版本	V1.0
日期	20230814

Bootloader(UART)方案

适用APT32F103X系列

Revision History

版本	日期	描述	作者
V1.0	2023-8-14	新建使用说明	YYM/WNN
V1.1	2023-8-31	添加串口工具打开说明	LHY

1. 概述

本文主要描述上位机通过UART通信口对从机板进行代码升级的过程。

如图Figure 1所示，从机板的内存空间分为两部分，低地址空间预先装载bootloader工程，用以配合上位机进行升级，从上位机接收到的app程序代码移至高地址空间。

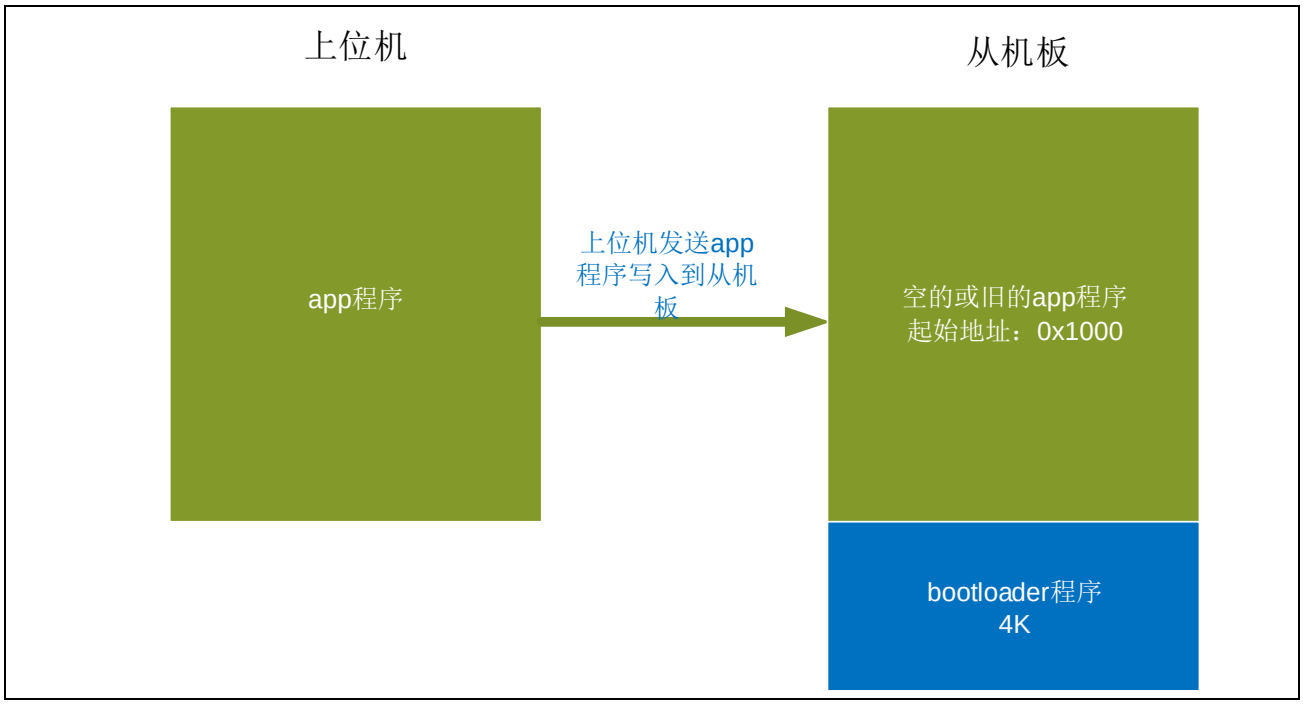

Figure 1主从板APP程序升级图

NOTE：在本demo的实现中

