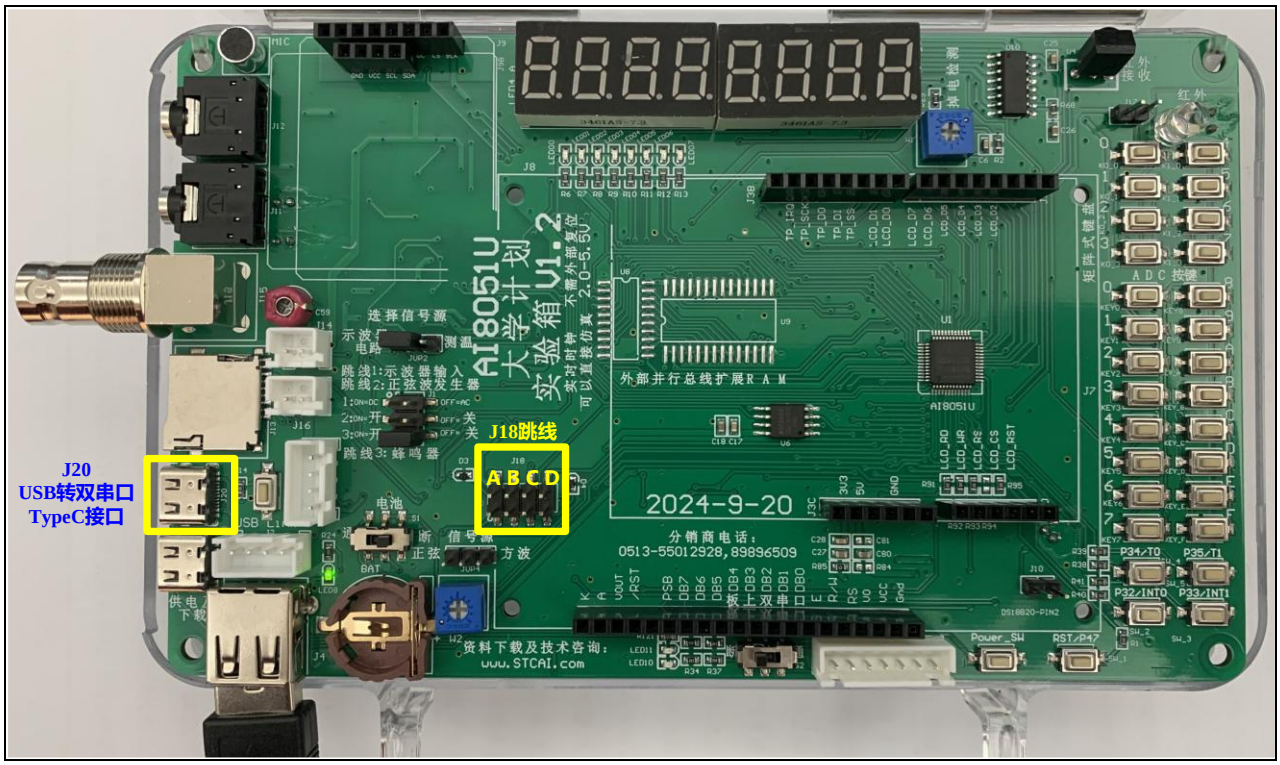
19.10 深大实验-串口通讯(硬件 UART1 与 UART2 数据透传)

19.10.1. 实验介绍

1、 熟悉双串口数据透传的应用
2、 了解 Ai8051U 实验箱原理图与 AIapp-ISP 系统软件的应用
3、 熟悉如何管理多文件项目

a) 认识实验箱：

Ai8051U 的实验箱正面图：



1、J20：　USB 转双串口 U2（Ai8H2K12U）TypeC 接口。本实验需要用 TypeA（连接电脑）-TypeC（连接实验箱 J20）线相连。（详见 J20-U2 原理图）

2、J18 跳线：本实验需要将实验箱上“J18 跳线 A、B、C、D”的跳线连上。

- J18.A：使主控芯片 UART2 的 P4.3/TxD2_2 与双串口 U2（Ai8H2K12U）联通。
- J18.B：使主控芯片 UART2 的 P4.2/RxD2_2 与双串口 U2（Ai8H2K12U）联通。
- J18.C：使主控芯片 UART1 的 P3.7/TxD_2 与双串口 U2（Ai8H2K12U）联通。
- J18.D：使主控芯片 UART1 的 P3.6/RxD_2 与双串口 U2（Ai8H2K12U）联通。

