应用笔记

在 CW32F030 上实现 IAP 功能

版本号：Rev 1.0

前言

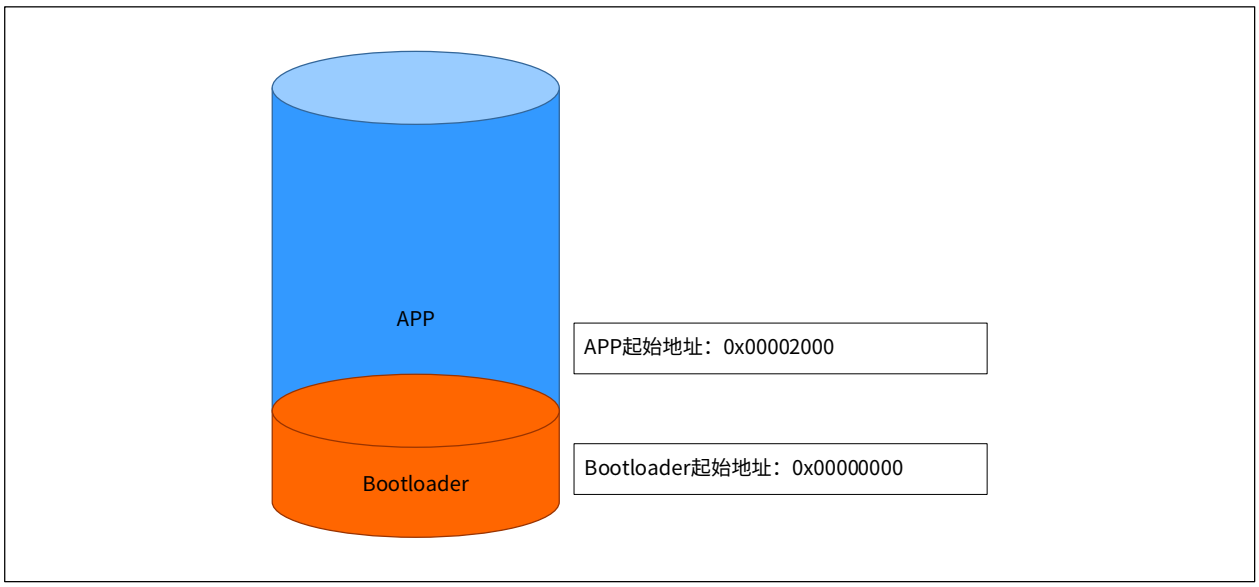
IAP 是 In Application Programing 的首字母缩写，在应用编程，即在程序运行的过程中进行编程（升级程序，更新固件）。

IAP 是用户自己的程序在运行过程中对 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。

目录

1 功能实现

IAP 功能的实现，一般将程序分为两个部分，即：Bootloader 和 APP。Bootloader 程序用于上电时判断程序是进入 IAP 升级流程还是进入 APP 应用程序执行过程。其一般放置在 FLASH 的开始部分，如下图示：