1） bootloader在csi驱动版本上进行开发的，详见3.1章节,代码空间为4K，代码空间起始地址为0x0000。

2）app为应用工程，详见3.3章节，代码起始地址为0x1000。

2. 应用

2.1 准备阶段

1） 编译好 bootloader 后，将工程下载到从机 MCU 中

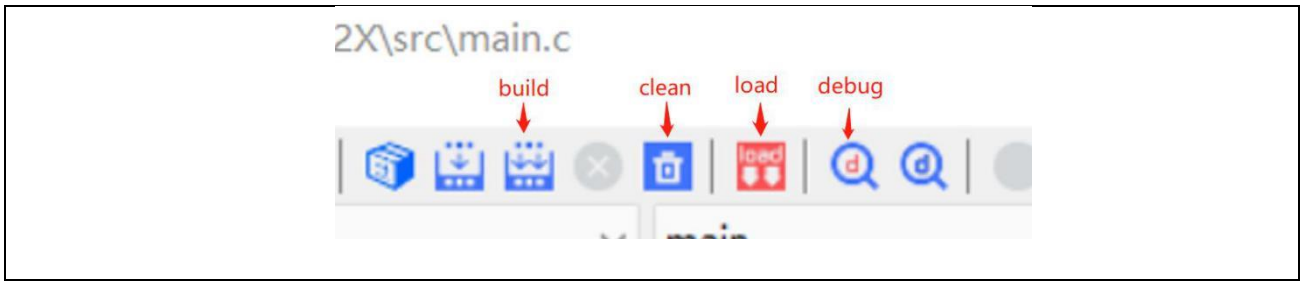

Figure 2 CDK按钮操作图

重新编译demo中的bootloader工程，先点clean 再build 。

点击load，将demo中bootloader的工程，使用仿真器烧录到从机mcu中。

2.2 升级阶段

2.2.1 串口连接

电脑端分别连接升级串口和打印串口，串口需要接地。

连接如图Figure 3所示：

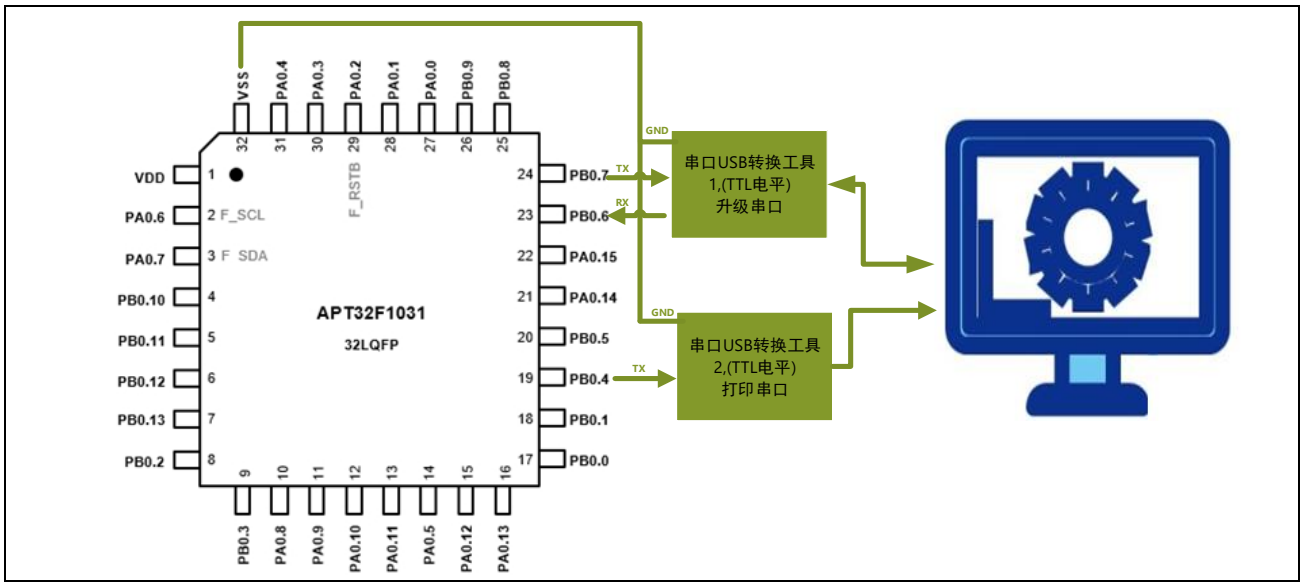