详见 J18 跳线原理图

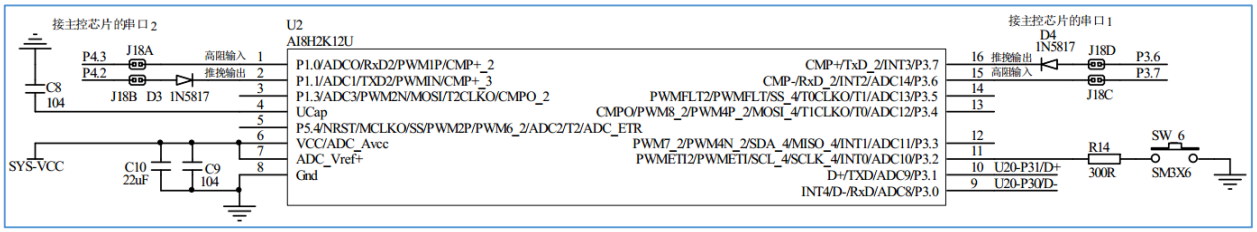
b) Keil 环境下多文件项目管理说明

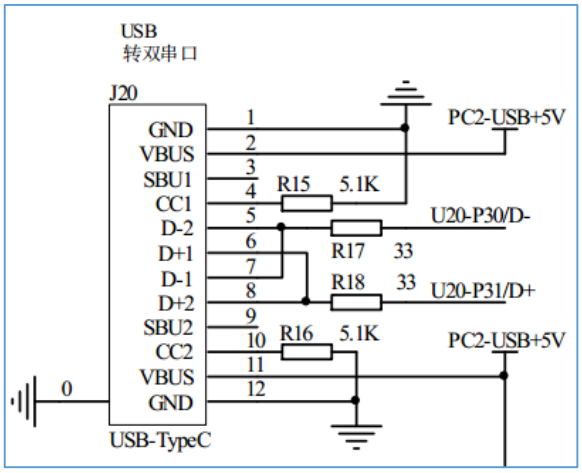
当项目的功能比较复杂时，就需要在 Keil 中建立多文件项目，以方便分工合作、代码复用、模块化管理、增强可读性和可维护性。

- 比较好的建议是将项目功能模块化，不同模块的实现代码放在不同的.c 文件中。
- 一般建议是一个模块对应一个.c 程序文件和一个.h 头文件
- 模块的初始化函数以及相关的数据处理函数都在.c 文件中实现
- 与模块相关的全局变量也必须在.c 文件中进行定义，一定不能在.h 文件中定义变量
- 如果有其他模块需要使用本模块定义的变量或函数，则这些函数和变量都需要在.h 文件中声明。
- 特别提醒：在.h 文件中声明外部变量必须使用 **extern** 关键字，否则就变成变量定义了，这样会出现变量重复定义的错误
- 为防止头文件被多次包含而产生错误或者警告，在头文件中使用类似如下的条件编译组合语句，可避免在同一个.c 文件中对同一个.h 头文件进行多次包含：

```
#ifndef XXXX
#define XXXX
...
#endif
```

