使用 STC 的 IAP 系列单片机
开发自己的 ISP 程序

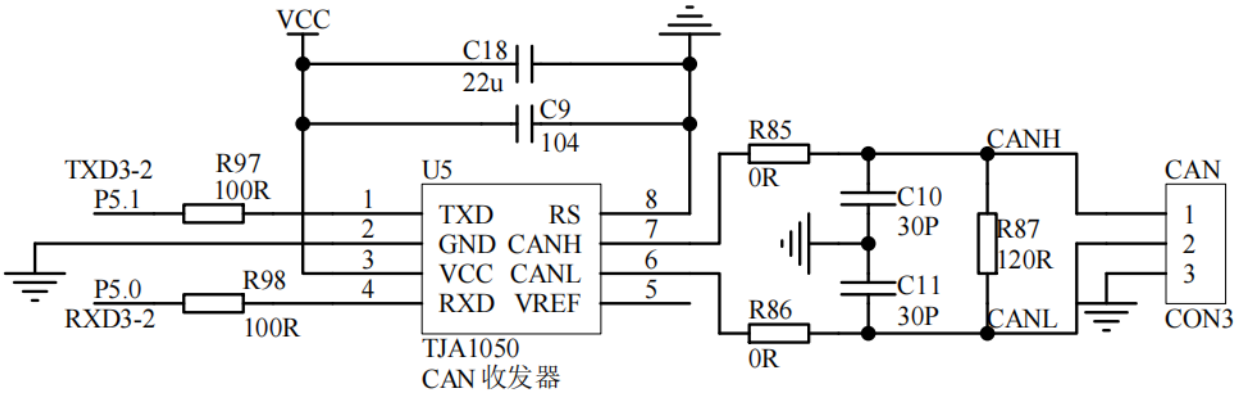
——基于 STC32G12K128

随着 IAP（In-Application-Programming）技术在单片机领域的不断发展，给应用系统程序代码升级带来了极大的方便。STC 的串口 ISP（In-System-Programming）程序就是使用 IAP 功能来对用户的程序进行在线升级的，但是出于对用户代码的安全着想，底层代码和上层应用程序都没有开源，为此 STC 推出了 IAP 系列单片机，即整颗 MCU 的 Flash 空间，用户均可在自己的程序中进行改写，从而使得有用户需要开发自己的 ISP 程序的想法得以实现。

本文以 STC32G12K128 为例，详细说明使用 STC 的 IAP 单片机开发用户自己的 ISP 程序的方法，并给出了基于 Keil 环境的 C 源码。

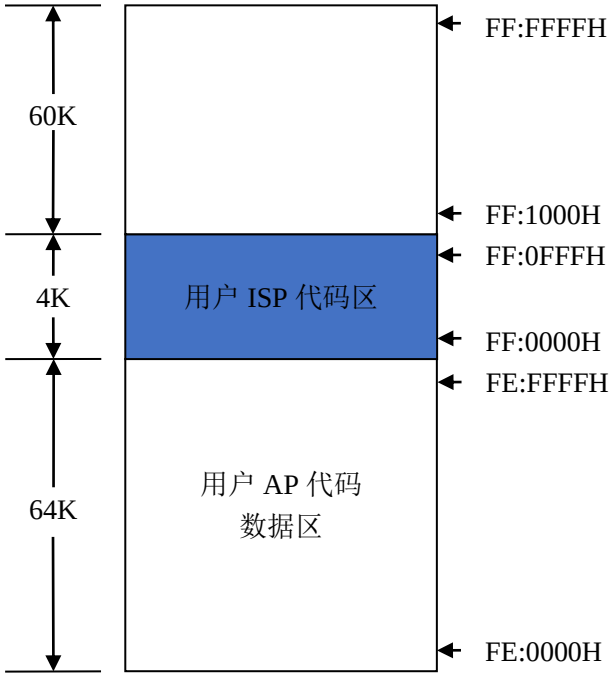
一. 硬件架构

USB 转 CAN 工具与待升级单片机，分别通过 P5.0、P5.1 接口外挂 CAN 收发器连接到 CAN 总线，参考电路如下：



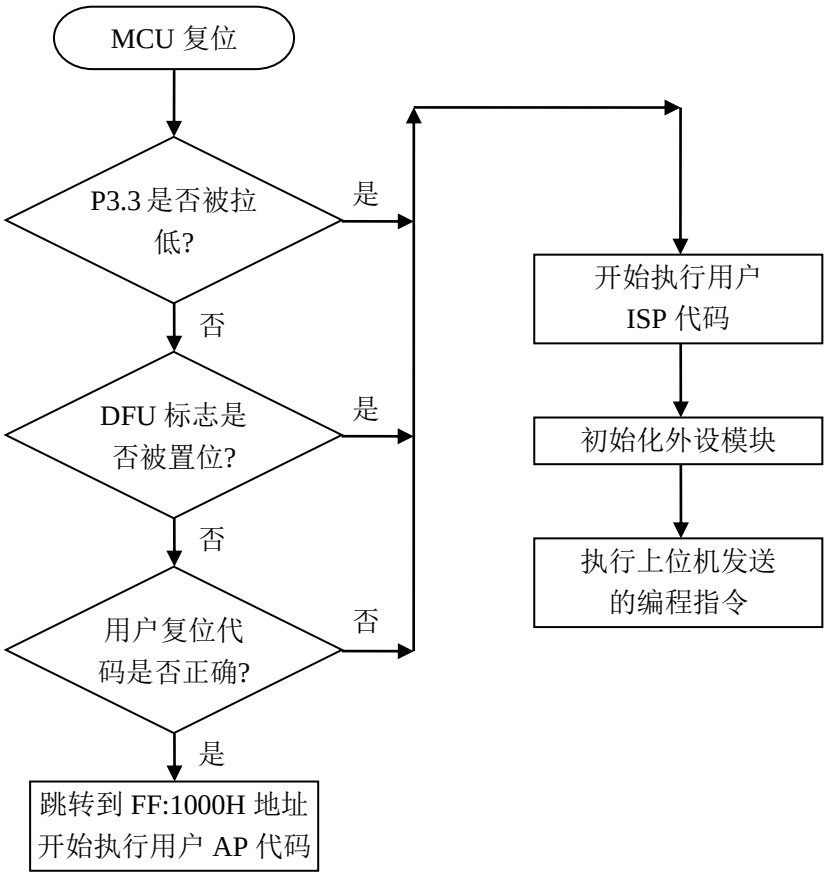
二． 内部 FLASH 规划

示例单片机使用 STC32G12K128，用户可以使用的最大程序空间为 128K 字节，整个 Flash 空间划分如下：



128K 的用户 FLASH 空间中，逻辑地址 FE:0000H~FE:FFFFH 为低 64K 块区，用户可任意使用。逻辑地址 FF:0000H~FF:FFFFH 为高 64K 块区，用户 ISP 代码会占据 FF:0000H～FF:0FFFH 的 4K 字节，剩余的 60K 字节的空间为用户程序区。当满足特定的下载条件时，用户 ISP 会根据上位机发送的指令对除 4K 用户 ISP 区以外的 124K 字节的 FLASH 进行擦除和编程，以达到更新用户程序的目的。