Figure 3 串口接线图

2.2.2 串口工具配置

打印串口，配置如下图所示，同时打开串口

Figure 4 打印串口配置

升级串口，配置如下图所示，同时打开串口

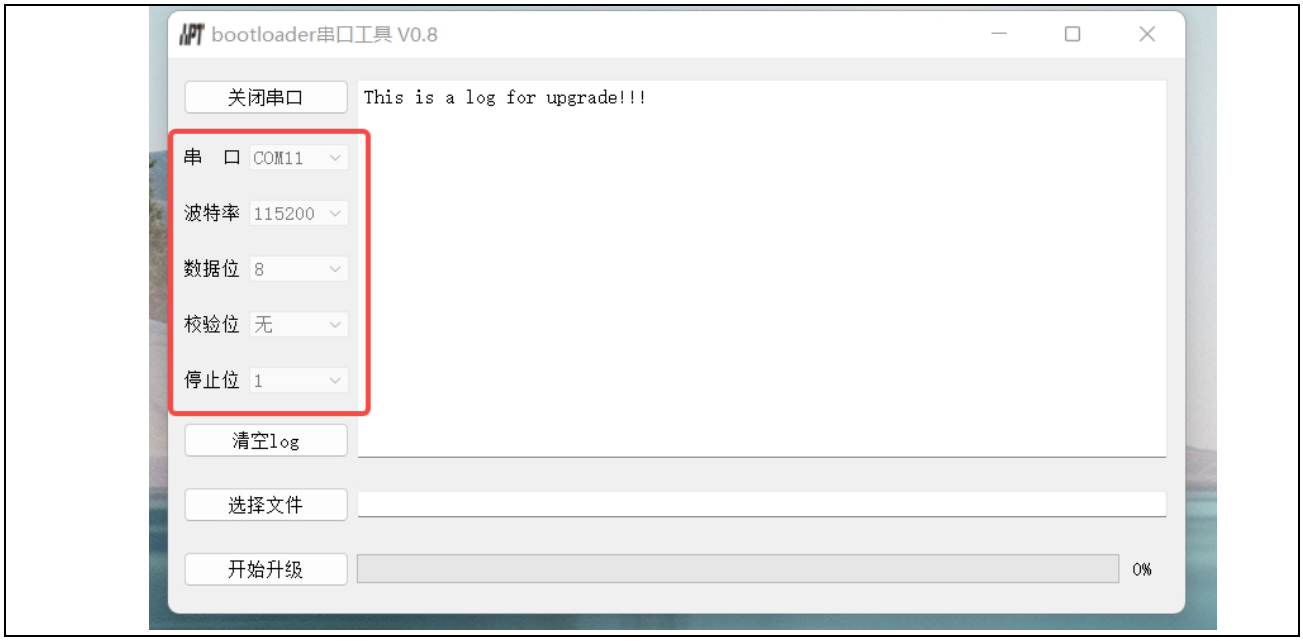

Figure 5 升级串口配置

可通过CDK->Tools->APT_BootloaderUpgradeTool，快捷打开串口升级工具

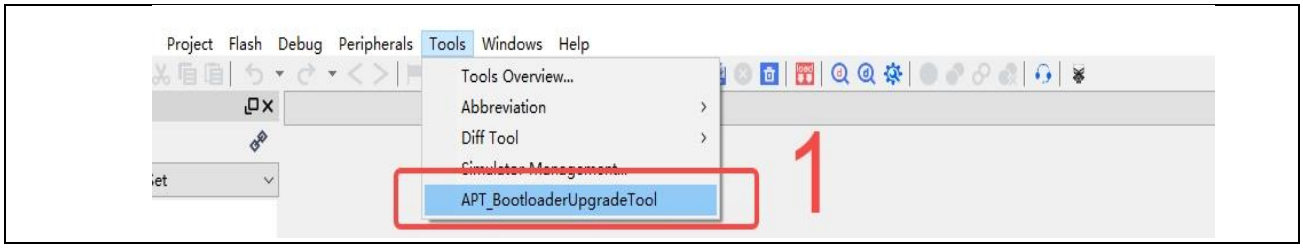

Figure 6 CDK->Tools

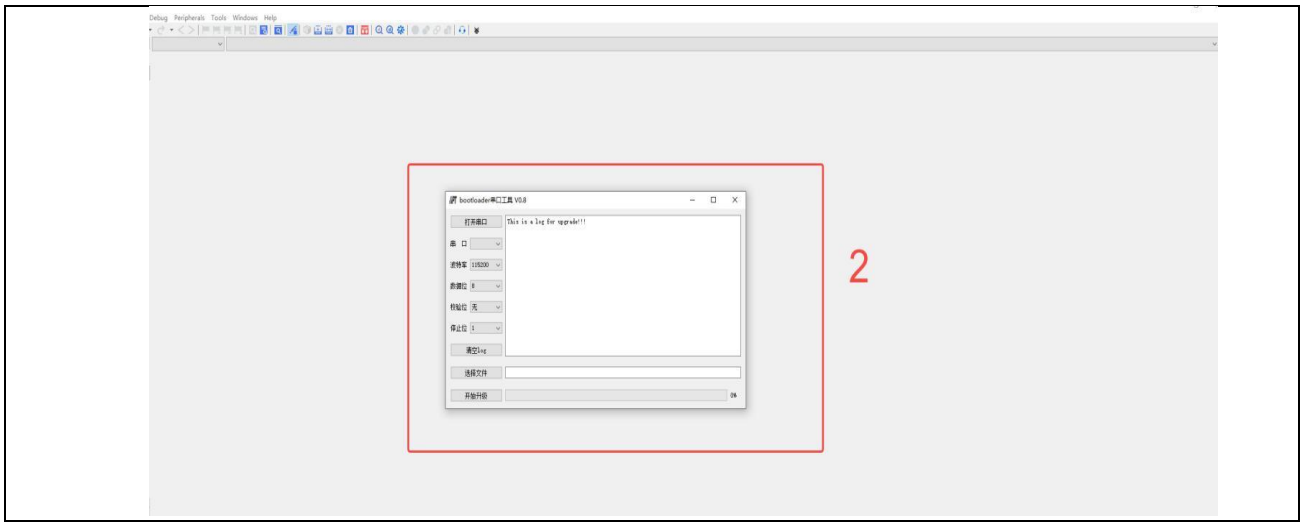

Figure 7 打开串口升级工具

2.2.3 升级操作过程

1)、app工具选择相应的升级文件，点击“开始升级”

