单区升级设计方案

# 1、设计背景

目前采用A、B区升级方案，浪费了大量的Flash内存空间；由于设备端库升级频率较快，代码体积64K芯片已经无法容纳新库，128K芯片的新库设备代码已经十分接近128Kflash的分区大小，若不更换新的升级方法，必然导致要升级硬件才能使用新库的新功能；考虑到目前老产品的维护问题，采取升级硬件的方法 基本完全放弃了老产品的维护，同时升级硬件也会相应的增加硬件成本。

# 2、设计预期

⑴、此版本的bootloader支持433M的无线升级、USB的本地升级、串口升级等升级方式，网络升级；

⑵、对于433M的升级同样支持批量升级；USB的本地升级、串口升级等本地升级前必须支持一个安全握手校验，防止非法侵入；

⑶、bootloader区采用读保护的方式，防止引导区非法读出数据和修改；

⑷、升级准备阶段采用握手的机制，升级过程中采用“UDP”的方式，升级校验的过程中，主机重发丢失的数据升级包，直到升级校验完成（重发次数有限制）；

⑸、对于64K/128K的老设备，也可通过过渡升级的方式进行更新固件；

# 3、设计方案

## 3-1、总的FLASH分区图和程序流程图



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Flash 大小** | **Bootloader\_0大小** | **Bootloader\_0\_flag**  **分区大小（保留）** | **Bootloader\_1大小** | **Bootloader\_1\_flag**  **分区大小（保留）** | **用户区大小** | **用户数据区大小** |
| 64K Flash | 1K | 1K | 18K | 1K | 43K | 保留① |
| 128K Flash | 1K | 1K | 22K | 1K | 103K | 保留① |
| 256K Flash | 2K | 2K | 22K | 2K | 228K | 保留① |
| 384K Flash | 2K | 2K | 22K | 2K | 356K | 保留① |
| 512K Flash | 2K | 2K | 22K | 2K | 484K | 保留① |
| 1M Flash | 16K | 16K | 16K+64K | 16K | 896K | 保留① |

说明：①、在bootloader\_1缺失的情况下，不允许升级APP程序；

②、用户数据区的大小是不同的设备产品而定义，不在做统一规定，在制作烧录包时，需要设计者指定最小的用户区保留大小；

③、bootloader\_0区域存储的特殊数据：

typedef struct //设备硬件信息< 0x08000300 >

{

u32 hardVer;

u32 Reserve1;

u32 chip\_type;

u32 data;

u32 time;

u8 str[100];

}Hard\_t;

typedef struct //设备分区表< 0x08000380 >

{

u32 boot0\_start\_addr; //bootloader0起始地址

u32 boot0\_flag\_start\_addr; //bootloader0\_flag起始地址

u32 boot1\_start\_addr; //bootloader1起始地址

u32 boot1\_flag\_start\_addr; //bootloader1\_flag起始地址

u32 user\_app\_start\_addr; //用户App的起始地址

u32 user\_data\_start\_addr; //用户数据区的起始地址

u32 flash\_end\_addr; //Flash芯片的结束地址

}DevPartionTable\_t;

④、bootloader\_0\_flag区域存储的特殊数据：

typedef struct

{

u32 jumpflag; //Bootloader\_0程序跳转标记

u32 SoftVer; //bootloader1软甲版本

u32 appSize; //bootloader1软件大小

u32 data; //bootloader1编译日期

u32 time; //bootloader1编译时间

}Boot0Info\_t;

typedef struct{

u32 SetAddrFlag; //重设有效地址标记

u32 DeviceID; //重设的设备地址

u32 DeviceIDInver; //设备地址的反码

}DeviceIDSet\_t;

typedef struct{

u32 EncryptFlashFlag; //FLASH加密标记

u32 EncryptFlashData; //Flash加密密钥

}FlashEncrypt\_t;