三． 程序的基本框架


MCU 复位
P3.3 是否被拉低?
是
否
DFU 标志是否被置位?
是
否
用户复位代码是否正确?
否
是
跳转到 FF:1000H 地址开始执行用户 AP 代码
开始执行用户 ISP 代码
初始化外设模块
执行上位机发送的编程指令

四． 下位机固件程序说明

下位机固件程序包括三部分：升级工具代码、用户 ISP 代码和用户 AP 代码

- **升级工具代码：**

  升级工具代码主要用于接收上位机的编程指令，转成 CAN 总线数据发送给需要升级 MCU。升级工具代码与上位机的通讯接口为 USB HID 接口，免驱安装。与待升级芯片的通讯接口为 CAN 接口，外挂 CAN 收发器连接到 CAN 总线。

  例程 CAN 总线使用扩展帧收发，CAN ID 包含指令内容，编程指令中 CAN ID 的 bit16~bit0 存放需要编程的 Flash 地址，数据部分存放需要写入 Flash 的内容。

  控制命令 CAN ID：

```
#define CAN_CMD_CONNECT      0x1a000000   //连接指令
#define CAN_CMD_READ         0x1a100000   //读取指令
#define CAN_CMD_PROGRAM      0x1a2xxxxx   //编程指令，其中 xxxxx 为写入地址
#define CAN_CMD_ERASE        0x1a300000   //擦除指令
#define CAN_CMD_REBOOT       0x1a400000   //重启指令
```

  应答命令 CAN ID：

```
#define REP_CMD_CONNECT      0x1b000000   //连接应答
#define REP_CMD_READ         0x1b100000   //读取应答
#define REP_CMD_PROGRAM      0x1b200000   //编程应答
#define REP_CMD_ERASE        0x1b300000   //擦除应答
#define REP_CMD_REBOOT       0x1b400000   //重启应答
```

- **用户 ISP 代码：**

  用户 ISP 代码主要用于接收 CAN 总线的编程指令，对用户 AP 区进行代码更新。用户 ISP 代码的通讯接口为 CAN 接口，外挂 CAN 收发器连接到 CAN 总线。

用户 ISP 代码执行的条件：

1、 P3.3 口被拉到低电平，断电并重新上电后。

当用户需要上电后直接执行用户 ISP，可将 P3.3 口经过 1K 电阻连接到 GND，再对 MCU 进行重新上电，用户 ISP 代码会立即执行。

2、 DFU 标志被置位。

用户 ISP 代码在扩展 RAM 的最后地址（XDATA：1FFCH～1FFFH）定义了一个 4 字节的 DFU 标志变量（DfuFlag）。当 DfuFlag 被赋值为 0x12abcd34 时，用户 ISP 代码也会开始执行。这个功能主要应用于：当 MCU 已经在执行用户 AP 代码后，若用户希望运行用户 ISP 代码更新程序时，则可将 DfuFlag 标志变量赋值为 0x12abcd34，然后软复位（向 IAP_CONTR 寄存器中写 20H），用户 ISP 代码就会开始运行了。

3、 用户 AP 代码的复位指令不规范。

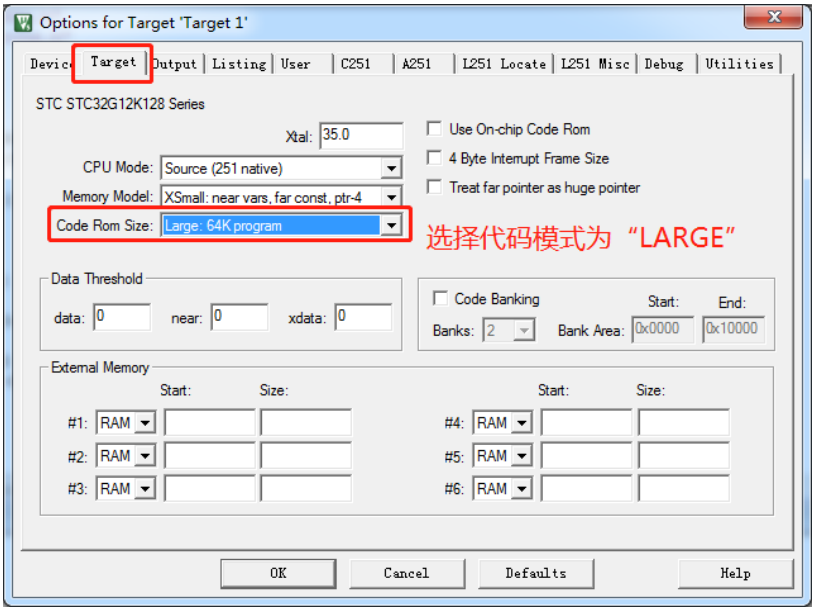
用户 AP 代码的复位代码必须是一条长跳转指令（第一个字节的指令码必须为 02H），且长跳转的地址必须跨过用户 ISP 代码的 FF:0000H～FF:0FFFH 的 4K 空间。否则用户 AP 的复位代码不规范，无法正常运行用户 AP 代码，此时会强制执行用户 ISP 代码。这种情况一般只会出现在第一次只单独下载了用户 ISP 代码，而没有用户 AP 代码的时候。