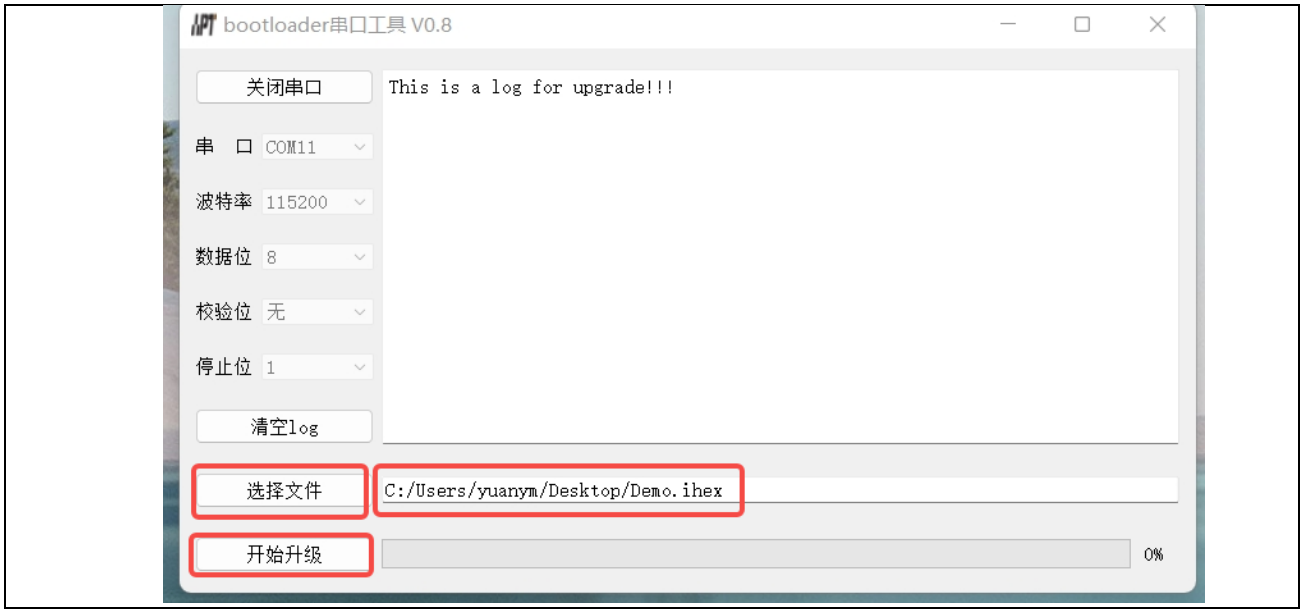

Figure 8 升级文件选择

2)、如果提示以下信息，需要对MCU重新上电

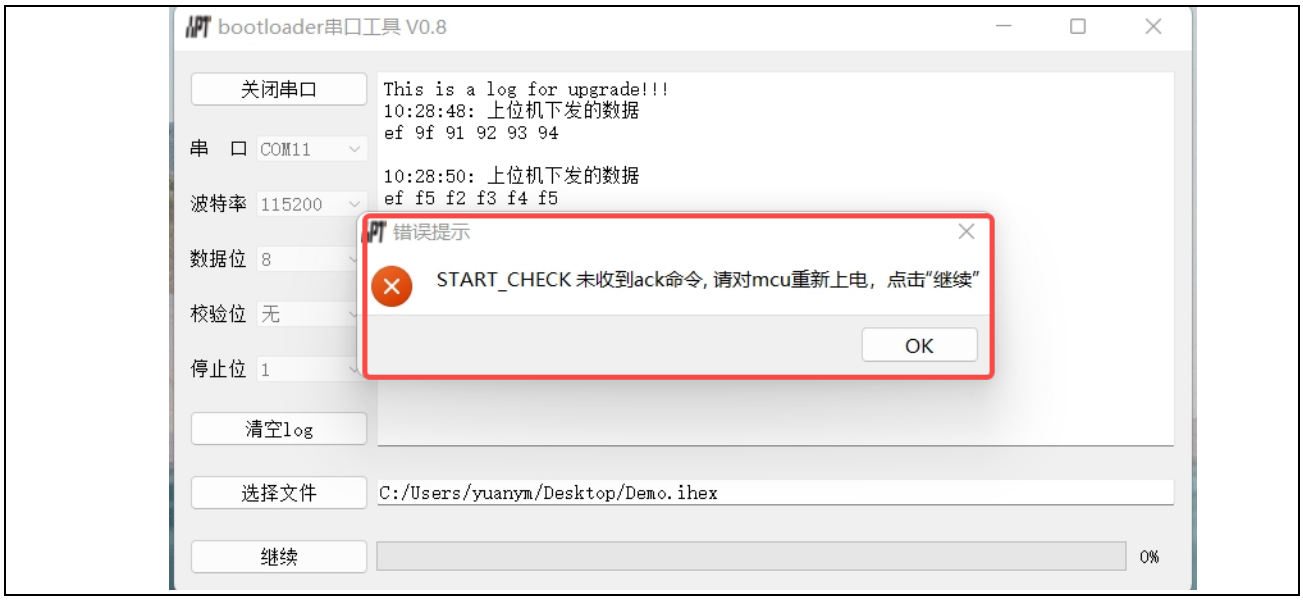

Figure 9 升级错误信息提示

3)、重新上电后，点击“继续” 按钮，即可进行重新升级

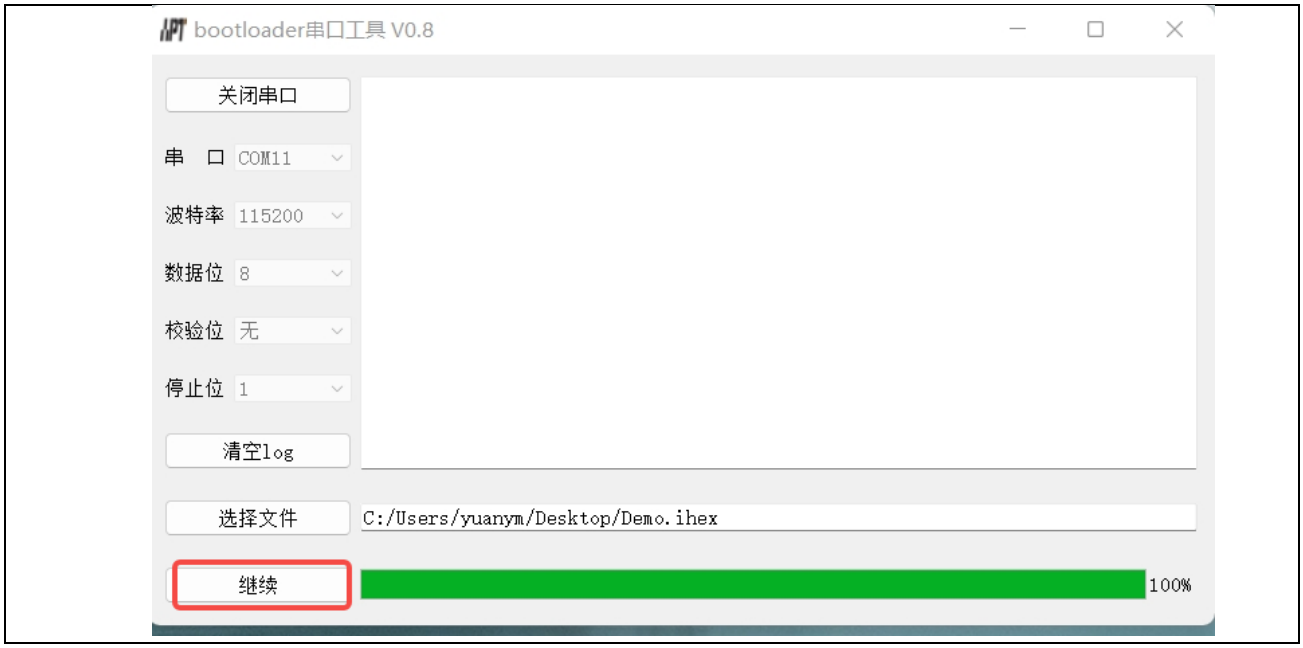

Figure 10 重新升级

4)、升级完成后，上位机工具会提示“升级成功”信息

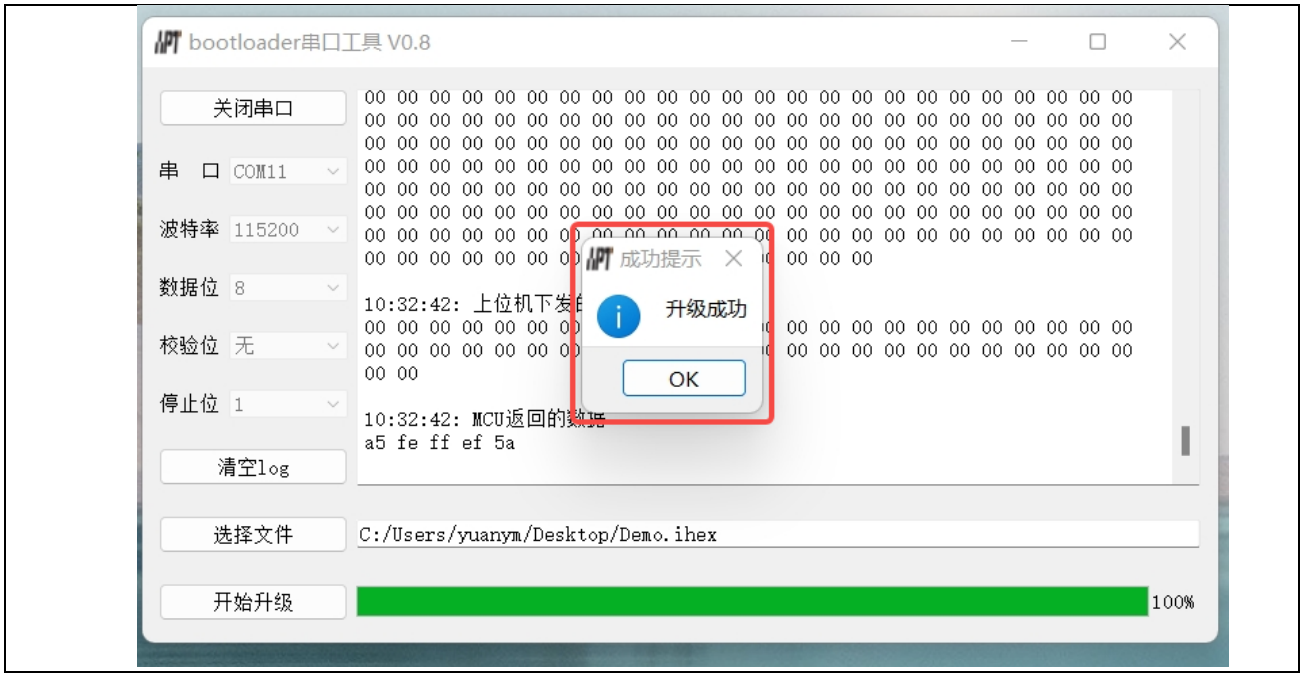