⑤、bootloader\_1\_flag区域存储的特殊数据：

typedef struct //设备软件信息

{

u32 jumpflag; //Bootloader\_1程序跳转标记

u32 deviceType;//设备类型

u32 SoftVer;//软件版本号

u32 appSize;//：APP程序的大小，以字节为单位

}Soft\_t;

typedef struct

{

u16 channel; //信道

u8 encrypt; //通信加密

u8 pace; //信道间隔

u8 Reserve[12];//保留空间

}RadioPara\_t;

⑥、Flash分区示意图

64K FLASH分区



128K FLASH分区



256K/384K/512K/768K/1024K FLASH分区



## 3-2、正在运行的设备接收到升级命令后的运行流程图



## 3-3、bootloader\_1升级APP程序流程

3-4、升级说明

①、当主机接收到升级命令后，清除升级标示，运行bootloader程序等待升级进行，为了防止多源同时进行的升级的问题，当设备选定一个升级源后，锁定该源进行升级，除非升级失败，否则该源不能随便被替换；

②、进入升级模式后，设备升级丢包重发的次数由主机定义，超过重发次数，设备等待超时，升级失败；

③、为了防止某些应用设备在升级失败变砖的情况，同时该设备无法可恢复的被拆卸（如魔方、面板等），在设备硬件电路设计的时候尽可能的留出一个重置管脚，用了设备bootloader引导的恢复；

④、对于老设备（没有采用单区升级的设备）的过渡升级问题，采用以下流程：



⑤、对于有硬件看门狗的设备，运行bootloader的程序必须小心喂狗；

⑥、程序升级过程均采用断点续传功能，EEPROM的0x2D00~0x2E00作为续传的缓冲区，F0阶段在设备端在擦除FLASH之前首先检查有有效的断点文件，若有，则需要通知升级主机；

# 4、升级属性

## 4-1、升级属性 [0xF4] [RW]

概述：设备进入升级。

对象：单节点操作

代码：0xF4

权限：写

### 4-1-1、写入（Write）

**描述：**设备进入升级状态

**发送(Request):**

|  |  |  |
| --- | --- | --- |
| **参数块** | **参数1** | **参数2…** |
| **内容** | 操作类型 | 实际参数区数据 |
| **字节数** | 1byte | n bytes |
| **说明** | 0xF1:升级准备属性 0xF3:升级数据校验属性 | 具体命令见下表（1）所示 |

**0xF1:升级准备：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数块** | **参数2** | **参数3** | **参数4** | **参数5** | **参数6** |
| 内 容 | 升级对象 | 设备类型 | 升级文件的总大小 | 每个升级包大小 | 升级软件版本号 |
| 字节数 | 1 | 2 | 4 | 1 | 4 |
| 说 明 | 0:升级APP区 1:升级bootloader1 |  |  | 单位（byte） |  |

**0xF3:升级数据校验属性：**

|  |  |  |
| --- | --- | --- |
| **参数块** | **参数2** | **参数3** |
| 内 容 | 4 | 2 |
| 字节数 | 升级文件大小 | 数据包CRC值 |
| 说 明 |  | 无 |

**应答(Response):**

**0xF1应答：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数块** | **参数1** | **参数2** | **参数3** | **参数4** | **参数6** |
| **内容** | 执行结果 | 当前设备类型 | 当前硬件版本 | 当前软件版本 | Flash型号 |
| **字节数** | 1 | 2 | 4 | 4 | 1 |
| **说明** | 见操作结果表 |  |  |  | 见下（1） |
|

