南京凌鸥创芯电子有限公司

LKS-Bootloader 使用说明

版本	修改时间	修改人	修改说明
1.1	2021-10-22	Wucf	文档整理
1.2	2021-11-30	Wucf	增加说明，检查 Linker 页勾选”Use Memory Layout from Target Dialog”
1.3	2021-12-14	Wucf	调整文件报文最大 200 字节，提高传输效率。 增加说明，起始地址应为 512 的倍数。
1.5	2022-11-25	Wucf	增加 CAN Bootloader 说明

目录

1. 概述

本文档主要介绍 lks08x 系列芯片 bootloader 功能的使用。需要用到的工程或工具包括：

- Bootloader 程序
- 用户应用 App 程序
- 上位机工具

2. Bootloader 程序

LKS 芯片上电后，Bootloader 程序率先启动，在 50ms 内如果收到串口发来的 IAP 请求，则执行升级过程。否则 50ms 后直接运行用户 App。

1. 设置跳转地址和 flash 编程地址；

   在 Bootloader 程序中，跳转地址和 flash 编程地址是相同的宏定义，保证该地址大于 booloader 程序 bin 文件大小。

   #define APP_ADDR 0XA00

2. 串口通信；
3. 接收上位机数据；
4. 执行跳转指令；

   跳转指令：

```
__disable_irq();                                //禁止所有中断
jump2app=(iapfun)*(UINT32 *)(APP_ADDR+4); //取中断复位向量函数
__set_MSP(*(volatile u32 *)APP_ADDR);       //设置堆栈
jump2app();                                     //执行复位向量函数
```

注意跳转指令禁止在中断中调用，如果在 bootloader 中断中调用跳转，该中断就不会返回，堆栈就不会有出栈操作，则中断会一直处于挂起中断，跳转后的 App 程序，对优先级小的中断不会响应。