19.10.2. 原理图





19.10.3. 实验程序代码

a) main.c -- 程序主函数

```
//main.c
//程序主函数
//短接 J18A，J18B，J18C，J18D 跳线
//USB 线连接电脑与实验箱 J20 接口，开启两个串口助手，分别打开 CDC1，CDC2 对应的 COM 口
//CDC1 串口助手发送的内容会被 CDC2 串口助手接收
//CDC2 串口助手发送的内容会被 CDC1 串口助手接收
```

```
#include "config.h"       //头文件中已包含 ai8051u.h 以及其他头文件

void main()
{
    u8 i;

    SYS_Init();           //系统初始化

    while (1)
    {
        if(B_RX1_TimeOut)           //判断是否完成一串数据的接收
        {
            B_RX1_TimeOut = 0;    //清除接收一串数据完成标志

            for(i=0; i<RX1_Cnt; i++)
            {
                Uart2_Send(RX1_Buffer[i]);       //将串口 1 收到的数据，从串口 2 发送出去
            }

            RX1_Cnt = 0;              //清除接收计数器，下一次接收内容时重新计数
        }

        if(B_RX2_TimeOut)           //判断是否完成一串数据的接收
        {
            B_RX2_TimeOut = 0;    //清除接收一串数据完成标志

            for(i=0; i<RX2_Cnt; i++)
            {
                Uart1_Send(RX2_Buffer[i]);       //将串口 2 收到的数据，从串口 1 发送出去
            }

            RX2_Cnt = 0;              //清除接收计数器，下一次接收内容时重新计数
        }
    }
}

void SYS_Init(void)       //系统初始化函数
{
    EAXFR = 1;          //使能访问扩展 XFR
    WTST = 0x00;        //设置最快速度访问程序代码
    CKCON = 0x00;       //设置最快速度访问内部 XDATA

    P0M0 = 0x00; P0M1 = 0x00;       //初始化 P0 口为准双向口模式
    P1M0 = 0x00; P1M1 = 0x00;       //初始化 P1 口为准双向口模式
    P2M0 = 0x00; P2M1 = 0x00;       //初始化 P2 口为准双向口模式
```

```
    P3M0 = 0x00; P3M1 = 0x00;     //初始化 P3 口为准双向口模式
    P4M0 = 0x00; P4M1 = 0x00;     //初始化 P4 口为准双向口模式
    P5M0 = 0x00; P5M1 = 0x00;     //初始化 P5 口为准双向口模式
    P6M0 = 0x00; P6M1 = 0x00;     //初始化 P6 口为准双向口模式
    P7M0 = 0x00; P7M1 = 0x00;     //初始化 P7 口为准双向口模式

    S1_S1 = 0;          //UART1 通道选择, 00: P3.0 P3.1, 01: P3.6 P3.7, 10: P1.6 P1.7, 11: P4.3 P4.4
    S1_S0 = 1;          //UART1 选择  P3.6 P3.7  作为串口收发通道

    S2_S = 1;           //UART2 通道选择: 0: P1.2 P1.3,    1: P4.2 P4.3

    Uart1_Init();       //UART1 接口初始化
    Uart2_Init();       //UART2 接口初始化

    EA = 1;             //使能全局中断
}
```