（1）、0（保留）、1（STM32F103C8<SRAM:20KB;FLASH:64KB>）、2（STM32F103CB<SRAM:20KB;FLASH:128KB>）、3（STM32F103RC<SRAM:64KB;FLASH:256KB>）、4（STM32F103RD<SRAM:64KB;FLASH:384KB>）、5（STM32F103RE<SRAM:64KB;FLASH:512KB>）、6（STM32F103RF<SRAM:96KB;FLASH:768KB>）、7（STM32F103RG<SRAM:96KB;FLASH:1MKB>）、8（STM32L151C8/C8-A<SRAM:10,16KB;FLASH:64KB>）、9（STM32L151CB/CB-A<SRAM:16,32KB;FLASH:128KB>）、10（STM32L151CC<SRAM:32KB;FLASH:256KB>）、11（STM32L151RB<SRAM:32KB;FLASH:128KB>）、12（STM32L151RC<SRAM:32KB;FLASH:256KB>）、13（STM32L151RD<SRAM:48KB;FLASH:384KB>）、14（STM32L151RE<SRAM:80KB;FLASH:512KB>）、15（STM32L152C8/C8-A<SRAM:10,16KB;FLASH:64KB>）、16（STM32L152CB/CB-A<SRAM:16,32KB;FLASH:128KB>）、17（STM32L152CC<SRAM:32KB;FLASH:256KB>）、

18（STM32L152RB<SRAM:32KB;FLASH:128KB>）、19（STM32L152RC<SRAM:32KB;FLASH:256KB>）、20（STM32L152RD<SRAM:48KB;FLASH:384KB>）、21（STM32L152RE<SRAM:80KB;FLASH:512KB>

**0xF3应答：**

|  |  |
| --- | --- |
| **参数块** | **参数1** |
| **内容** | 执行结果 |
| **字节数** | 1 |
| **说明** | 见操作结果表 |
| **备注** |  |

### 4-1-2、获取（Read）

**描述：获取设备升级准备就绪状态**

**发送(Request):**

|  |  |  |
| --- | --- | --- |
| **参数块** | **参数1** | **参数2…** |
| **内容** | 操作类型 | 实际参数区数据 |
| **字节数** | 1byte | n bytes |
| **说明** | 0xF0: 获取设备升级准备就绪状态 0xF2: 升级数据包校验属性 | 具体命令见下表（1）所示 |

**发送(Request): 0xF0获取：**

|  |  |
| --- | --- |
| **参数块** | **参数2** |
| 内 容 | 升级对象 |
| 字节数 | 1 |
| 说 明 | 1:升级APP区 2:升级bootloader1 |

**描述：获取设备缺包ID**

**发送(Request): 0xF2获取：**

|  |  |
| --- | --- |
| **参数块** | **参数2** |
| 内 容 | 无 |
| 字节数 | 无 |
| 说 明 | 无 |

**应答(Response): 0xF0应答：**

|  |  |
| --- | --- |
| **参数块** | **参数1** |
| **内容** | 执行结果 |
| **字节数** | 1 |
| **说明** | 1标识符合升级条件,其他标识不符合 |
| **备注** |  |

**应答(Response): 0xF2应答：**

|  |  |  |
| --- | --- | --- |
| **参数块** | **参数1** | **参数2** |
| **内容** | 执行结果 | 缺包ID |
| **字节数** | 1 | 2\*N |
| **说明** | 见操作结果表 | 各对应ID |
| **备注** |  |  |

## 4-2、升级数据发送属性 [0xF5] [W]

概述：升级文件数据包发送属性

对象：广播操作

代码：0xF5

权限：写

### 4-2-1、数据包发送属性（Write）

**发送(Request):**

|  |  |  |
| --- | --- | --- |
| **参数块** | **参数1** | **参数2** |
| 内 容 | 包ID | 数据包内容 |
| 字节数 | 2 | x |
| 说 明 |  | 由进入升级状态时上位机传参确定 |

# 5、设计缺陷

⑴、在从老设备升级至新的代码分区的过程中，流程过于复杂，设备有完全挂掉的可能，必须要重新烧录才能解决，对于某些不可拆卸的设备来说是致命的；

⑵、bootloader失效会导致整个设备变砖，必须烧录固件才能使用，防止误操作造成设备瘫痪；