 | 0xFE | 0xFF | 0x7E |

注意：主机命令如果不需要发送数据信息，可以去掉”数据长度”+”数据”。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令  码反码 | 执行结果 | 是否存在  APP 程序 | APP 程序数据校验码高  8 位 | APP 校  验码低 8  位 | Boot 版本号 | 校验码 | 包尾 |
| 0x7C | 0x01 | 0xFE | 0x55/0xAA | 0x55/0xAA | 0xXX | 0xXX | 0xXX | 0xXX | 0x7E |

从机格式：

注意：本协议中，一般 0x55 代表成功，0xAA 代表失败。

1. 关闭 Flash 寄存器写保护(指令码 0x02)

命令作用：操作 Flash 寄存器需要先解除 Flash 寄存器读保护（在擦写 Flash 前一定要先执行这个指令，否则擦写 Flash 会失败）

主机格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令反码 | 校验码 | 包尾 |
| 0x7C | 0x02 | 0xFD | 0xFF | 0x7E |

注意：主机命令如果不需要发送数据信息，可以去掉”数据长度”+”数据”。

从机格式：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令码反码 | 执行结果 | 数据 | 数据 | 数据 | 数据 | 校验码 | 包尾 |
| 0x7C | 0x02 | 0xFD | 0x55/0xAA | 0x0 | 0x0 | 0x0 | 0x0 | 0xXX | 0x7E |

1. 打开 Flash 寄存器写保护(指令码 0x03)

命令作用：程序下载完成，打开 Flash 寄存器写保护（程序更新全部完成了，即将跳转到 APP 运行程序前，运行该指令）

主机格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令反码 | 校验码 | 包尾 |
| 0x7C | 0x03 | 0xFC | 0xFF | 0x7E |

注意：主机命令如果不需要发送数据信息，可以去掉”数据长度”+”数据”。 从机格式：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令码反码 | 执行结果 | 数据 | 数据 | 数据 | 数据 | 校验码 | 包尾 |
| 0x7C | 0x03 | 0xFC | 0x55/0xAA | 0x0 | 0x0 | 0x0 | 0x0 | 0xXX | 0x7E |

1. APP 程序区全擦除(指令码 0x04)

命令作用：擦除整个 APP 程序区数据，APP 程序地址范围：0x01001000~0x0x01007FFF。用户可以根据自身需求决定是否执行该命令。

主机格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令反码 | 校验码 | 包尾 |
| 0x7C | 0x04 | 0xFB | 0xFF | 0x7E |

注意：主机命令如果不需要发送数据信息，可以去掉”数据长度”+”数据”。

从机格式：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令码反码 | 执行结果 | 数据 | 数据 | 数据 | 数据 | 校验码 | 包尾 |
| 0x7C | 0x04 | 0xFB | 0x55/0xAA | 0x0 | 0x0 | 0x0 | 0x0 | 0xXX | 0x7E |

1. 烧写 APP 区程序(指令码 0x05)

命令作用：覆盖更新程序数据所对应 Flash 空间，程序有多大，即覆盖多大的Flash。

例如：APP 程序只有 10K，那么只会覆盖 APP 程序区前 10K 的空间，后续 Flash 空间不擦也不写。

主机格式：

注意：主机命令如果不需要发送数据信息，可以去掉”数据长度”+”数据”。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令码反码 | 烧写数据地址高8位 | 烧写数据地址 | 烧写数据  地址 | 烧写数据地址低8位 | 烧写数据地址累加校验码 | Flash烧写  数据长度 | 烧写数据 | … | 烧写数据 | 校验码 | 包尾 |
| 0x7C | 0x05 | 0xFA | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | ... | 0xXX | 0xXX | 0x7E |

注意：1. 烧写数据地址（总共 32 位数据）：从 0 开始，即APP 起始地址（0x01001000）对应 0x0

* 1. 烧写数据地址累加校验码：对地址数据做累加校验
  2. Flash 烧写数据长度：数据填写范围为 0~255，实际长度需自动+1。例如该值为 0xFF，实际数据长度应为 256 个。
  3. 烧写数据：根据 Flash 烧写长度值来定，实际数据个数为 1~256.
  4. 校验码为整个包的数据，去掉包头包尾以及本身 3 个数据，做累加和。

