

## 19.14 深大实验-串口通讯(USB-CDC 发命令控制硬件 UART1 与 UART2 多机通讯)

### 19.14.1. 实验介绍

1、熟悉串口 1/UART1 和串口 2/UART2 的通讯原理

2、了解多机通讯原理

多机通讯的过程如下:

① 置全部从机的 SM2=1, 处于只接收地址帧状态。

② 主机首先发送呼叫地址帧信息, 将第 9 位 TB8 设置为 1, 以表示发送的是呼叫地址帧。

③ 所有从机接收到呼叫地址帧后, 各自将接收到的主机呼叫的地址与本机的地址相比较:

➢ 若比较结果相等, 则为被寻址从机, 清除 SM2=0, 准备接收从主机发送的数据帧, 直至全部数据传输完;

➢ 若比较不相等, 则为非寻址从机, 仍维持 SM2=1 不变, 对其后发来的数据帧不予理睬, 即接收到的数据帧内容不装入 SBUF, 不置位, RI=0, 不会产生中断请求, 直至被寻址为止。

④ 主机在发送完呼叫地址帧后, 接着发送一连串的数据帧, 其中的第 9 位 TB8=0, 以表示为数据帧。只有那些 SM2 位为 0 的从机 (即已接收地址帧的从机) 才会接收数据帧。

⑤ 当主机改变从机通信时间则再发呼叫地址帧, 寻呼其他从机, 原先被寻址的从机经分析得知主机在寻呼其他从机时, 恢复其 SM2=1, 对其后主机发送的数据帧不予理睬。

上述过程均在软件控制下实现。

3、了解 Ai8051U 实验箱原理图

4、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:



S2 开关: S2 开关默认是“断”, 本实验要将 S2 开关拨到“通”, 使主控芯片 UART1 的 P3.7/TxD\_2 与 UART2 的 P4.2/RxD2\_2 联通。UART1 (主机) 与 UART2 (从机) 通讯时, LED11 会闪烁。(详见 S2 开关原理图)

#### b) Keil 环境下多文件项目管理说明

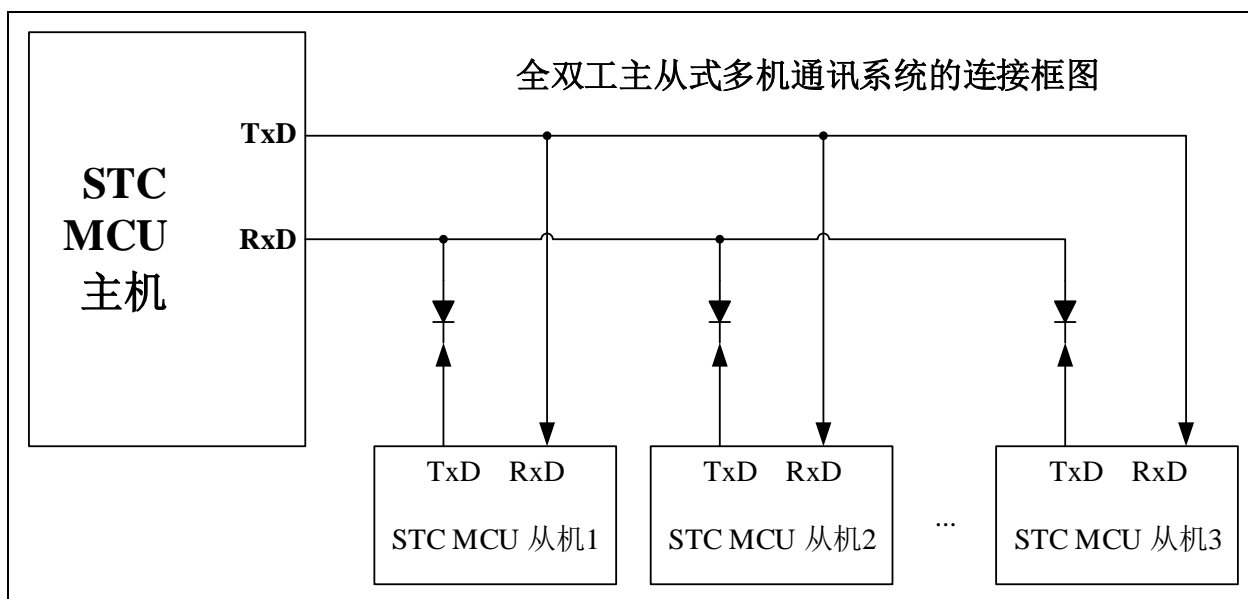
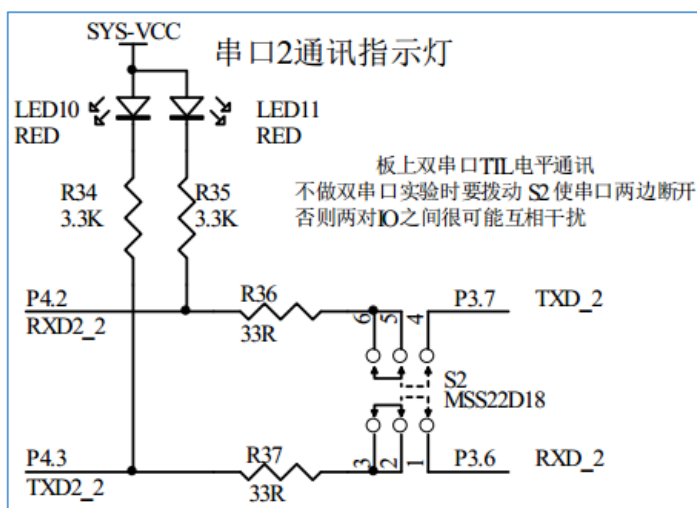
当项目的功能比较复杂时, 就需要在 Keil 中建立多文件项目, 以方便分工合作、代码复用、模块化管理、增强可读性和可维护性。