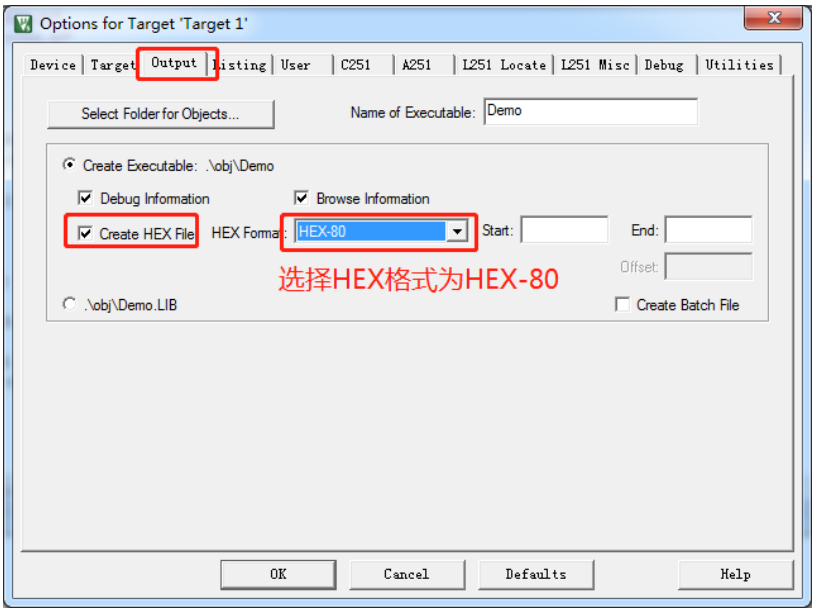
- **用户 AP 代码**

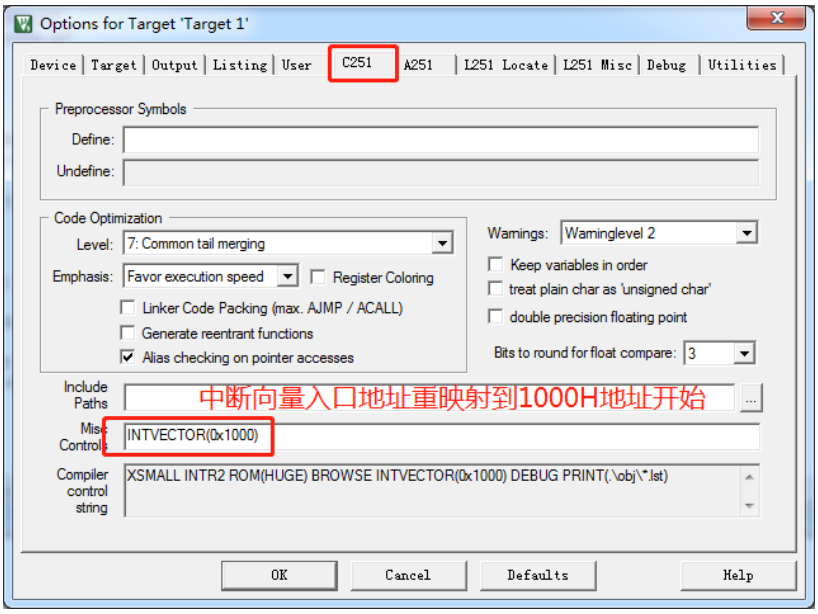
用户 AP 代码是用户的正常功能代码。由于用户 ISP 代码使用了 FF:0000H～FF:0FFFH 的 4K 空间，用户的 AP 代码必须从 FF:1000H 开

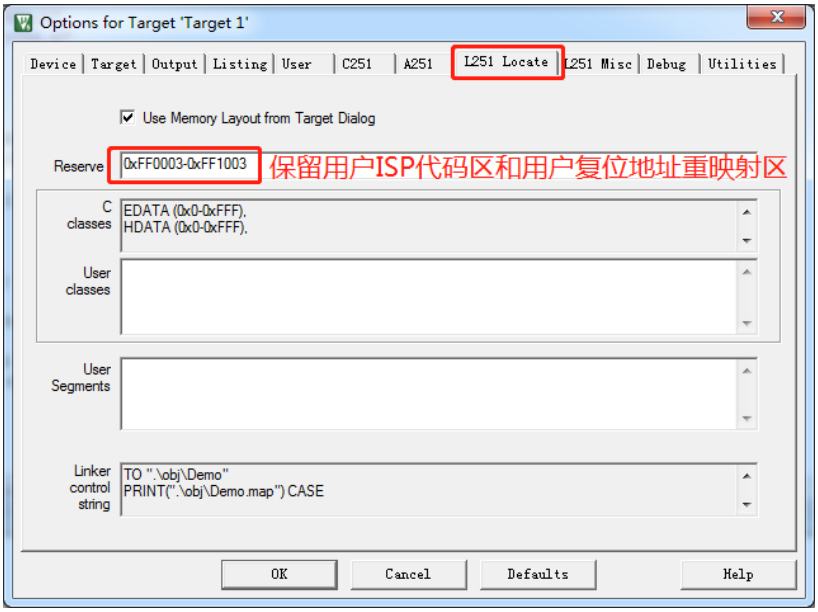
始执行，用户 AP 代码原本位于 FF:0000H～FF:0002H 的复位跳转指令被重映射到 FF:1000H～FF:1002H 的地址（重映射的工作上位机应用程序会自动处理，用户在编写 AP 代码时无需关心）。另外单片机的中断入口地址也在用户 ISP 代码的 4K 空间以内，也需要重映射到 FF:1000H 开始的地方，这个重映射需要在 Keil 软件中对项目进行一些简单设置即可。