3. App 程序

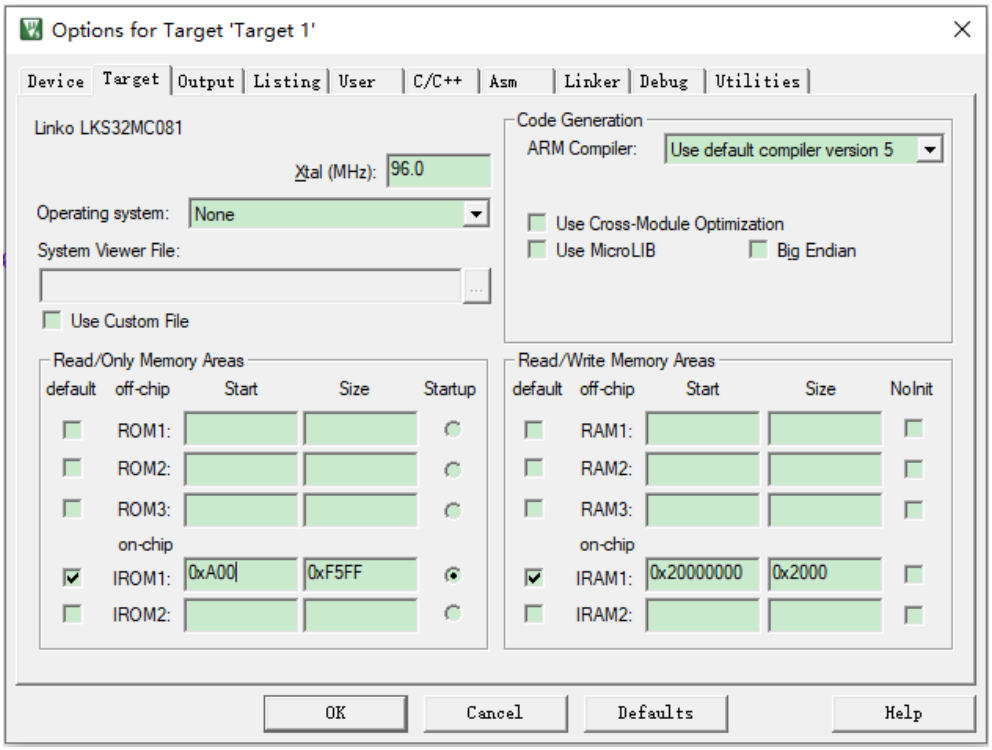
在普通程序的基础上，程序代码做地址偏移和重映射中断向量表入口地址。

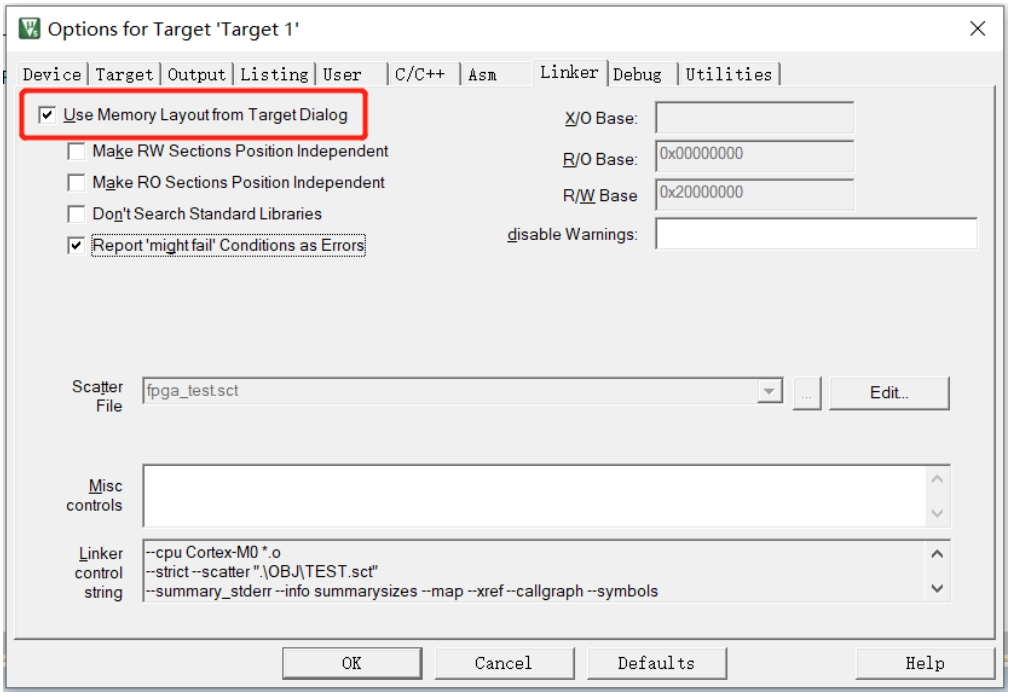
通过 keil 编译偏移后的代码，偏移 0xA00 地址，在重映射中断向量表入口地址前，必须关闭所有中断。

3.1.Keil 设置

点击 Keil 中的魔法棒”Options for Target”。

- Target 页：设置 IROM1 的起始地址为 0xA00（可以为其他值，必须与 bootloader 程序中跳转的地址一致。请选择扇区大小 512 的整数倍作为起始地址，否则会造成 Flash 空间浪费。如果 App 与 Bootloader 使用了同一个扇区，为 App 擦除 Flash 时甚至会造成 Bootloadre 程序进入 HardFault）。
- Linker 页：确保” Memory Layout from Target Dialog”选项已勾选，否则前一步设置无效。





3.2. 中断向量表重映射

重映射中断向量表入口地址为 0xA00（可以为其他值，必须与 bootloader 程序中跳转的地址一致）。

```
int main(void)
{
    __disable_irq();

    Clock_Init();
    delay_init(96);

    REG32(0xE000ED08) = 0x00000A00;//重映射中断向量入口地址

    delay_ms(1);
```