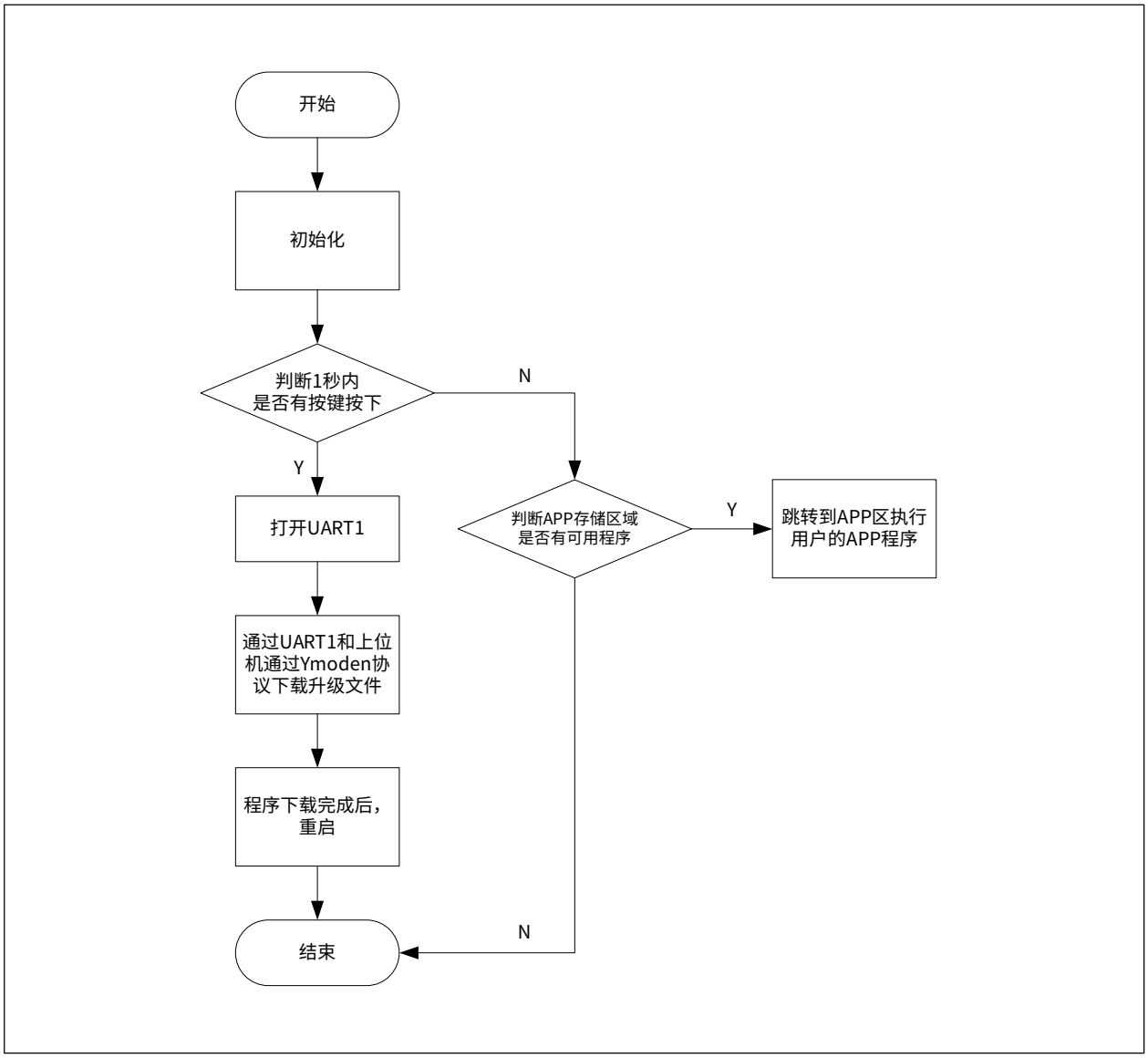


CW32F030 的 Flash 起始地址为 0x00000000，划分 0x00000000 ~ 0x00001FFFF 这 8Kbyte 的空间用于存放 Bootloader 程序，APP 的起始地址选择为 0x00002000。

CW32F030 采用 ARM® Cortex®-M0+ 内核，具有向量表偏移寄存器 VTOR，故其从 Bootloader 中实现向 APP 中跳转较为简单，仅设置向量表偏移即可。

本应用中 Bootloader 采用 UART1 作为 IAP 的通讯接口，以 Ymodem 协议进行文件的传输。APP 程序通过 SYSTICK 定时器计时，并通过中断的方式每 200ms 对口线 PB09 翻转一次，驱动 LED1 闪烁。