Figure 11 升级成功提示

5)、升级成功后，会跳转到APP程序中运行，同时可在打印串口看到以下信息

Figure 62 串口打印信息

3. 实现原理

3.1 bootloader实现原理

整个 bootloader的程序流程如图Figure 11所示：

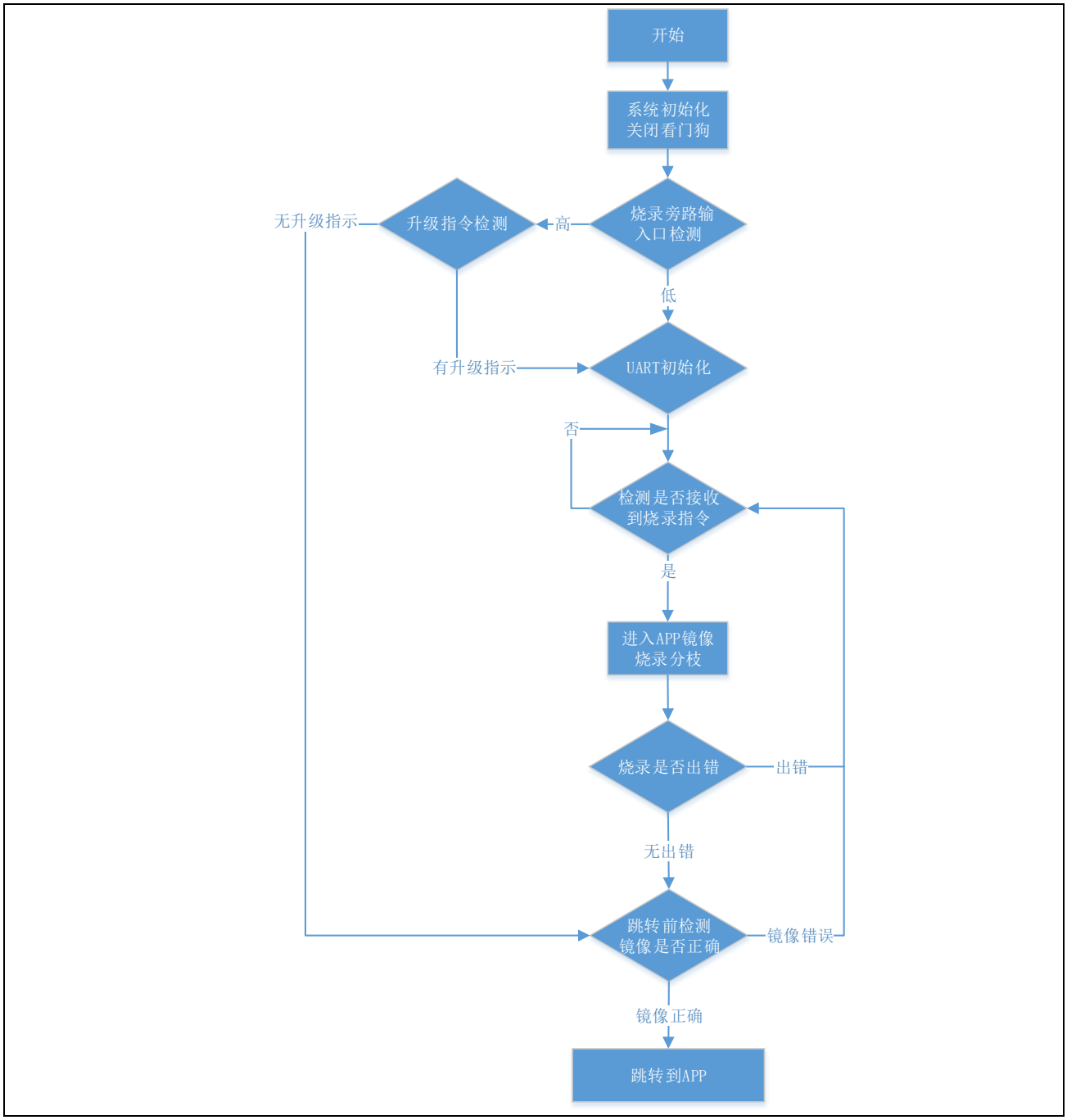

Figure 73 bootloader实现流程图

NOTE：在本demo实现中，

1) 烧录旁路输入口为PB06

2) 升级指示检测为：SYSCON->UREG0寄存器是否为0x01

3.2 app实现原理

bootlooder升级，可以通过以下两种方式实现：

1）、上位机连接后，将check pin管脚置低，然后从机上电，此时从机检测到烧录旁路口为低，会进入bootloader模式，这种升级模式对app应用程序无要求。

2）、通过app跳转的模式，在确保从机加载了正确的app程序，从机可以先上电，运行至app应用程序中。此时可以通过串口指令，使程序跳转回bootloader模式下。这个过程对app应用程序有一定的要求，需要占用一个串口且具备识别升级指令的功能。详见Figure 12 app实现流程图。