4. 支持的连接类型

截止 v1.5 版本，已添加的连接类型有 UART 和 CAN。

4.1. UART 连接配置

UART 连接下，默认支持 9600 波特率。

下位机 APP 程序起始地址 0x0A00，即 Flash 中 2.5k 的位置，请根据文档第三节内容进行修改。

4.2. CAN 连接配置

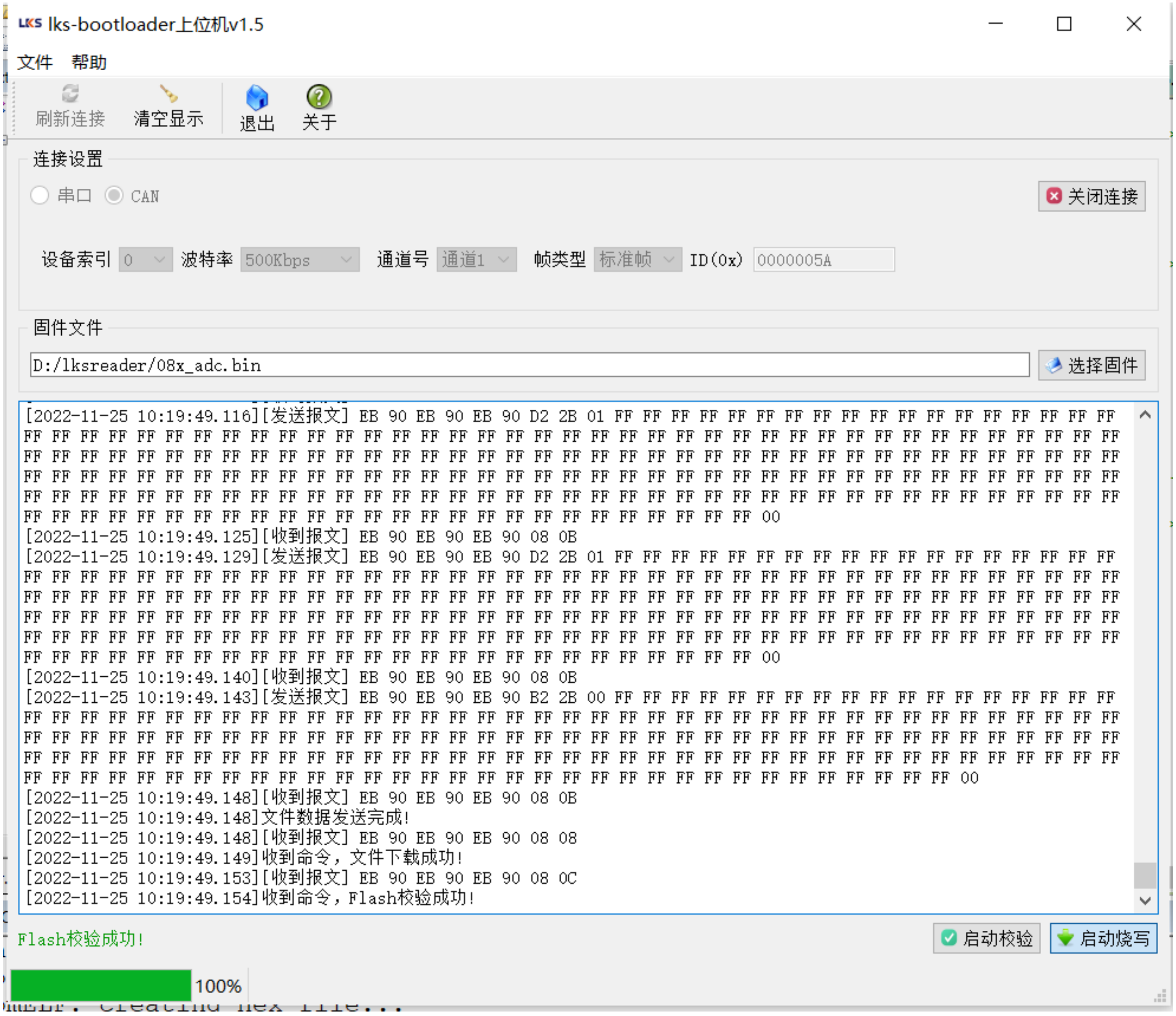
CAN 连接下，默认支持 500Kbps 波特率，接收标准帧+扩展帧，过滤 ID=0x0000005A，可根据需要修改配置。

由于 CAN 的 Bootloader 对应 hex 文件稍大一些，下位机 APP 程序起始地址 0x0C00，即 Flash 中 3k 的位置，请根据文档第三节内容进行修改。

candata.h

```
1 #ifndef CANDATA_H_
2 #define CANDATA_H_
3
4 #include "lks32mc08x.h"
5
6 #define APP_ADDR        0xC00           // APP程序起始地址
7 #define CAN_ID          0x5A            // CAN过滤的ID
8
```