b) uart.c -- 初始化 UART 接口

```
//uart.c
//初始化 UART 接口
//中断处理 UART 收发数据结果

#include "config.h"       //头文件中已包含 ai8051u.h 以及其他头文件

u8   RX1_Cnt;             //接收计数
bit  B_TX1_Busy;          //发送忙标志
bit  B_RX1_TimeOut;       //接收超时/空闲标志

u8   RX2_Cnt;             //接收计数
bit  B_TX2_Busy;          //发送忙标志
bit  B_RX2_TimeOut;       //接收超时/空闲标志

u8   RX1_Buffer[256];     //接收缓冲
u8   RX2_Buffer[256];     //接收缓冲

void Uart1_Isr(void) interrupt 4
{
    if (TI)               //检测串口 1 发送中断
    {
        TI = 0;           //清除串口 1 发送中断请求位
        B_TX1_Busy = 0;   //清除发送忙标志
    }

    if (RI)               //检测串口 1 接收中断
```

```
    {
        RI = 0;                 //清除串口 1 接收中断请求位
        RX1_Buffer[RX1_Cnt++] = SBUF;    //将收到数据存入缓冲区
    }

    if(UR1TOSR & 0x01)          //检测是否产生超时中断（产生超时中断说明一串数据已经接收完成）
    {
        B_RX1_TimeOut = 1;      //设置超时中断用户标志
        UR1TOSR = 0x80;         //设置 RTOCF 清除超时标志位 TOIF
    }
}

void Uart1_Init(void)           //115200bps@40.000MHz
{
    SCON = 0x50;                //8 位数据,可变波特率
    AUXR |= 0x40;               //定时器时钟 1T 模式
    AUXR &= 0xFE;               //串口 1 选择定时器 1 为波特率发生器
    TMOD &= 0x0F;               //设置定时器模式
    TL1 = 0xA9;                 //设置定时初始值
    TH1 = 0xFF;                 //设置定时初始值
    ET1 = 0;                    //禁止定时器中断
    TR1 = 1;                    //定时器 1 开始计时
    ES = 1;                     //使能串口 1 中断

    UR1TOCR = 0xc0;
        //使能超时接收，使能超时中断，超时时钟选择 1:系统时钟  0:串口数据位率(波特率)
    UR1TOTL = 0x10;             //设置超时时间：16 个串口数据位时间
    UR1TOTH = 0x00;
    UR1TOTE = 0x00;             //需要写 UR1TOTE 后，新的 TimeOut 值才会生效
}

void Uart1_Send(u8 dat)         //通过串口 1 发送数据
{
    SBUF = dat;                 //写入需要发送的数据
    B_TX1_Busy = 1;             //设置发送忙标志
    while(B_TX1_Busy);          //判断是否发送完成
}

void Uart2_Isr(void) interrupt 8
{
    if (S2CON & 0x02)           //检测串口 2 发送中断
    {
        S2CON &= ~0x02;         //清除串口 2 发送中断请求位
        B_TX2_Busy = 0;         //清除发送忙标志
    }
```

```
    if (S2CON & 0x01)           //检测串口 2 接收中断
    {
        S2CON &= ~0x01;         //清除串口 2 接收中断请求位
        RX2_Buffer[RX2_Cnt++] = S2BUF;   //将收到数据存入缓冲区
    }

    if(UR2TOSR & 0x01)          //检测是否产生超时中断（产生超时中断说明一串数据已经接收完成）
    {
        B_RX2_TimeOut = 1;      //设置超时中断用户标志
        UR2TOSR = 0x80;         //设置 RTOCF 清除超时标志位 TOIF
    }
}

void Uart2_Init(void)           //115200bps@40.000MHz
{
    S2CON = 0x50;               //8 位数据,可变波特率
    AUXR |= 0x04;               //定时器时钟 1T 模式
    T2L = 0xA9;                 //设置定时初始值
    T2H = 0xFF;                 //设置定时初始值
    AUXR |= 0x10;               //定时器 2 开始计时
    IE2 |= 0x01;                //使能串口 2 中断

    UR2TOCR = 0xc0;
        //使能超时接收，使能超时中断，超时时钟选择 1:系统时钟  0:串口数据位率(波特率)
    UR2TOTL = 0x10;             //设置超时时间：16 个串口数据位时间
    UR2TOTH = 0x00;
    UR2TOTE = 0x00;             //需要写 UR2TOTE 后，新的 TimeOut 值才会生效
}

void Uart2_Send(u8 dat)         //通过串口 2 发送数据
{
    S2BUF = dat;                //写入需要发送的数据
    B_TX2_Busy = 1;             //设置发送忙标志
    while(B_TX2_Busy);          //判断是否发送完成
}
```

c) config.h -- 项目配置的头文件

```
#ifndef __CONFIG_H__            //防止头文件被重复包含
#define __CONFIG_H__

#define HIRC            40000000UL
#define FOSC            40000000UL
#define SYSCLK          FOSC
#define MAIN_Fosc       FOSC
```