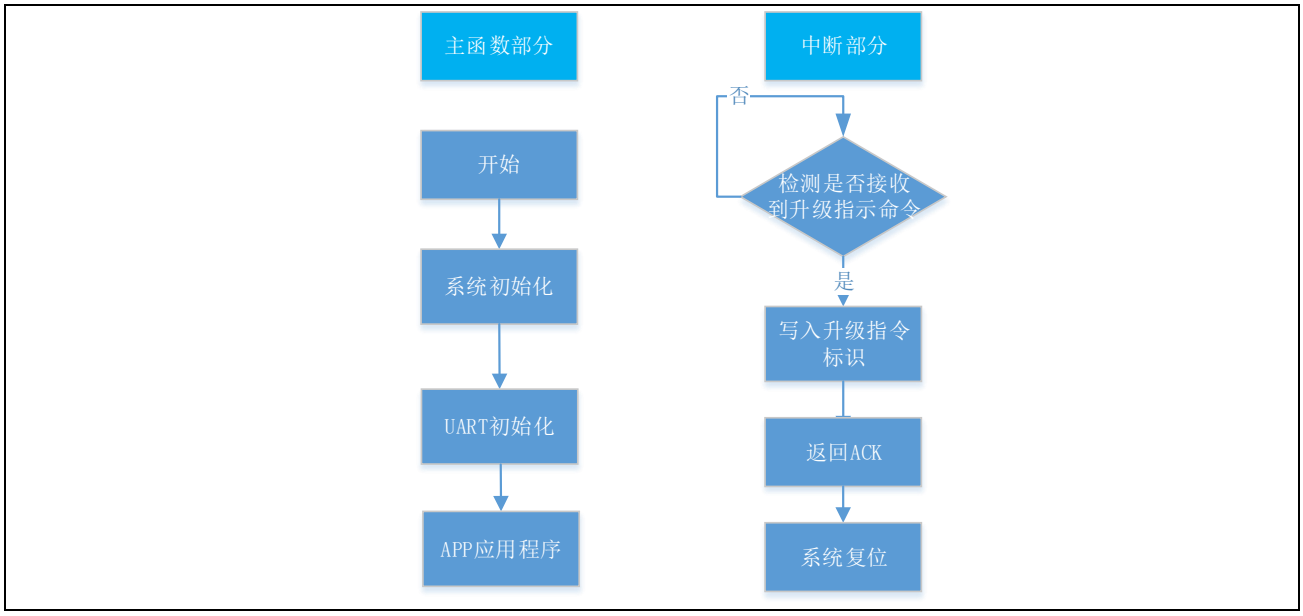

Figure 84 app实现流程图

NOTE：本demo实现中

1） 升级指示命令为：0xEF,0x9F,0x91,0x92,0x93,0x94

2） 升级指示标识为：SYSCON->UREG0 寄存器写入 0x01

3.3 对bootloader区的保护

当bootloader的代码空间小于4K时，可以选择IFC_4K选项使能硬件保护锁。此时固件更新时，启动代码可以擦除所有4K以外的PROM区域，然后将新的固件烧写进去，同时启动代码本身不会被擦除，保持不变。设置如图Figure 9：

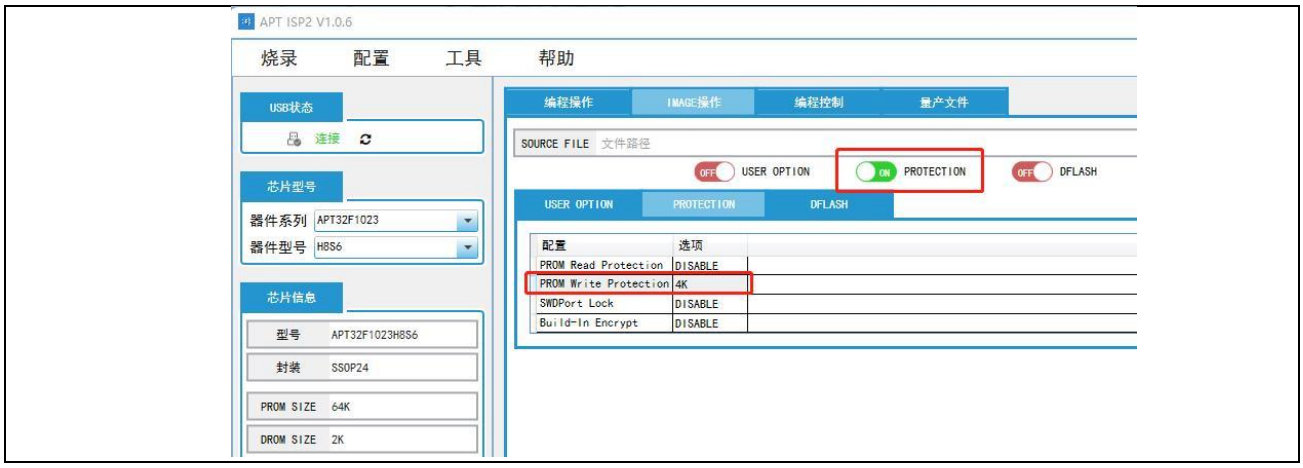

Figure 95 bootloader区保护配置图

3.4 通信协议介绍

例如：start addr协议

1）、上位机发送为 0xEF,0x10,0x00,0x00,0x10,0x00

其中：0xEF为CMD HEAD

0x10为CMD ID

0x00,0x00,0x10,0x00为升级程序写入的起始地址0x1000,，高位在前。

2）、从机回复：0xA5,0x00,0x00,0x10,0x00

其中：0xA5为ACK HEAD

0x00,0x00,0x10,0x00为升级程序写入的起始地址0x1000,，高位在前。

Table 1 主从机协议

协议名称	长度	命令数据(上位机发送)	ACK(从机回复)
start check	6	0xEF,0xF5,X1,X2,X3,X4	0XA5, X1,X2,X3,X4
jump boot	6	0xEF,0x9F,X1,X2,X3,X4	0XA5, X1,X2,X3,X4

start addr	6	0xEF,0x10,X1,X2,X3,X4	0XA5, X1,X2,X3,X4
end addr	6	0xEF,0x20,X1,X2,X3,X4	0XA5, X1,X2,X3,X4
jump addr	6	0xEF,0x30,X1,X2,X3,X4	0XA5, X1,X2,X3,X4
check num	6	0xEF,0x40,X1,X2,X3,X4	0XA5, X1,X2,X3,X4
prog start	6	0xEF,0x80,X1,X2,X3,X4	0XA5, X1,X2,X3,X4