Bootloader_CAN 工程的编译优化等级设置为 Levle1，以减小 hex 文件的体积。



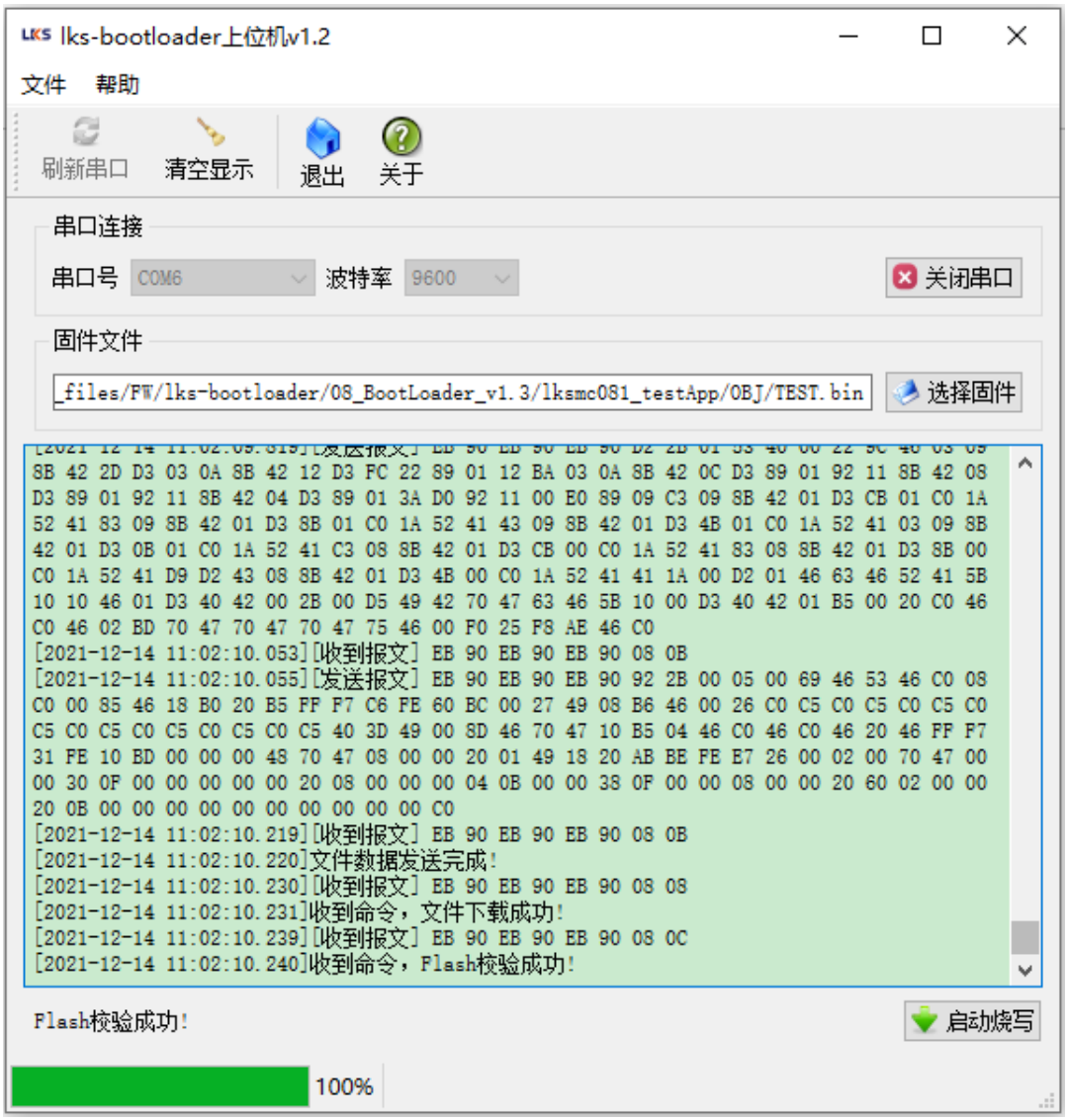
5. 上位机程序

选择 App 生成的 bin 文件，通过串口将 bin 下传到 Bootloader。上位机工具运行在 Windows 系统下，需要 USB 转串口的工具与目标板连接。