从机格式：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令码反码 | 执行结果 | 数据 | 数据 | 数据 | 数据 | 校验码 | 包尾 |
| 0x7C | 0x05 | 0xFA | 0x55/0xAA | 0x0 | 0x0 | 0x0 | 0x0 | 0xXX | 0x7E |

1. 校验刚才更新完成的 APP 程序区 checksum (指令码 0x06)

命令作用：1.校验整个APP 程序区（要在烧写程序完成后才能执行该命令）

2.升级 APP 程序前，故意将进入 APP 程序区标志位故意改写成 0x1234，确保升级过程中，进入 APP 程序区标志位并非为 0x5555 或者 0xFFFF。

注意：1. Flash 数据 Checksum 校验方法采用CRC16 算法，具体算法在 Bootloader 程序的 flash.c 文件中可以找到（函数 u16 CRC16\_Check(void)）。

2.程序Checksum 计算以扇区为单位，一个扇区是 512 字节。例如，用户的 APP 程序空间占用为

0x1001000~0x1001380，程序 Ckecksum 需要计算的数据地址为 0x1001000~0x10013FF。主机格式：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令反码 | 预留 | 预留 | Checksum 高位 | Checksum 地位 | 校验码 | 包尾 |
| 0X7C | 0x06 | 0xF9 | 0x00 | 0x00 | 0xXX | 0xXX | 0xXX | 0x7E |

注意：主机命令如果不需要发送数据信息，可以去掉”数据长度”+”数据”。

从机格式：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令码反码 | 执行结果 | 数据 | 数据 | Checksum高8位 | Checksum低8位 | 校验码 | 包尾 |
| 0x7C | 0x06 | 0xF9 | 0x55/0xAA | 0x0 | 0x0 | 0xXX | 0xXX | 0xXX | 0x7E |

1. 程序更新完成后，执行APP 区程序 (指令码 0x07)

命令作用：跳转到APP 程序，执行 APP 程序主机格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令反码 | 校验码 | 包尾 |
| 0x7C | 0x07 | 0xF8 | 0xFF | 0x7E |

注意：主机命令如果不需要发送数据信息，可以去掉”数据长度”+”数据”。 从机格式：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令码反码 | 执行结果 | 数据 | 数据 | 数据 | 数据 | 校验码 | 包尾 |
| 0x7C | 0x07 | 0xF8 | 0x55/0xAA | 0x0 | 0x0 | 0x0 | 0x0 | 0xXX | 0x7E |

注意：主机有可能收不到从机发来的命令，因为在从机发送数据前，已经跳转到APP 了。

1. 芯片复位，程序从 APP 进入 Bootloader (指令码 0x08)

命令作用：当程序正常运行在APP 区，需要升级程序时，调用该命令，芯片复位进入Bootloader， 执行升级程序的指令。

注意：该命令需要 APP 程序里软件命令配合，该命令不在 Bootloader 中。，若用户自己的 APP 程序里不写该命令，则上位机发送该命令无法让芯片进入Bootloader。

主机格式：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | 指令码 | 指令反码 | 数据 | 数据 | 数据 | 是否更新程序 | 校验码 | 包尾 |
| 0x7C | 0x08 | 0xF7 | 0xXX | 0xXX | 0xXX | 0x55/0xAA | 0xXX | 0x7E |

注意：是否更新程序（0x55）:需要更新程序，从机需要先往地址 0x1000E00 写入 0xAAAA（代表需要更新程序），然后复位芯片，进入Bootloader。

是否更新程序（0xAA）:不需要更新程序，从机只是复位，进入Bootloader，但不会写需要 更新程序的标志位。

从机格式：无，在 APP 程序中，响应该命令后复位进入Bootloader，无需响应主机。