用户 AP 代码大小小于 60K 的项目设置

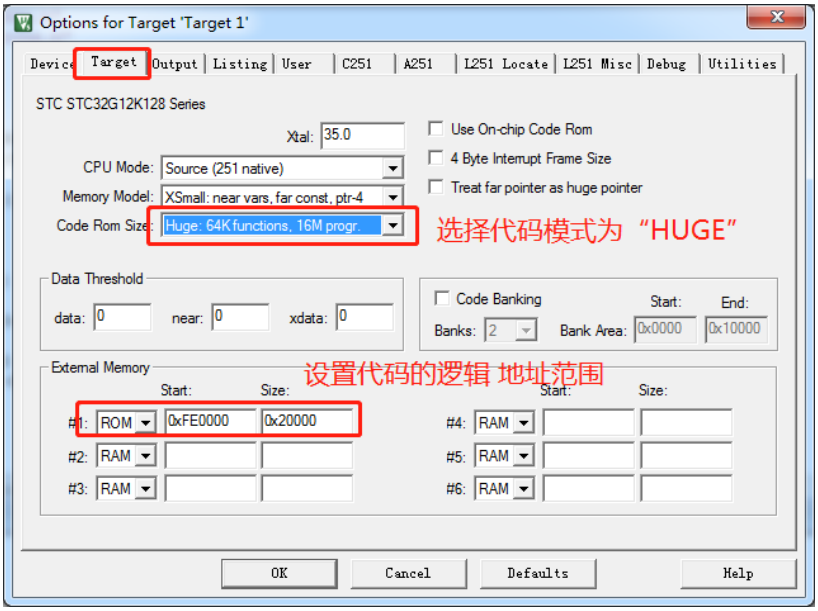


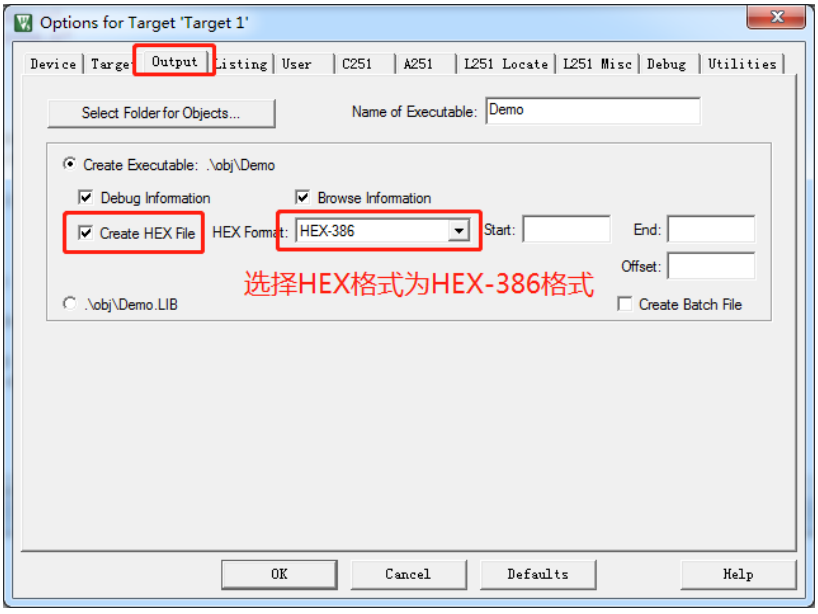


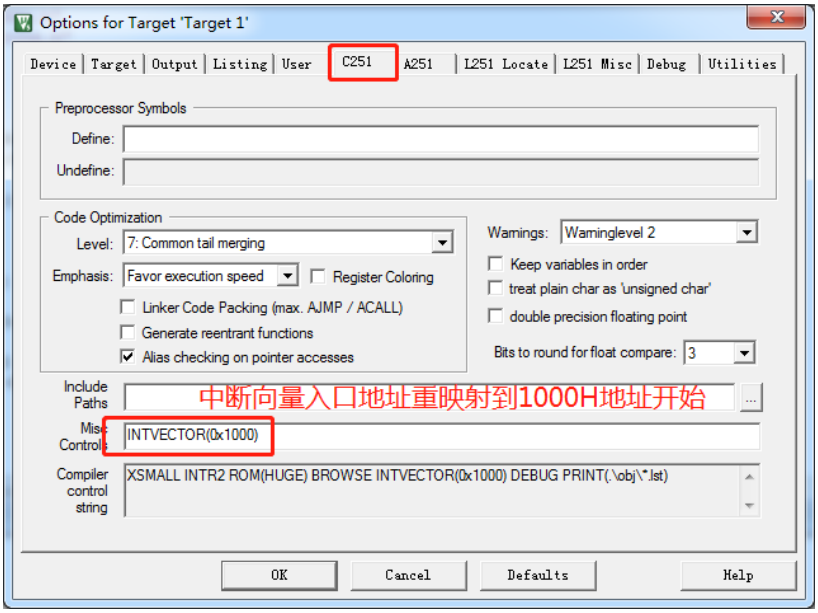

Options for Target 'Target 1'
Device
Target
Output
Listing
User
C251
A251
L251 Locate
L251 Misc
Debug
Utilities
Preprocessor Symbols
Define:
Undefine:
Code Optimization
Level:
7: Common tail merging
Emphasis:
Favor execution speed
Register Coloring
Linker Code Packing (max. AJMP / ACALL)
Generate reentrant functions
Alias checking on pointer accesses
Warnings:
Warninglevel 2
Keep variables in order
treat plain char as 'unsigned char'
double precision floating point
Bits to round for float compare:
3
Include Paths
中断向量入口地址重映射到1000H地址开始
Misc Controls
INTVECTOR(0x1000)
Compiler control string
XSMALL INTR2 ROM(HUGE) BROWSE INTVECTOR(0x1000) DEBUG PRINT(.\obj*.lst)
OK
Cancel
Defaults
Help

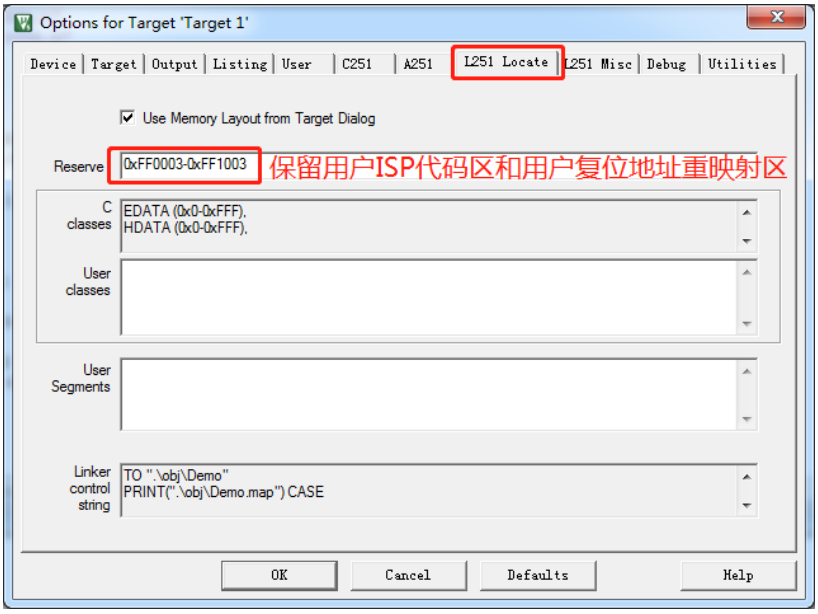

Options for Target 'Target 1'
Device
Target
Output
Listing
User
C251
A251
L251 Locate
L251 Misc
Debug
Utilities
Use Memory Layout from Target Dialog
Reserve
0xFF0003-0xFF1003
保留用户ISP代码区和用户复位地址重映射区
C classes
EDATA (0x0-0xFFF),
HDATA (0x0-0xFFF),
User classes
User Segments
Linker control string
TO ".\obj\Demo"
PRINT(".\obj\Demo.map") CASE
OK
Cancel
Defaults
Help

用户 AP 代码大小大于 60K 的项目设置









若需要在用户 AP 代码中实现软复位到用户 ISP 升级程序的功能，则需要用户端 AP 代码进行如下所示的几点设置。

```
#include "stc32g.h"
#include "intrins.h"

typedef unsigned char u8;
typedef unsigned int u16;
typedef unsigned long u32;

#define FOSC            24000000UL
#define T1MS            (65536 - FOSC/1000)

#define DFU_TAG         0x12abcd34      //DFU强制执行标志
long xdata DfuFlag _at_ 0x1ffc;         //DFU标志，定义在xdata的最后4字节
```

```
        if (P32 == 0)
        {
            DfuFlag = DFU_TAG;          //当需要执行用户ISP代码时，将强制执行标志赋值到DFU标志变量中
            IAP_CONTR = 0x20;           //然后执行软复位
        }

        if(B_Can1Read)
        {
            B_Can1Read = 0;

            CANSEL = 0;         //选择CAN1模块
            n = CanReadMsg(CAN1_Rx);    //读取接收内容
            if(n>0)
            {
                for(i=0;i<n;i++)
                {
                    if(CAN1_Rx[i].ID == 0x1a000000)  //升级握手命令
                    {
                        DfuFlag = DFU_TAG;  //当需要执行用户ISP代码时，将强制执行标志赋值到DFU标志变量中
                        IAP_CONTR = 0x20;   //然后执行软复位
                    }
                }
            }
        }
```

```
    TMOD = 0x00;
    T0x12 = 1;
    TL0 = T1MS;
    TH0 = T1MS >> 8;
    TR0 = 1;
    ET0 = 1;

    DfuFlag = 0;                        //上电正常执行用户AP，时需要将DFU标志清零
    cnt200 = 0;

    EA = 1;
```

当检测到满足用户 ISP 下载条件后，只需要将 DFU_TAG 赋值到 DfuFlag 变量中，然后软复位即可。

五． 上位机应用程序说明

上位机演示程序是基于 MFC 的对话框项目，对于 USB 的 HID 接口的访问是直接调用 Windows 的 API 函数。界面较简单，只是为这一功能的实现提供了一个框架，其他的功能及要求均还可以往上面添加。