上位机启动烧写后，每 30ms 左右发送一次下载请求，直到得到确认为止。下位机 Bootloader 等待时间为 50ms，理论上能稳定命中。

上位机操作步骤如下：

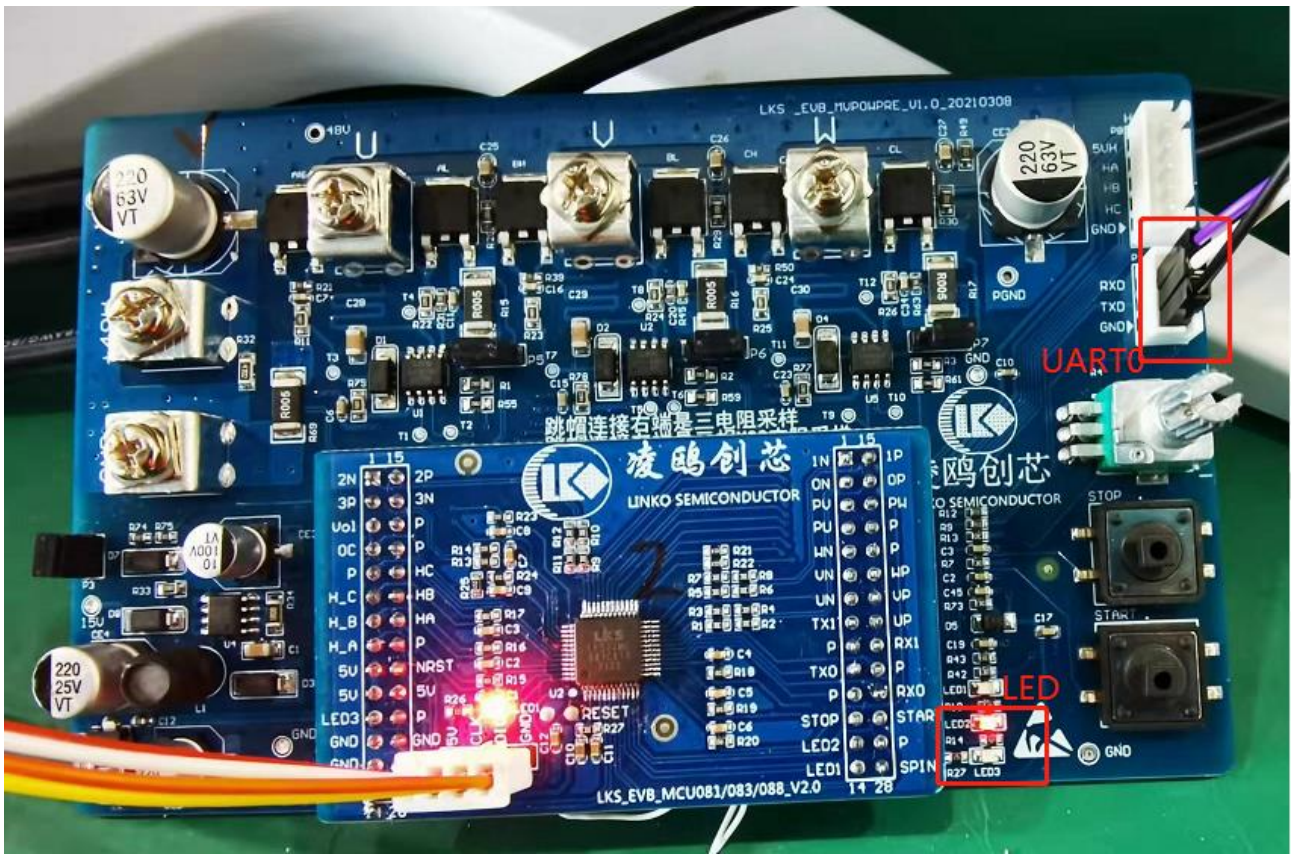
1. 打开上位机；
2. 连接上位机与下位机之间的串口线；
3. 配置波特率与 bootloader 串口波特率匹配；
4. 打开串口；
5. 选择文件；
6. 启动烧写；
7. 下位机上电，等待程序下载和跳转。



6. 实验现象

提供的 Bootloader 和 testApp 例程可以直接在 lks08 的开发板上执行。开发板的 UART0 用于连接上位机串口。

按照文档第 4 节提供的步骤执行完毕后，开发板的 LED 以 200ms 的频率持续闪烁。



7. 协议说明

采用 UART 通信，波特率 9600，停止位 1 位，无奇偶校验位。对于多字节数据，传输时低字节在前，与 M0 使用的字节序一致。

7.1. 下行命令，下发请求

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	0CH
请求命令	06H-烧写请求/05H-校验请求

请求下发文件的总长度，4 字节	fileLength_B0
	fileLength_B1
	fileLength_B2
	fileLength_B4
请求下发文件的校验码，4 字节	fileCrc_B0
	fileCrc_B1
	fileCrc_B2
	fileCrc_B4

上位机启动下载后每 30ms 发送一次请求帧，收到请求确认后停止发送。文件校验码采用四字节表示，供下位机进行 Flash 校验时使用。四字节 CRC 的计算方式如下，计算正确的前提是使用正确的程序文件，长度是 4 的倍数，否则应考虑使用 0xFF 补充字节数。

```
u32 WordXorCrc(u8 *content, u32 len)
{
    u32 crc = 0;
    for (u32 i = 0; i < len; i+=4)
    {
        u32 data = *(u32*)content[i];
        crc ^= data;
    }

    return crc;
}
```