2 Bootloader 程序设计



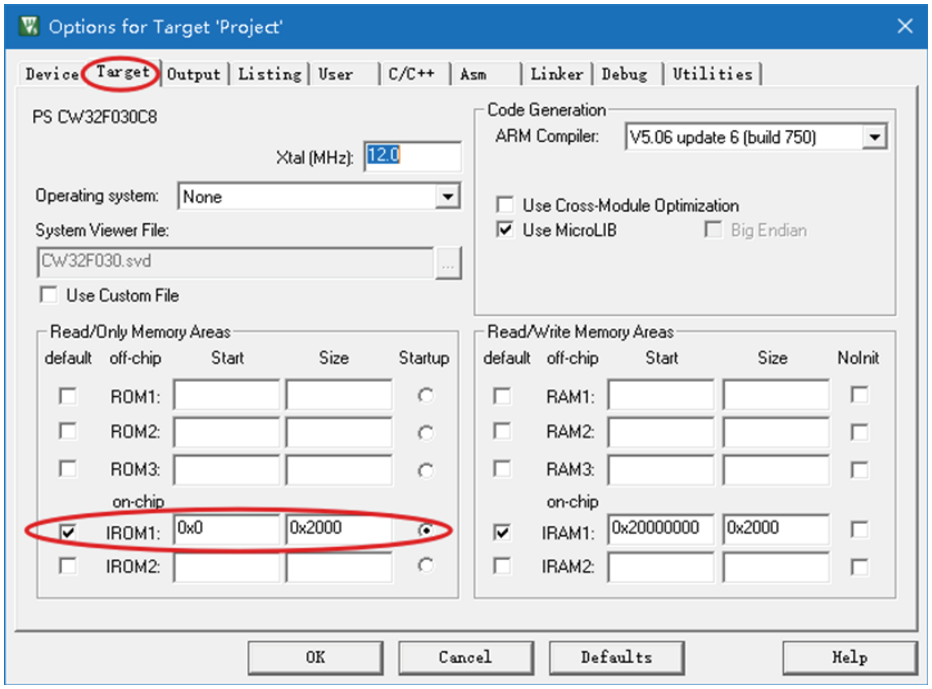
2.1 参考代码

```
int main(void)
{
    volatile uint32_t u32Ticks, u32ElapsedTicks;

    RCC_Configuration();        // 配置系统的工作时钟，SYSCLK=HCLK=PCLK=64MHz
    InitTick(SystemCoreClock);    // 配置 SYSTICK 频率为 1ms
    GPIO_Configuration();       // 配置 PA01 引脚为按键 KEY1 的输入

    u32Ticks = GetTick();
    do
    {
        u32ElapsedTicks = GetTick() - u32Ticks;
        if (!PA01_GETVALUE())   // 检测按键
        {  // 按下按键
           break;
        }
    } while(u32ElapsedTicks < 1000);   // 等待 1s
    if (u32ElapsedTicks < 1000)
    {  // 1s 内有按键按下，进入串口升级流程
        UART1_Configuration();  // 配置串口，波特率 115200
        SerialDownload();       // 通过 YMODEM 协议下载升级程序
    }
    else
    { // 超时，从 boolloader 程序向用户 APP 程序跳转
        __disable_irq();  // 关中断

        if (((*(__IO uint32_t*)ApplicationAddress) & 0x2FFE0000 ) == 0x20000000)  // 判断跳转的地址是否有合法程序存在
        {
            // 向用户的 APP 程序进行跳转
            JumpAddress = *(__IO uint32_t*) (ApplicationAddress + 4);   // ResetHandle 函数的地址
            Jump_To_Application = (func_ptr_t) JumpAddress;   // 将地址强制转换为函数指针
            __set_MSP(*(__IO uint32_t*) ApplicationAddress);  // 设置用户 APP 程序的栈地址
            Jump_To_Application();   // 跳入用户 APP 程序的 ResetHandle 处
        }
    }
    while (1);
}
```