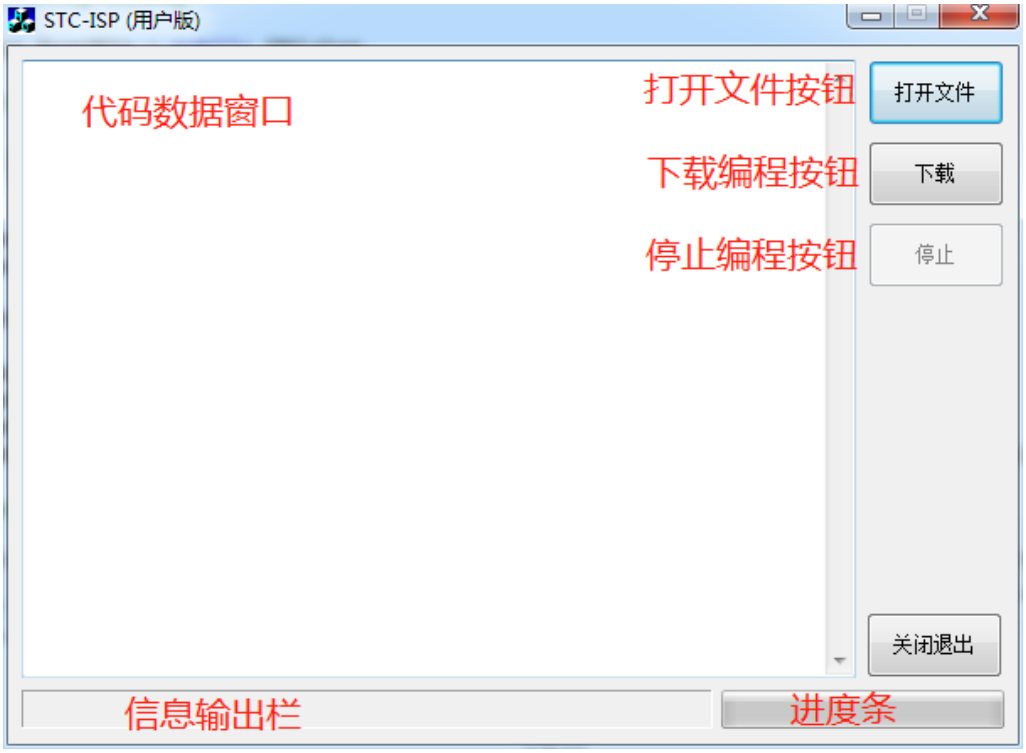
上位机程序的核心模块是基于类 CStcIsp_UserDlg 的一个静态线程函数“static UINT Download(LPVOID pParam);”

```
////////////////////////////////////////////////////////////////////////////
// CStcIsp_UserDlg dialog

class CStcIsp_UserDlg : public CDialog
{
// Construction
public:
    CStcIsp_UserDlg(CWnd* pParent = NULL);  // standard constructor
    virtual ~CStcIsp_UserDlg();

    void ShowMessage(LPCTSTR lpMsg, ...);
    BOOL LoadCode(LPCTSTR lpszFile, BOOL bHex);

    static UINT Download(LPVOID pParam);

    BOOL HidOpen(WORD VID = 0x34bf, WORD PID = 0xff01);
    void HidClose();
    BOOL HidRead(BYTE *pData, DWORD dwInterval = 100);
    BOOL HidWrite(BYTE bCmd, DWORD dwAddress, BYTE bSize, BYTE *pData, DWORD dwInterval = 100);

// Dialog Data
    //{{AFX_DATA(CStcIsp_UserDlg)
    enum { IDD = IDD_STCISP_USER_DIALOG };
    CStatic m_ctlMessageStatic;
```

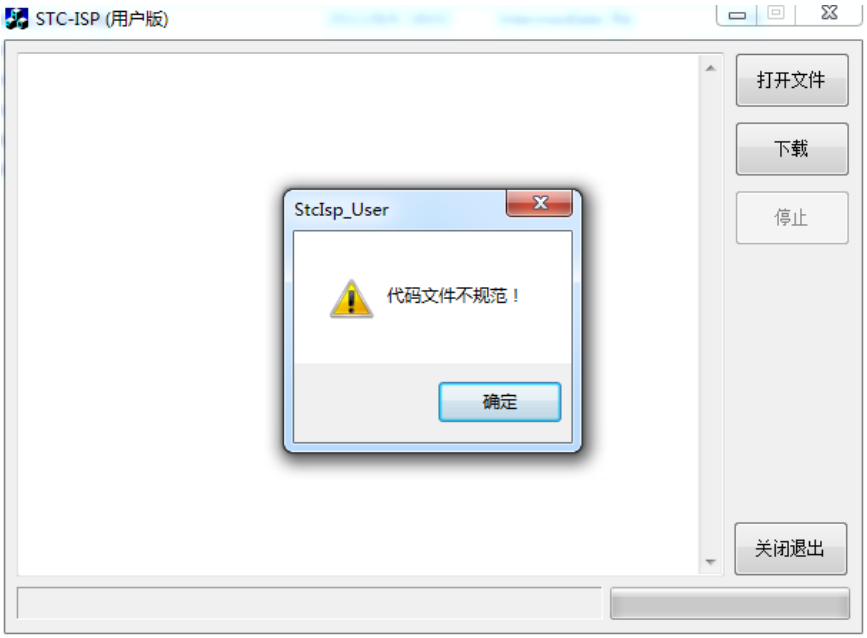
此函数负责与下位机通讯，发送各种通讯命令来完成对用户程序的更新。用户可以根据各自不同的需求增加命令。