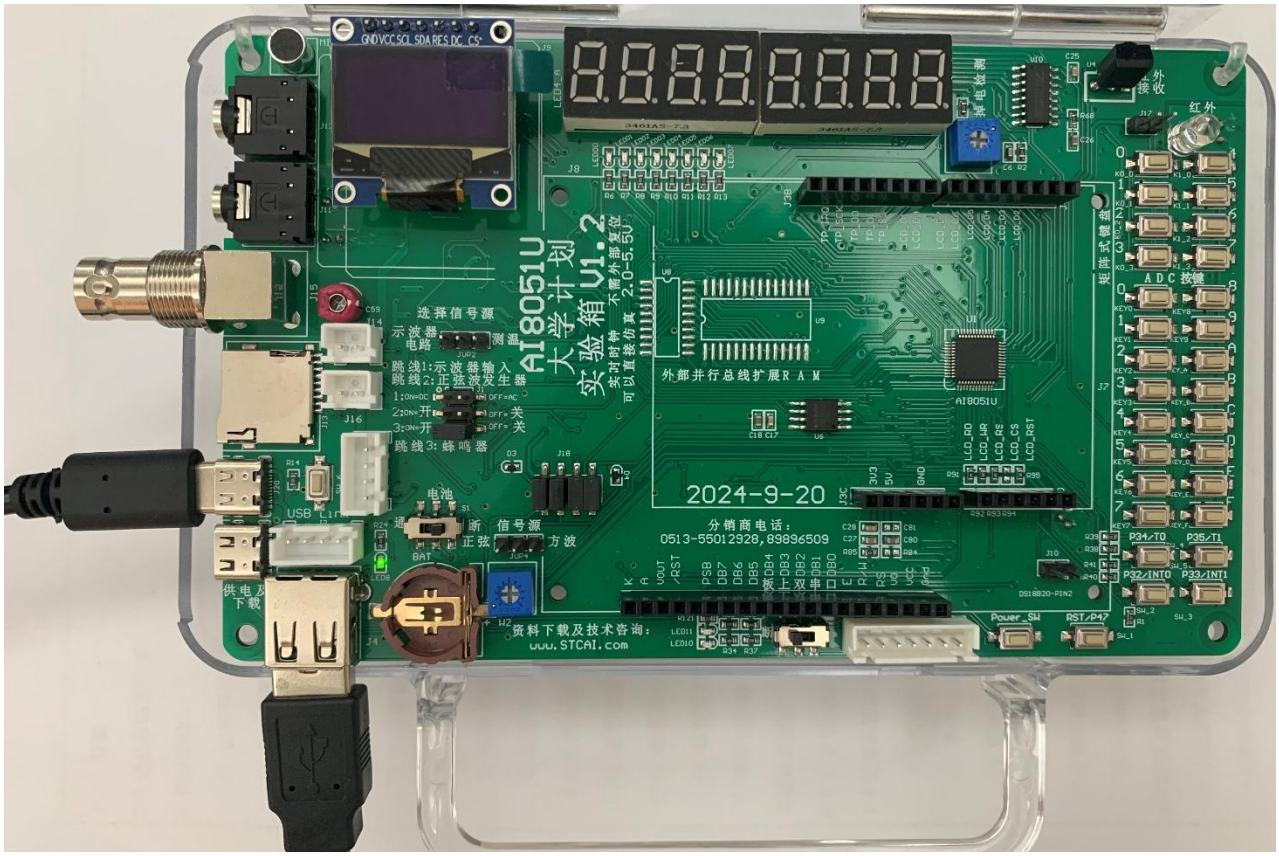
```
#include <ai8051u.h>              //包含外部头文件
#include <stdio.h>
#include <intrins.h>

#include "def.h"                  //包含项目头文件
#include "uart.h"

void SYS_Init(void);              //函数声明

#endif
```

d) uart.h -- 项目配置文件的头文件

```
#ifndef __UART_H__          //防止头文件被重复包含
#define __UART_H__

extern    u8    RX1_Cnt;              //接收计数
extern    bit   B_RX1_TimeOut;        //接收超时/空闲标志

extern    u8    RX2_Cnt;              //接收计数
extern    bit   B_RX2_TimeOut;        //接收超时/空闲标志

extern    u8    RX1_Buffer[256];      //接收缓冲
extern    u8    RX2_Buffer[256];      //接收缓冲


void      Uart1_Init(void);           //函数声明
void      Uart2_Init(void);           //函数声明
void      Uart1_Send(u8 dat);         //函数声明
void      Uart2_Send(u8 dat);         //函数声明

#endif
```