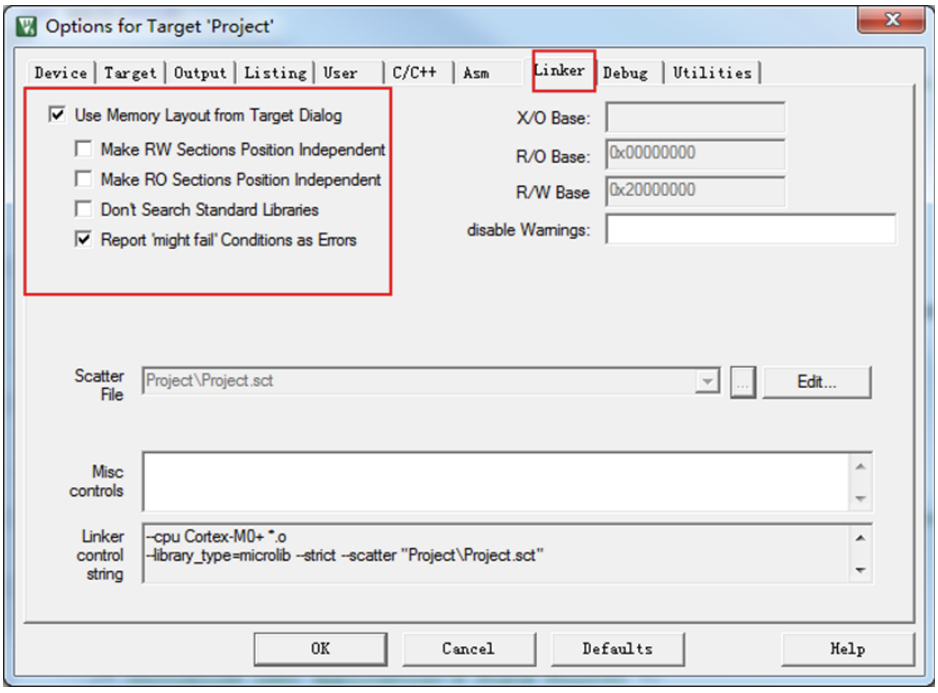
2.2 程序编译

以 MDK-ARM 为例，编译时注意如下选项：

1. 设置 Bootloader 程序的起始地址和占用 ROM（FLASH）空间的大小，本例中 Bootloader 从地址 0x00000000 处执行，占用 7.46Kbyte 的空间，故分配 8Kbyte（size=0x2000）的 FLASH 空间保留给 bootloader。如下图：