4. 常见问题

4.1 APP_START_ADDR 以及 APP_EIP_ADDR 的定义

这两个宏位于bootloader工程中，但具体的值需要协同app工程来确定。

APP_START_ADDR ： app代码所在ROM区起始地址，定义在app工程的链接文件gcc_flash.ld中。

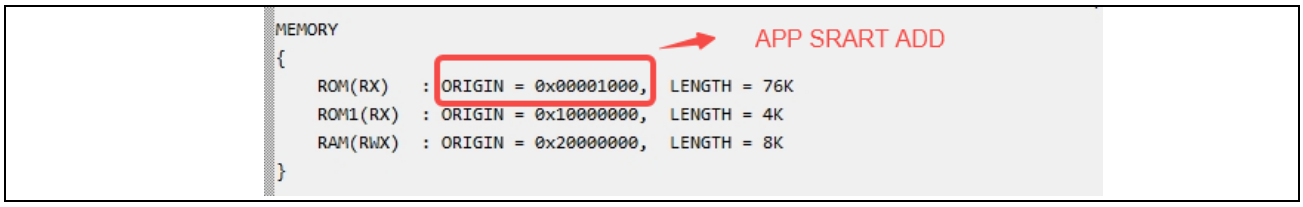

Figure 106 linker文件空间分配

APP_EIP_ADDR ：app的入口地址，在app工程生成的ihex文件中位于倒数第二行，app入口地址为0x1000，如下：

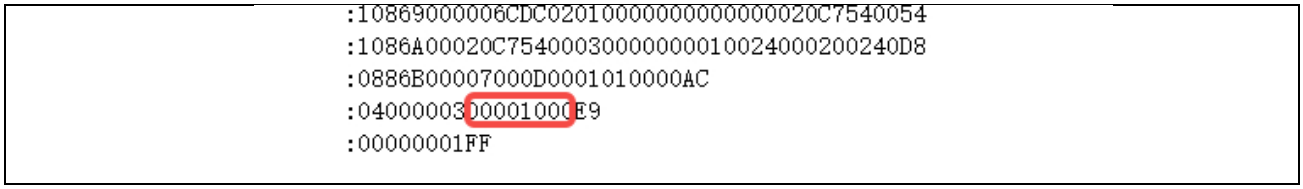

Figure 117 app入口地址

请根据app工程编译出来的实际值，手动更新bootloader工程中的此宏定义值

4.2 如何修改bootloader和app的ROM分区定义？

如果实际应用中bootloader和app空间划分和本demo不一致，需要修改bootloader工程的两个文件bootloader.h 和 gcc_flash.ld以及 APP工程的gcc_flash.ld。

1） bootloader 工程中的 bootloader.h 文件

如果要改bootloader和app的分区地址，那么需要修改 APP_START_ADDR 以及 APP_EIP_ADDR 的定

义。

```
#define APP_START_ADDR          0x1000ul
#define APP_ EIP_ADDR           0x1000ul
#define APP_EIP_VALUE           0x517ul
#define ROM_END_ADDR            0xffff
```

2） bootloader 工程中的 gcc_flash.ld 文件

以下定义ROM区代码需要修改为实际分区值(地址需要1KB对齐)，例如改为4K

```
ROM(RX)    : ORIGIN = 0x00000000,   LENGTH = 4K
```

3） app 项目 gcc_flash.ld 文件

```
ROM(RX)    : ORIGIN = 0x00001000,   LENGTH = 76K   (客户可自行指定)
```

4.3 跳转到app或者bootloader的函数

```
/*****************************************************************************

函数功能：跳转到IAP函数（直接复位mcu 就跳转到了bootloader）

*****************************************************************************/

void bootloader_reset(void)
{
        SYSCON -> RSR = 0xFFFFFFFF;
        SYSCON -> IDCCR = 0xE11E0081; //系统软复位
}
```

```
/*****************************************************************************

函数功能：IAP 跳转到 app 函数

*****************************************************************************/

void bootloader_jump_to_app(void)
{
        asm("mv t2,%0\n"::"r"(APP_START_ADDR):);
        asm(" jalr t2 ");
}
```

4.4 上位机与bootloader通讯异常，不能正常收发数据

1） 首先检查 UART 线是否连接正确，主机的 TX 连接从机的 RX,主机的 RX 连接从机的 TX

2） 查看程序中主从机波特率是否配置一致

波特率从机配置部分：

```
#define BOOTLOADER_PROG_BORD    115200
```

3） 查看主从机升级命令是否一致。

升级命令如下：

```
#define BOOTLOADER_PROG_CHECK     0xF2F3F4F5
#define BOOT_CMD_CHECK             0xEF
#define BOOT_CMD_ACK_CHECK         0xA5

#define ID_START_CHK               0xF5
#define ID_JUMP_BOOT               0x9F
#define ID_START_ADD               0x10
#define ID_END_ADD                 0x20
#define ID_JUMP_ADD                0x30
#define ID_CHECK_NUM               0x40
#define ID_PROG_DATA               0x80
```