六． 上位机应用程序的使用方法

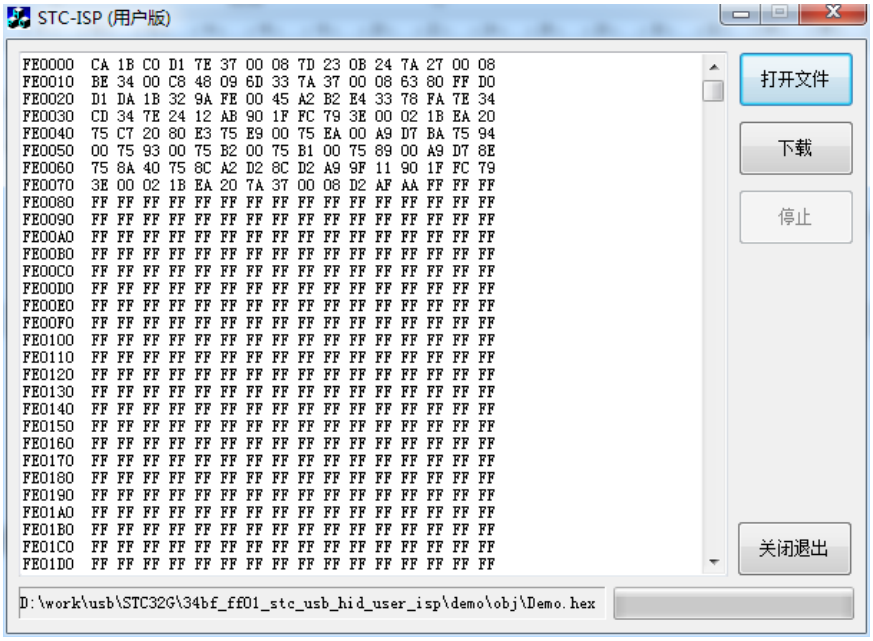
1、打开上位机界面，如下图



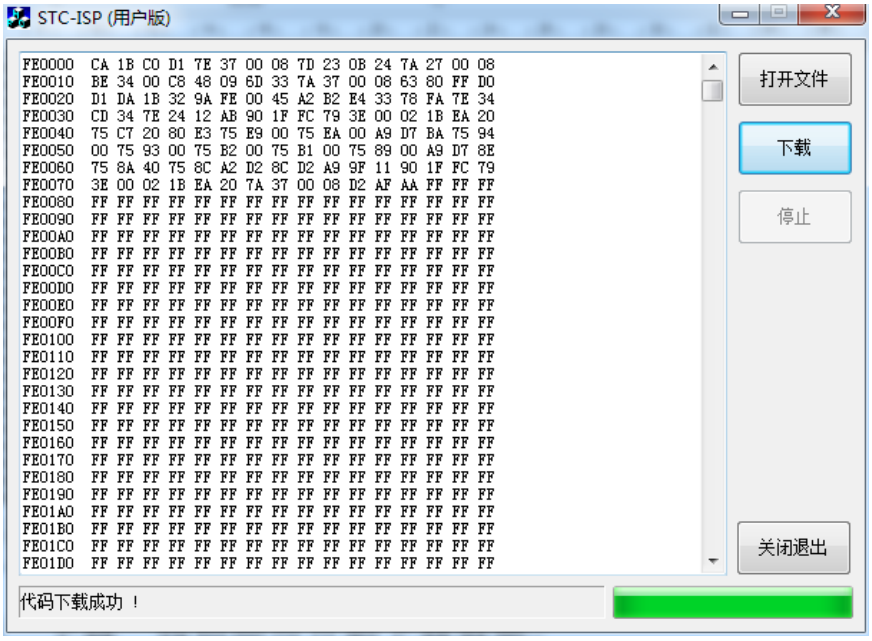
2、点击“打开文件”按钮来打开要下载的源数据文件，Bin 或者 Intel hex 格式均可以。注意用户 AP 代码项目一定要按照前面所介绍的方式进行建立和设置，否则格式不符合规范就会弹出如下弹窗。