2. 链接时，使用 IDE 的对话框配置，如下图：

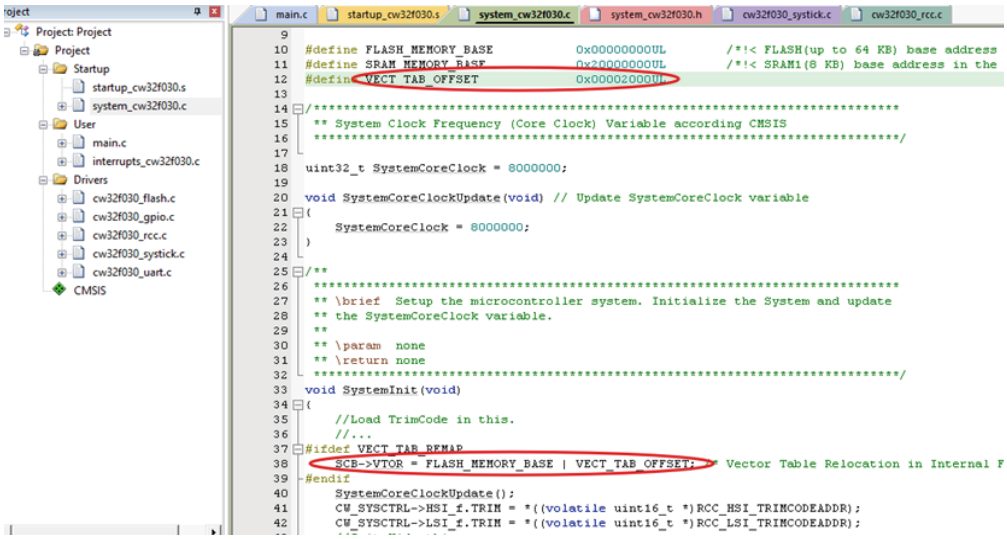


3 APP 程序设计

APP 程序可以先按正常程序的设计流程进行设计和调试，等调试测试通过后，需要进行如下的修改：

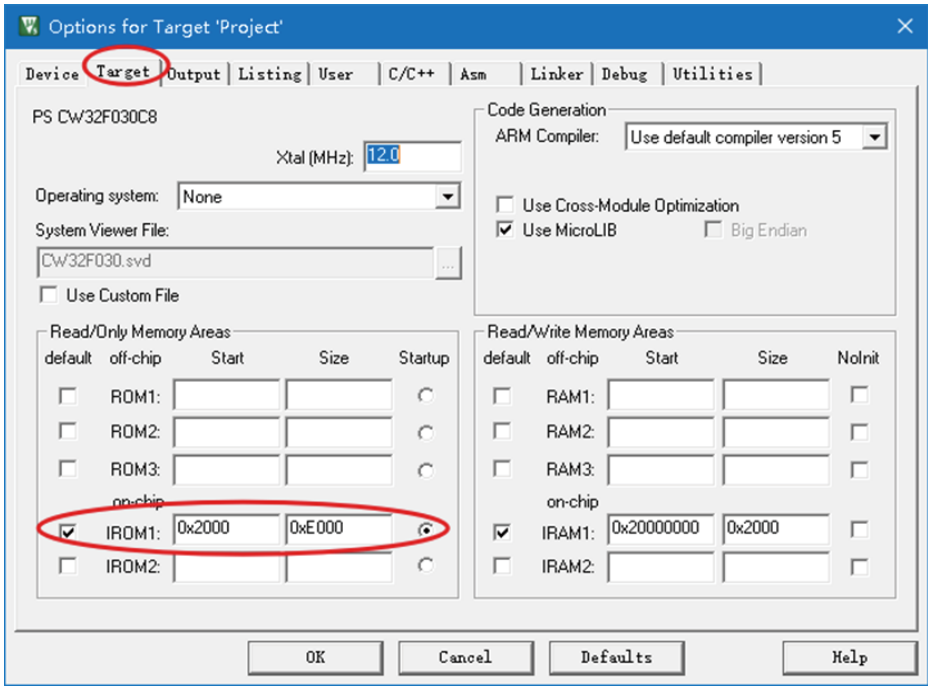
1. 在对系统初始化时，修改中断向量表的偏移地址

   如下图将 VECT_TAB_OFFSET 修改为 0x00002000，这个偏移量就是用户的程序准备在 FLASH 中存放的地址，本例设置为 0x00002000。这样用户程序的中断向量表就被重定位到 0x00002000 处，与 ARM® Cortex®-M0+ 内核默认的中断向量表地址 0x00000000 相区分。