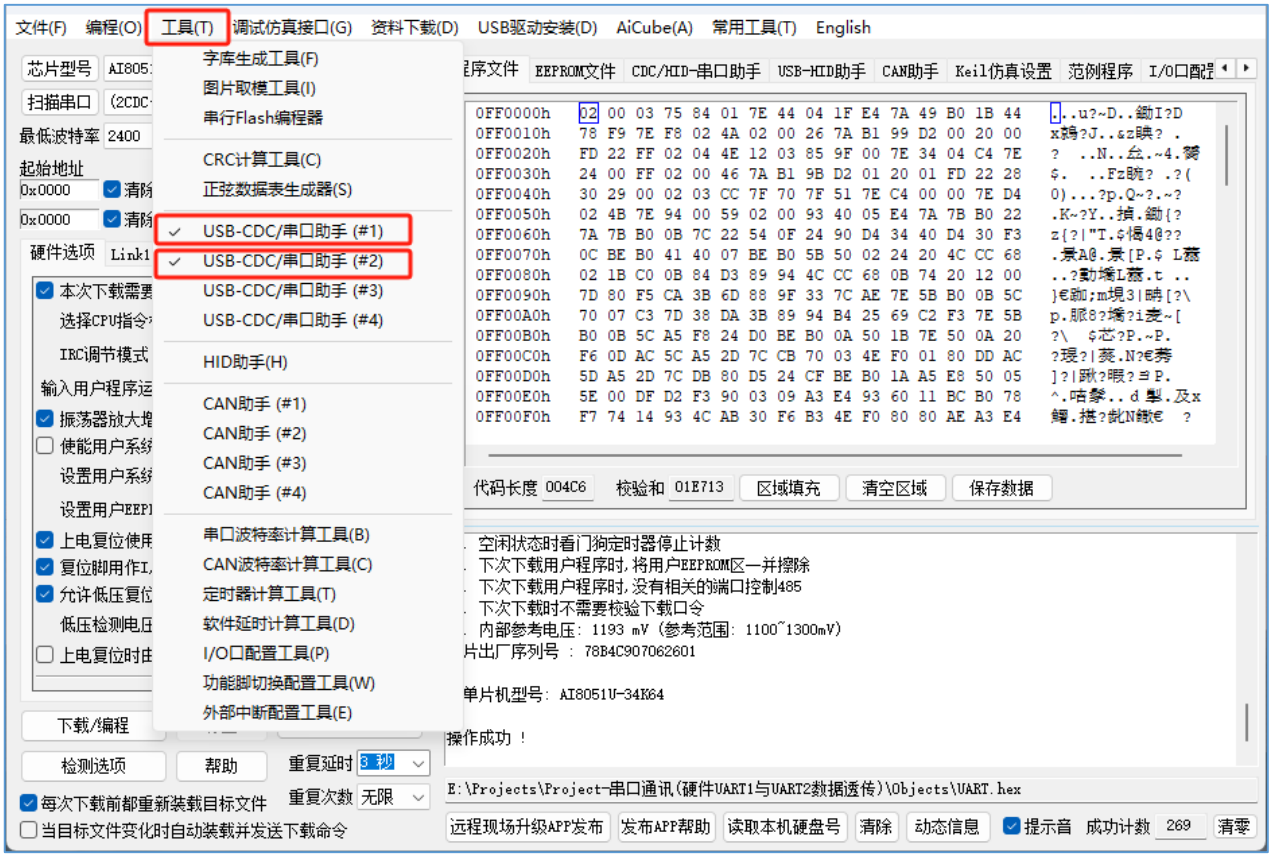
19.10.4. 程序下载

录入代码，保存，编译。按实验要求连接并设置好 Ai8051U 实验箱，如下图：

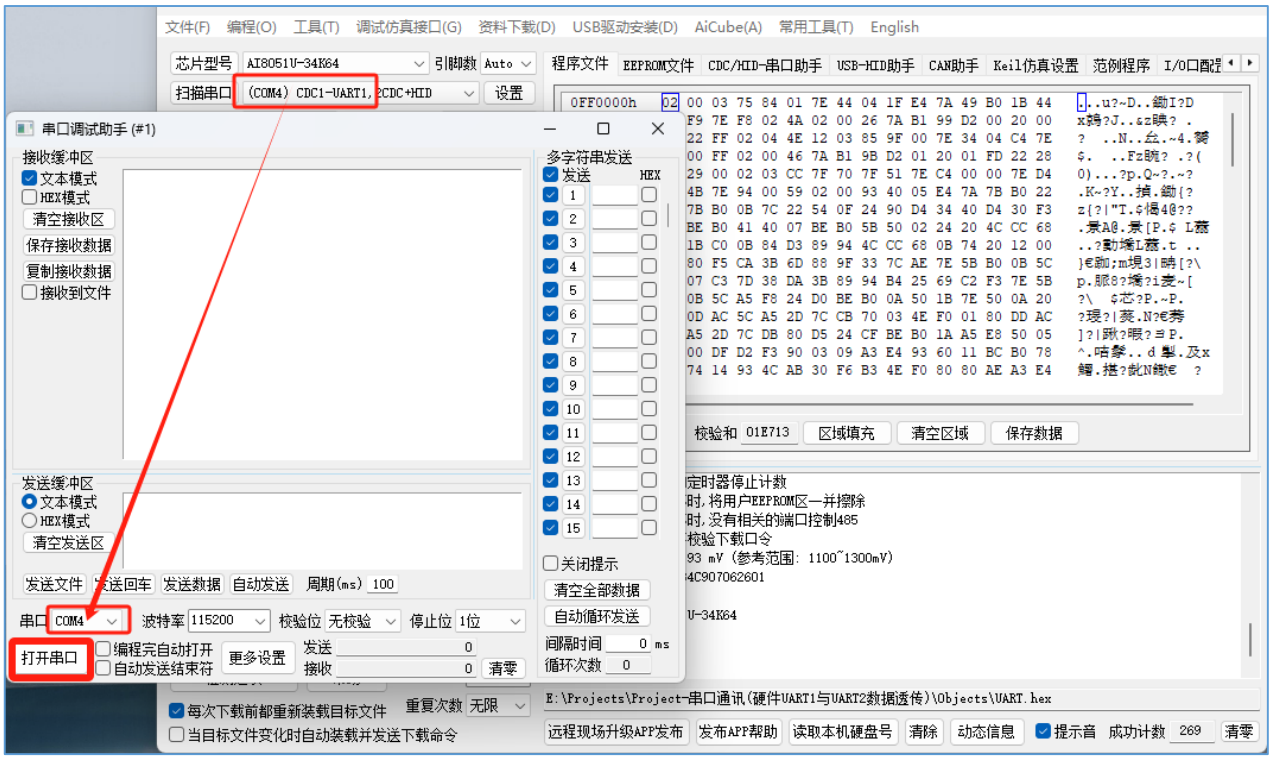