打开格式规范的代码文件，如下图所示：

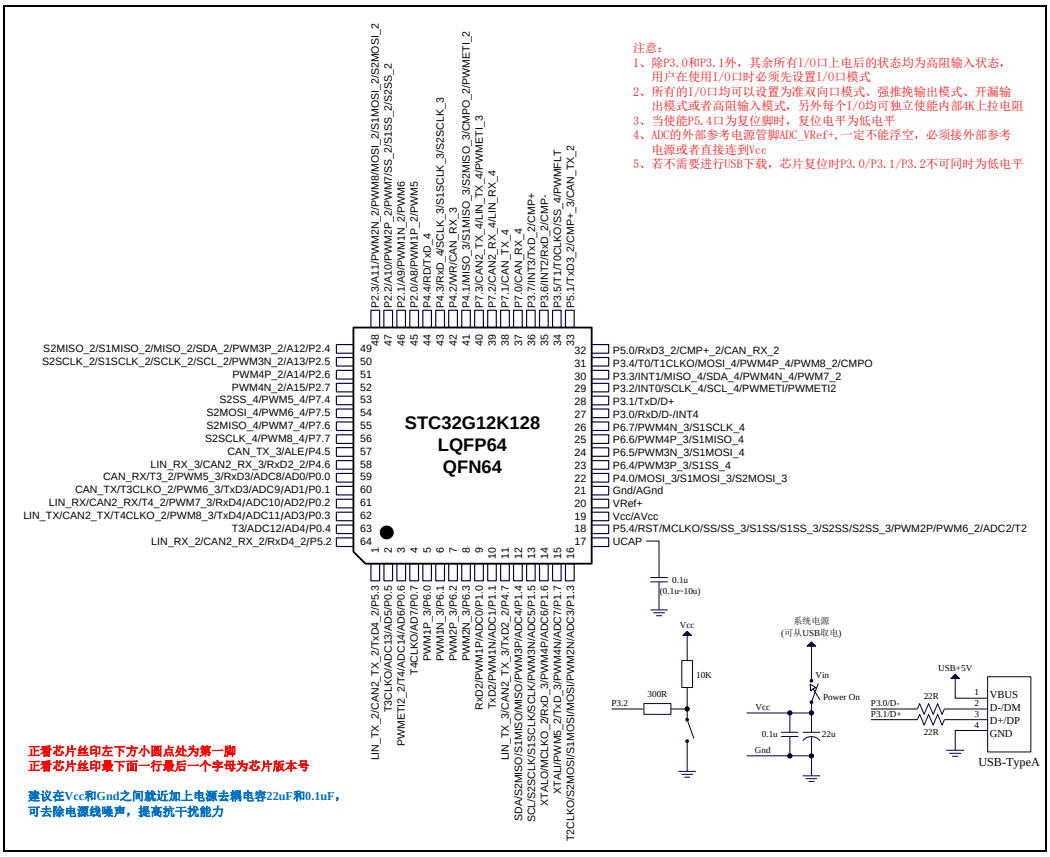


3、点击“下载”按钮即可开始下载数据



七. 整体操作流程

1、 烧录升级工具，首先参考线路图，将单片机和电脑的 USB 口连接好。



USB-ISP 下载程序步骤：

1、按下板子上的 P3.2/INT0 按键，就是 P3.2 接地

2、给目标芯片重新上电，不管之前是否已通电。

===电子开关是按下停电后再松开就是上电

等待 STC-ISP 下载软件中自动识别出“STC USB Writer (HID1)”，识别出来后，就与 P3.2 状态无关了，这时可以松开 P3.2 按键

===传统的机械自锁紧开关是按上来停电，按下去是上电

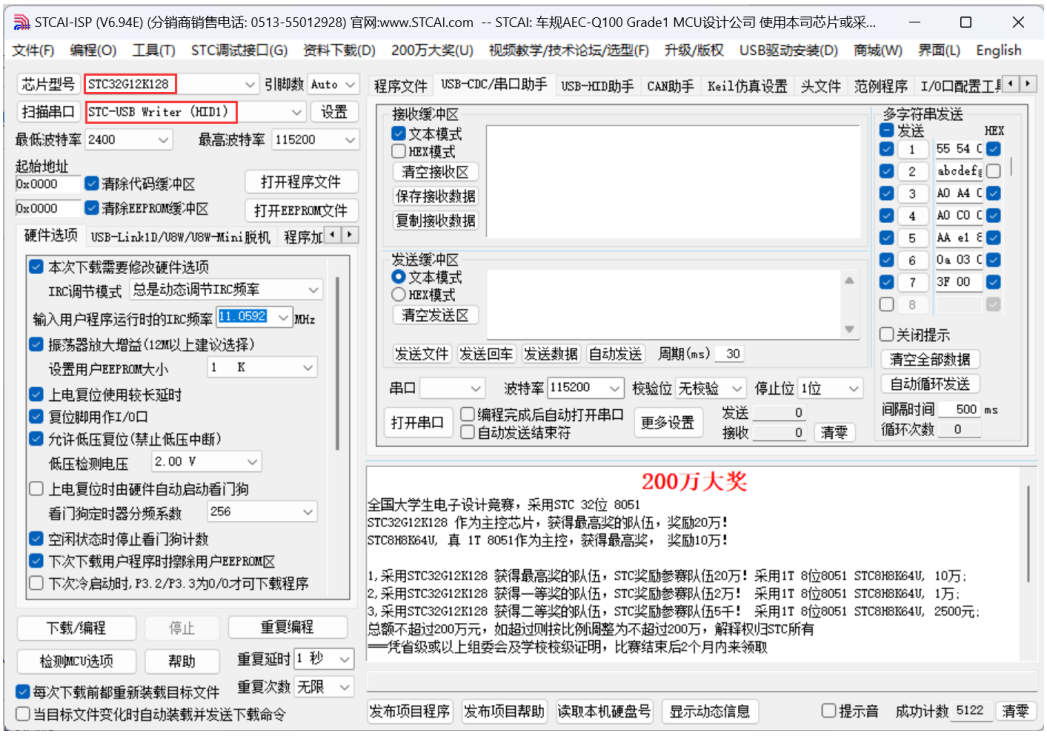
3、点击软件中的“下载/编程”按钮（注意：USB 下载与串口下载的操作顺序不同）下载成功 ！

===另外从用户区软复位到系统区也是等待 USB 下载。

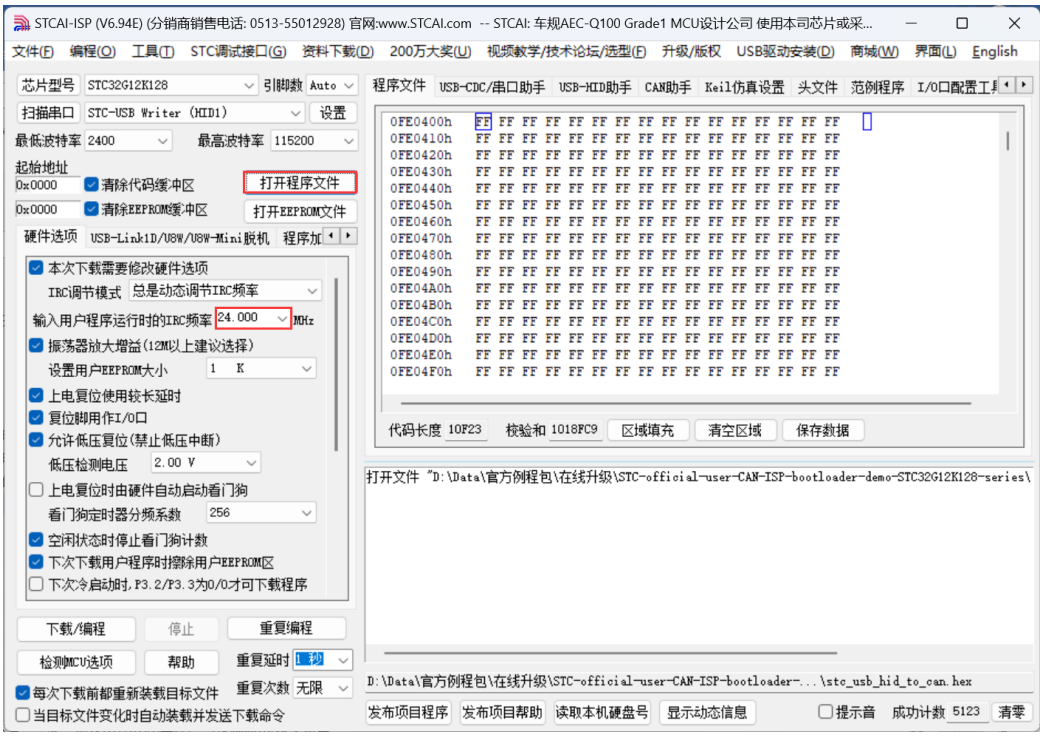
2、 打开最新的 STC-ISP 下载软件，选择“STC32G12K128”目标单片机型号。

3、 按下线路图中的 P3.2 口的按键不要松开，然后按下线路图中的电

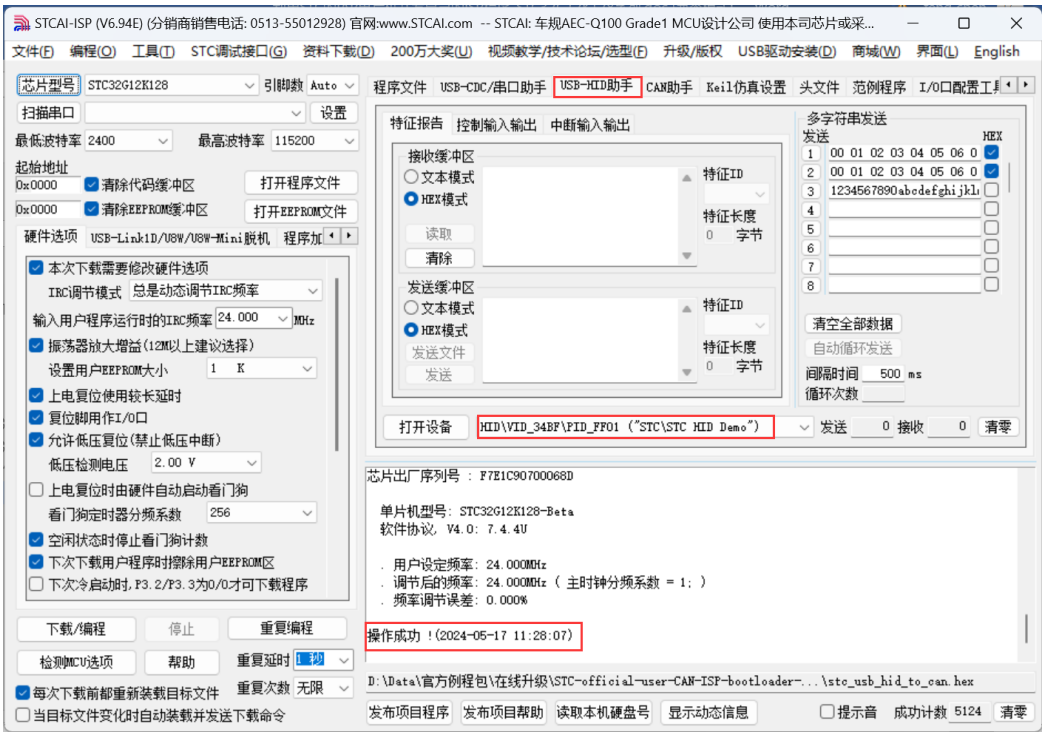
源按键断电，再松开电源按键，目标板重新上电，此时芯片会进入 STC 的 USB 下载模式，如下图：