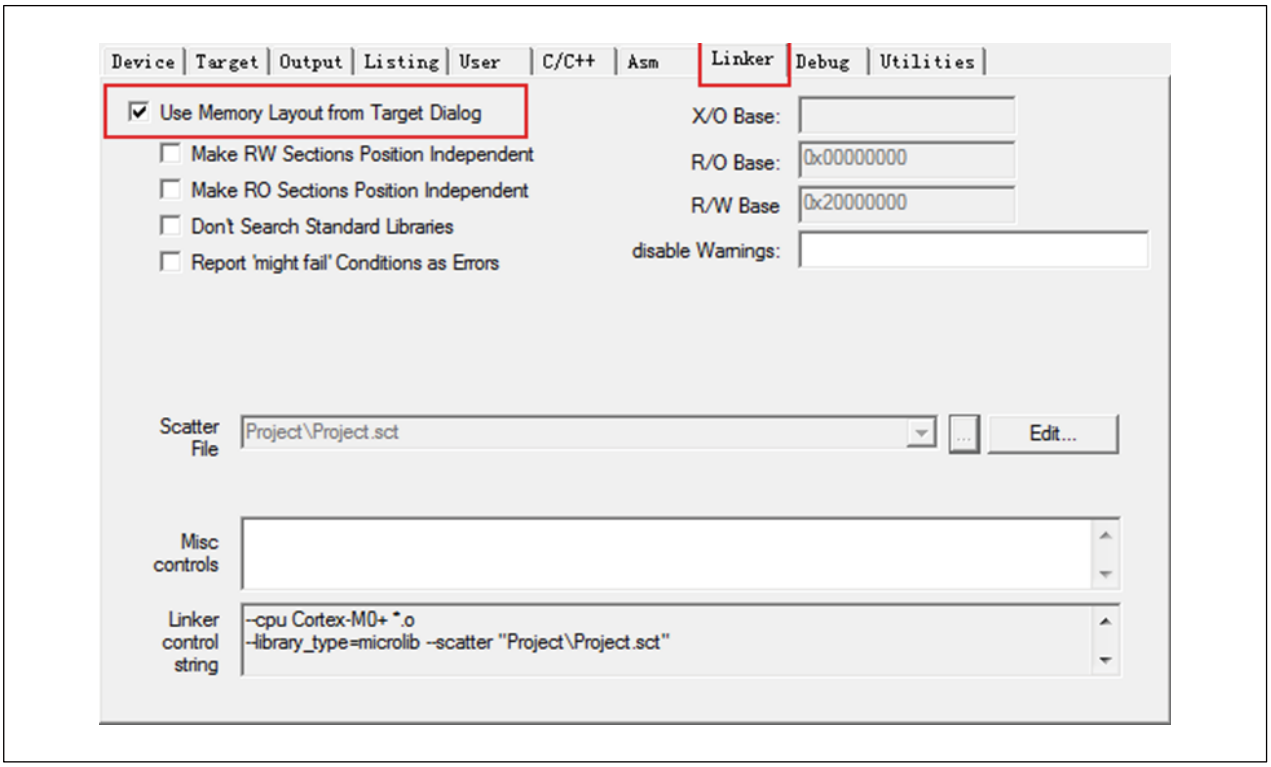


2. 修改编译配置

   如下图，START 填入的值，即为 APP 程序在 FLASH 中存放的起始地址，也是中断向量表需要偏移的位置。



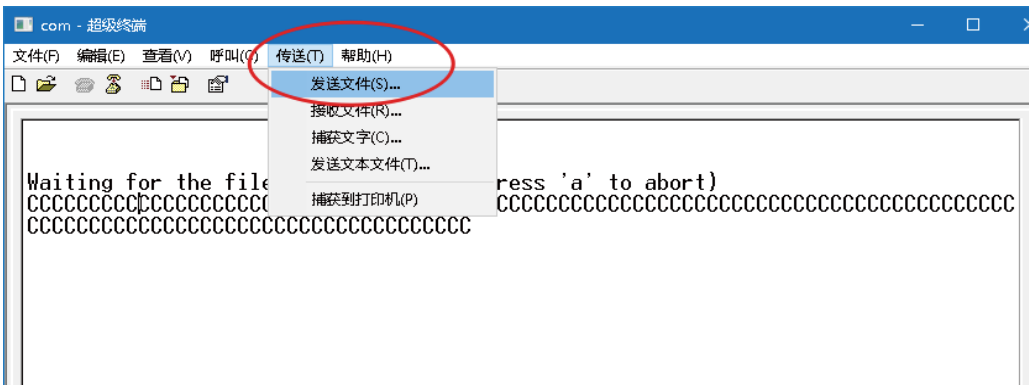
3. 链接时，使用 IDE 的对话框配置，如下图：



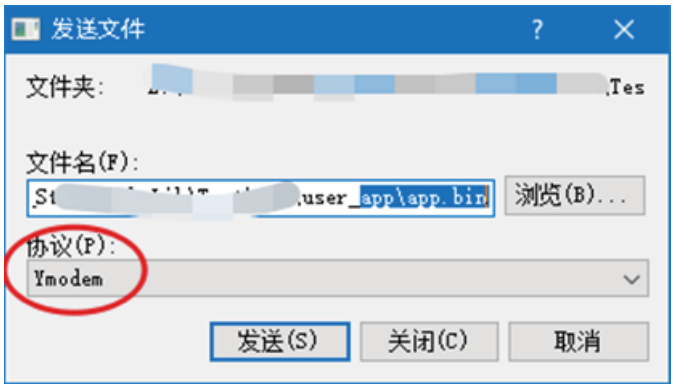
4 演示

上位机采用 Windows 的超级终端，设置串口波特率为 115200bps，8 位数据位，1 位停止位。下位机 CW32F030C8T6 StarKit 复位后，按下 KEY1 后，超级终端将有如下显示：