运行 AIapp-ISP 软件系统，按步骤打开并下载程序“uart.hex”，如下图：

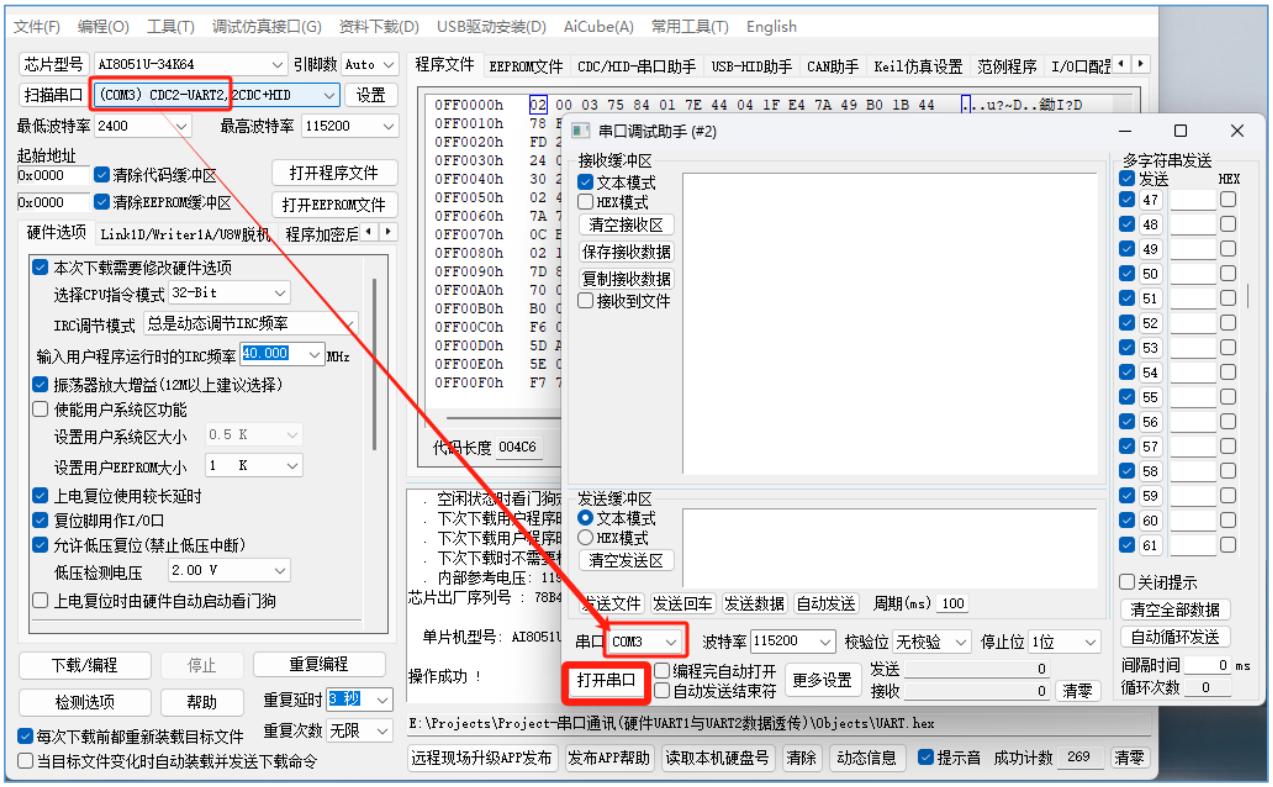
在 AIapp-ISP 软件系统的菜单栏，点击“工具”→勾选“USB-CDC/串口助手(#1)”和“USB-CDC/串口助手(#2)”，如下图：