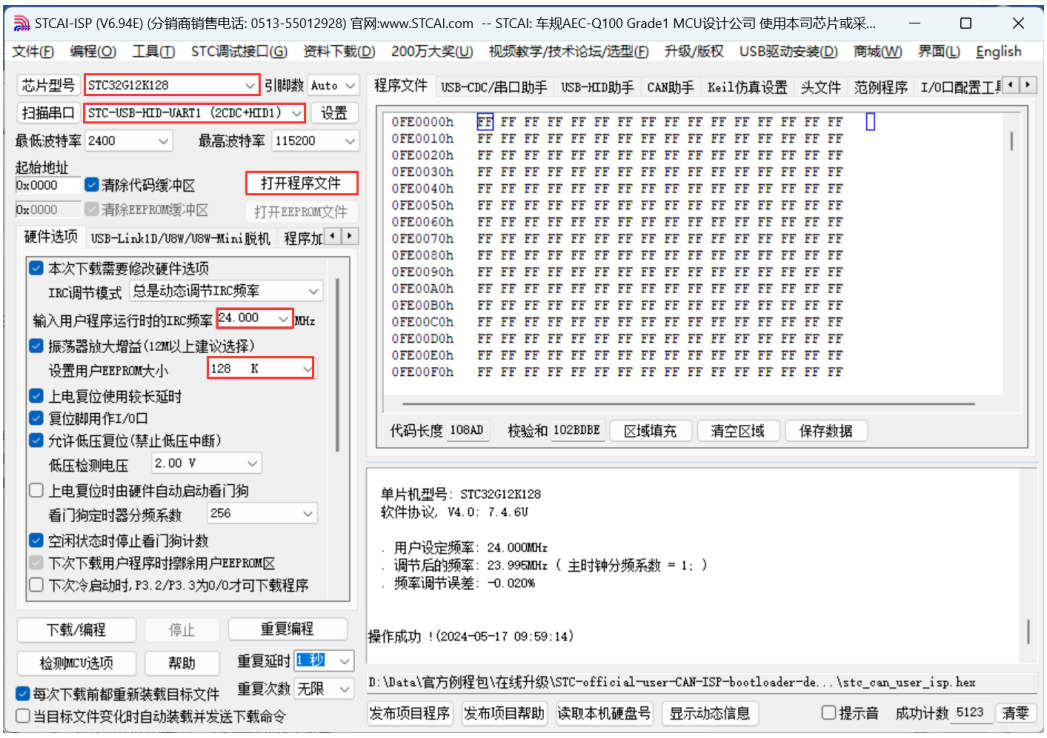
4、 打开范例程序包中的“\tool\obj\stc_usb_hid_to_can.hex”升级工具代码 hex 文件，并按照如下图所示设置硬件选项，工作频率为 24MHz



5、 点击下载/编程按钮，将升级工具代码下载到目标单片机中（下载成功后在 USB-HID 助手里面可以看到 STC HID Demo 设备）



6、 在需要更新功能的目标单片机里烧录用户 ISP 程序，打开范例程序包中的“\isp\obj\stc_can_user_isp.hex”用户 ISP 代码 hex 文件，并按照如下图所示设置硬件选项，工作频率为 24MHz、EEPROM 大小为 128K（此项很重要）

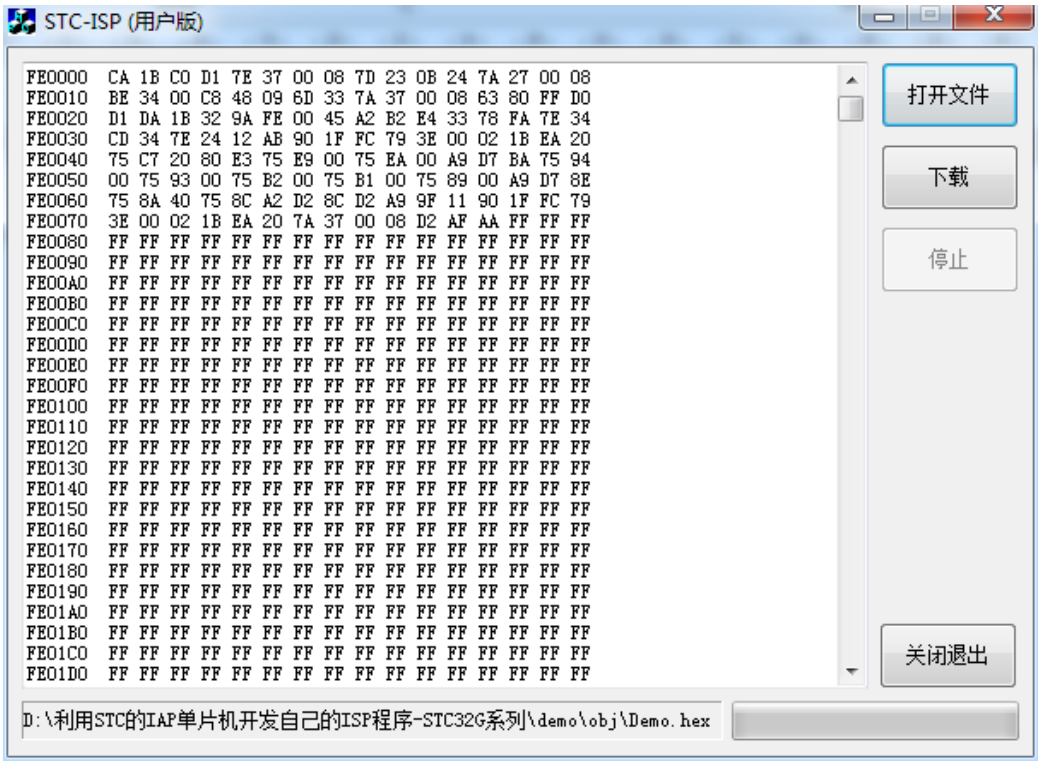


通过 USB 接口或者烧录器烧写程序到芯片里。

7、 给目标单片机重新断电并上电一次，设置的 EEPROM 大小的硬件选项必须重新上电后才能生效（重要，容易被忽略），此时还没有用户 AP 代码，用户 ISP 代码被强制运行。

8、 参考第一章硬件架构连接好烧录工具与目标芯片。

9、 打开范例程序包中的“\app\StcIsp_User.exe”，点击“打开文件”按钮，打开范例程序包中的“\demo\obj\Demo.hex”



10、 点击“下载”按钮，即可完成用户 AP 代码的更新

若能够正确下载，下载完成后，会显示如下图所示的“代码下载成功”画面