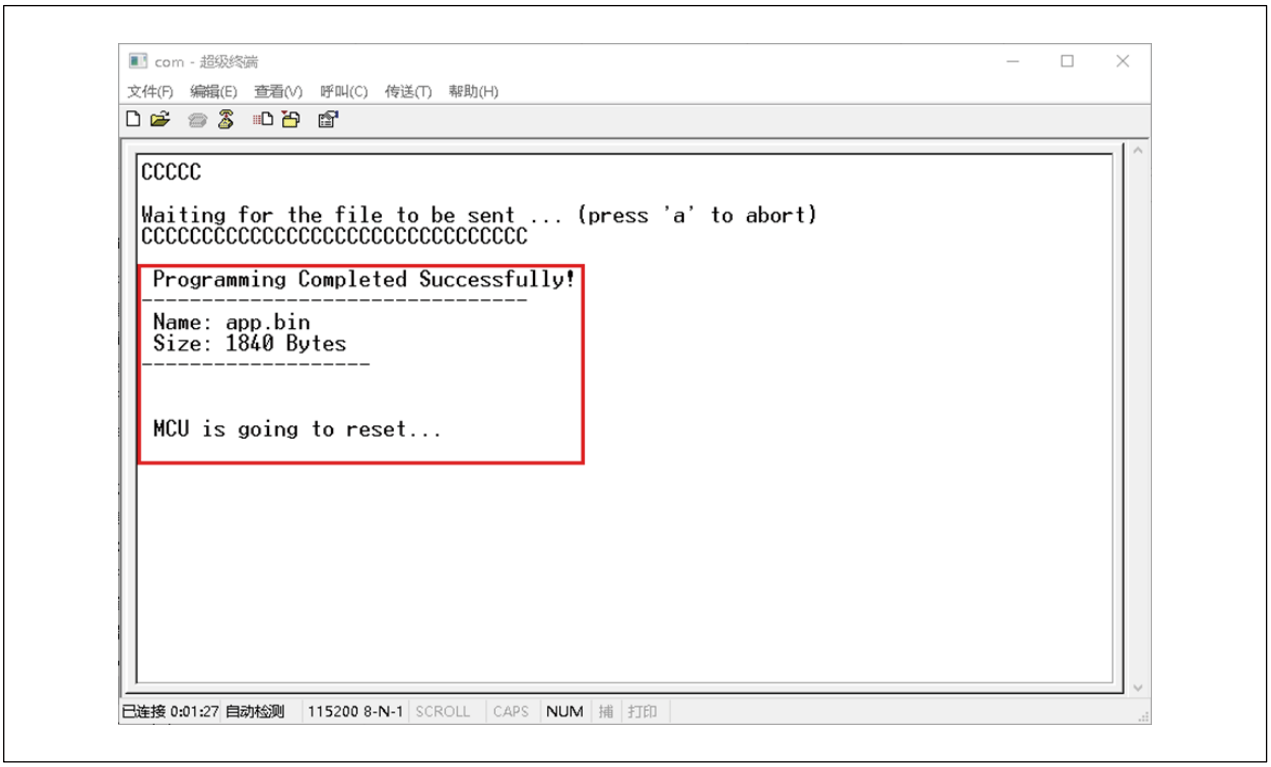
屏幕不断显示字符“C”，提示下位机准备接受文件。选择“传送”-“发送文件”，如下图：



在出现的对话框中选取用户的 APP 文件，并将协议选择为“Ymodem”，并点击“发送”，如下图：



程序传输完成后，屏幕将有如下提示：



此时，CW32F030C8T6 StarKit 板上的 LED1 将以 200ms 的间隔闪烁，说明用户程序已正常运行。

5 版本信息

表 5-1 文档修订信息

日期	版本	变更信息
2022-03-31	Rev 1.0	初始发布