7.2.上行命令，请求确认

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH

	90H
报文长度	08H
请求命令	07H

确认上位机的请求，此时下位机开始根据文件总长度擦除需要的 Flash 扇区。

7.3.上行命令，Flash 擦除完成

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	08H
请求命令	17H

Flash 擦除完成后，上位机可以开始下发文件数据。

7.4.下行命令，下发文件数据

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	N+10
文件内容命令	2BH
是否有后续帧	01/00
N 字节文件内容，N≤200	

文件内容的 CRC 校验码，1 字节	

报文中文件数据的校验码采用单字节表示，计算方法如下。

```
u8 WordXorCrc(u8 *pData, u32 len)
{
    u8 crc = 0;
    for (u32 i = 0; i < len; i++)
    {
        crc ^= pData[i];
    }

    return crc;
}
```

7.5.上行命令，收到文件数据确认

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	08H
请求命令	0BH

文件数据采用一发送+一确认的形式循环进行，通信双方保持同一节奏。

7.6.上行命令，文件接收完成

报文头，6 字节	EBH
	90H
	EBH

	90H
	EBH
	90H
报文长度	08H
请求命令	08H

下位机确认文件接收完成，之后自动执行 Flash 数据校验。

7.7.上行命令，数据校验成功

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	08H
请求命令	0CH

数据校验成功表示整个流程已成功完成。下位机经过短暂的延时后，跳转执行用户程序。

7.8.上行命令，数据校验失败

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	08H
请求命令	0DH

数据校验失败表示数据虽然下发完成，但写入的 Flash 数据不完整，需要重新执行。下位机经过短暂的延时后软复位重新执行 Bootloader，上位机也重新开始周期下发请求命令，整个流程自动重新开始。

7.9.上行错误帧，校验码错误

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	08H
请求命令	09H

如果下位机收到文件数据校验码错误，则汇报该错误帧。上位机可以选择重发当前帧或重新开始流程。

7.10. 上行错误帧，文件长度错误

报文头，6 字节	EBH
	90H
	EBH
	90H
	EBH
	90H
报文长度	08H
请求命令	0AH

如果下位机收到一帧中的文件数据不是 4 的倍数，则无法按 WORD 写入 Flash，汇报该错误。上位机应检查自身的数据下发逻辑，保证每一帧中的文件数据长度为 4 的倍数。

7.11. 程序升级流程

7.12. 程序校验流程

7.13. 报文示例

以下展示上位机侧一个完整流程的报文内容。

[发送报文] EB 90 EB 90 EB 90 10 06 38 05 00 00 0C 8B 22 38

[发送报文] EB 90 EB 90 EB 90 10 06 38 05 00 00 0C 8B 22 38

[发送报文] EB 90 EB 90 EB 90 10 06 38 05 00 00 0C 8B 22 38

[发送报文] EB 90 EB 90 EB 90 10 06 38 05 00 00 0C 8B 22 38

[收到报文] EB 90 EB 90 EB 90 08 07

收到命令确认，等待擦除 flash!

[收到报文] EB 90 EB 90 EB 90 08 17

flash 擦除完毕!

准备文件数据!

开始发送文件数据!

[发送报文] EB 90 EB 90 EB 90 D2 2B 01 00 07 00 20 49 01 00 00 57 01 00 00 59 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 61

[收到报文] EB 90 EB 90 EB 90 08 0B

[发送报文] EB 90 EB 90 EB 90 D2 2B 01 5B 01 00 00 00 00 00 00 00 00 00 00 5D 01 00 00 91 09 00 00 65 0B 00 00 67 0B 00 00 69 0B 00 00 6B 0B 00 00 CD 06 00 00 .. .. 9C

[收到报文] EB 90 EB 90 EB 90 08 0B

.....循环发送文件数据.....

.....接收数据确认.....

[发送报文] EB 90 EB 90 EB 90 2A 2B 00 F5 E7 00 00 00 20 01 40 FF 1F 00 00 FF FF 2F 00 FC 0B 00 00 00 00 00 20 00 07 00 00 04 01 00 00 69

[收到报文] EB 90 EB 90 EB 90 08 0B

文件数据发送完成!

[收到报文] EB 90 EB 90 EB 90 08 08

收到命令，文件下载成功!

[收到报文] EB 90 EB 90 EB 90 08 0C

收到命令，Flash 校验成功!