就会打开“串口调试助手(#1)”和“串口调试助手(#2)”两个浮窗，
在“串口调试助手(#1)”窗口：选择与 CDC1 对应的 COM4 串口，并找到“打开串口”按钮，如下图：



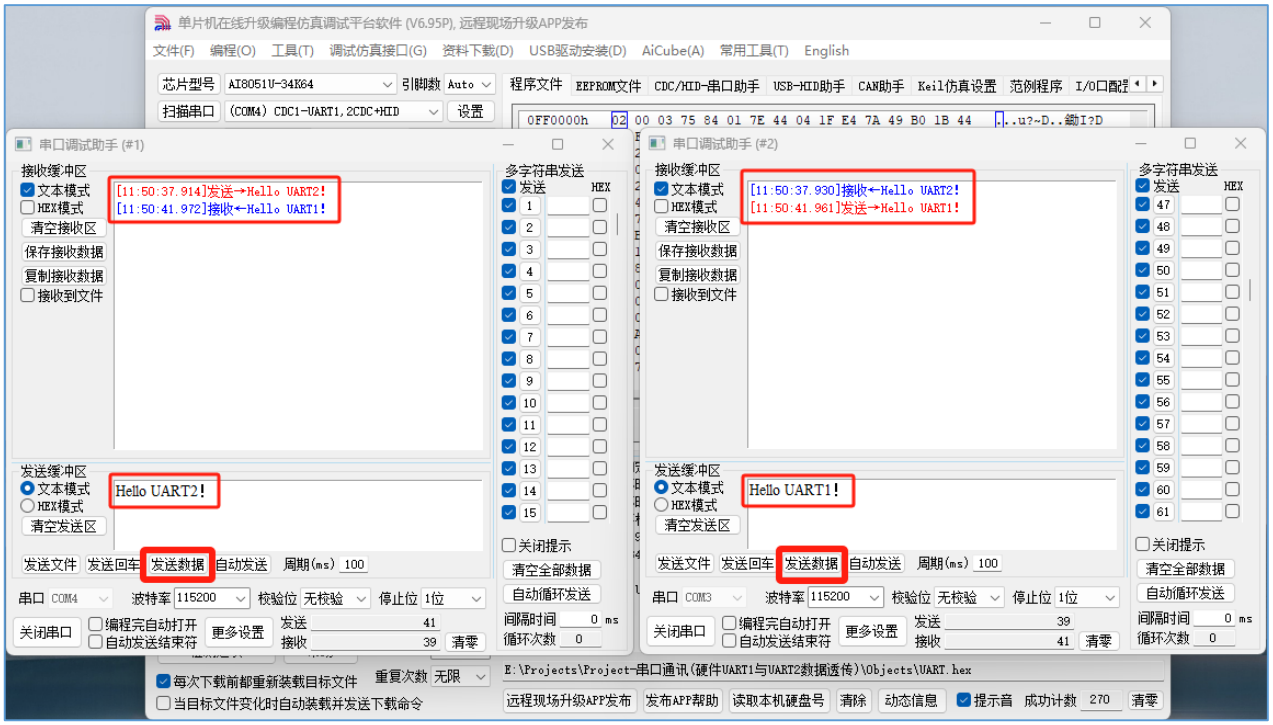
在“串口调试助手(#2)”窗口：选择与 CDC2 对应的 COM3 串口，并找到“打开串口”按钮，如下图：



一切准备就绪！

19.10.5. 验证实验效果

在“串口调试助手(#1)”窗口的发送缓冲区输入“Hello UART2！”
在“串口调试助手(#2)”窗口的发送缓冲区输入“Hello UART1！”
分别点击两个 ISP 界面的“打开串口”按钮，然后分别点击“发送数据”，如下图：



完成实验验证。