➢ 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
```

## 19.14.2. 原理图



## 19.14.3. 实验程序代码

### a) main.c -- 程序主函数

```
//main.c
//程序主函数
//将 S2 拨动开关拨到“通”
//USB 线连接电脑与实验箱 J3 或者 J4 接口，烧录程序后，串口助手打开 CDC 对应的 COM 口
//通过串口助手给 MCU 发送从机地址，串口 2 地址是"0x32"，发送字符"2"或者 hex 格式"0x32"
//地址匹配情况下，串口助手就能输出串口 1 每秒钟发送给串口 2 的数据

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

u8 addr;

void main()
{
    SYS_Init();        //系统初始化

    while (1)
    {
        if(fls)        //判断是否完成一串数据的接收
        {
            fls = 0;    //清除接收一串数据完成标志
            Uart1_SendAddr(addr);    //发送从机地址，地址匹配的从机才能收到数据
            printf("AI8051U UART1-UART2 Test Programme!\r\n");    //UART1 发送一个字符串
        }

        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

            USB_SendData(RX2_Buffer,RX2_Cnt);    //将串口 2 收到的数据，从 USB-CDC 接口发送出去
            RX2_Cnt = 0;    //清除接收计数器，下一次接收内容时重新计数
        }

        if (bUsbOutReady)    //查询是否有接收到数据
        {
            addr = UsbOutBuffer[0];

            usb_OUT_done();    //当前包的数据处理完成
        }
    }
}
```

```

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

    usb_init();             //USB 库函数初始化
    Uart1_Init();           //UART1 接口初始化
    Uart2_Init();           //UART2 接口初始化
    Timer0_Init();          //定时器 0 初始化,用于产生 1s 的时间基准

    EA = 1;                 //总中断允许位打开
}

```

## 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
        SM2 = 0;           //本机被选中后,进入数据接收状态
        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)
    {
        SM2 = 1;          //若发送完成,则重新开始地址检测

        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 = 0xf8;          //9 位数据,可变波特率,开启地址筛选
    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

    SADDR = 0x31;          //设置 UART1 设备地址
    SADEN = 0xff;          //设置需要校验的地址位, 置 1 的位需要校验
```

```
}

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
}

char putchar(char c)      //将 printf 函数映射到 UART1 接口
{
    Uart1_Send(c);        //调用串口 1 发送函数（如果要映射到串口 2 的话，这里修改为串口 2 发送函数）
    return c;
}

void Uart1_SendAddr(u8 dat) //通过串口 1 发送地址
{
    TB8 = 1;              //本次发送的是地址帧

    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

    TB8 = 0;              //后续发送的不是地址帧
}

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
        S2SM2 = 0;         //本机被选中后,进入数据接收状态
        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）
    {
        S2SM2 = 1;         //若发送完成,则重新开始地址检测
    }
}
```

```

        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 = 0xf8;          //9 位数据,可变波特率,开启地址筛选
    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

    S2ADDR = 0x32;         //设置 UART2 设备地址
    S2ADEN = 0xff;         //设置需要校验的地址位, 置 1 的位需要校验
}

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
}

void Uart2_SendAddr(u8 dat) //通过串口 2 发送地址
{
    S2TB8 = 1;             //本次发送的是地址帧

    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

    S2TB8 = 0;             //后续发送的不是地址帧
}

```

### c) timer.c -- 定时器初始化以及定时器中断服务程序

```
//timer.c
```

//定时器初始化以及定时器中断服务程序

//定时器 0:1s 定时, 产生 1s 的标志位

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

bit  fls;                     //1s 标志位

void Timer0_Init(void)        //1 秒@40.000MHz
{
    TM0PS = 0x32;
    //设置定时器时钟预分频 ( 注意:并非所有系列都有此寄存器,详情请查看数据手册 )
    AUXR &= 0x7F;             //定时器时钟 12T 模式
    TMOD &= 0xF0;             //设置定时器 0 为模式 0
    TL0 = 0xB1;               //设置定时器 0 初始值
    TH0 = 0x00;               //设置定时器 0 初始值
    TF0 = 0;                  //清除 TF0 标志
    TR0 = 1;                  //定时器 0 开始计时
    ET0 = 1;                  //使能定时器 0 中断

    fls = 0;                  //初始化 1s 标志位
}

void Timer0_Isr(void) interrupt 1 //定时器 0 中断服务程序
{
    fls = 1;                  //设置 1s 标志位
}
```

#### d) 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"
#include "timer.h"
#include "ai_usb.h"
```



```
void SYS_Init(void);           //函数声明
```

```
#endif
```

### e) 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
void Uart1_SendAddr(u8 dat);   //函数声明
void Uart2_SendAddr(u8 dat);   //函数声明

#endif
```

### f) timer.h -- 项目配置文件的头文件

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

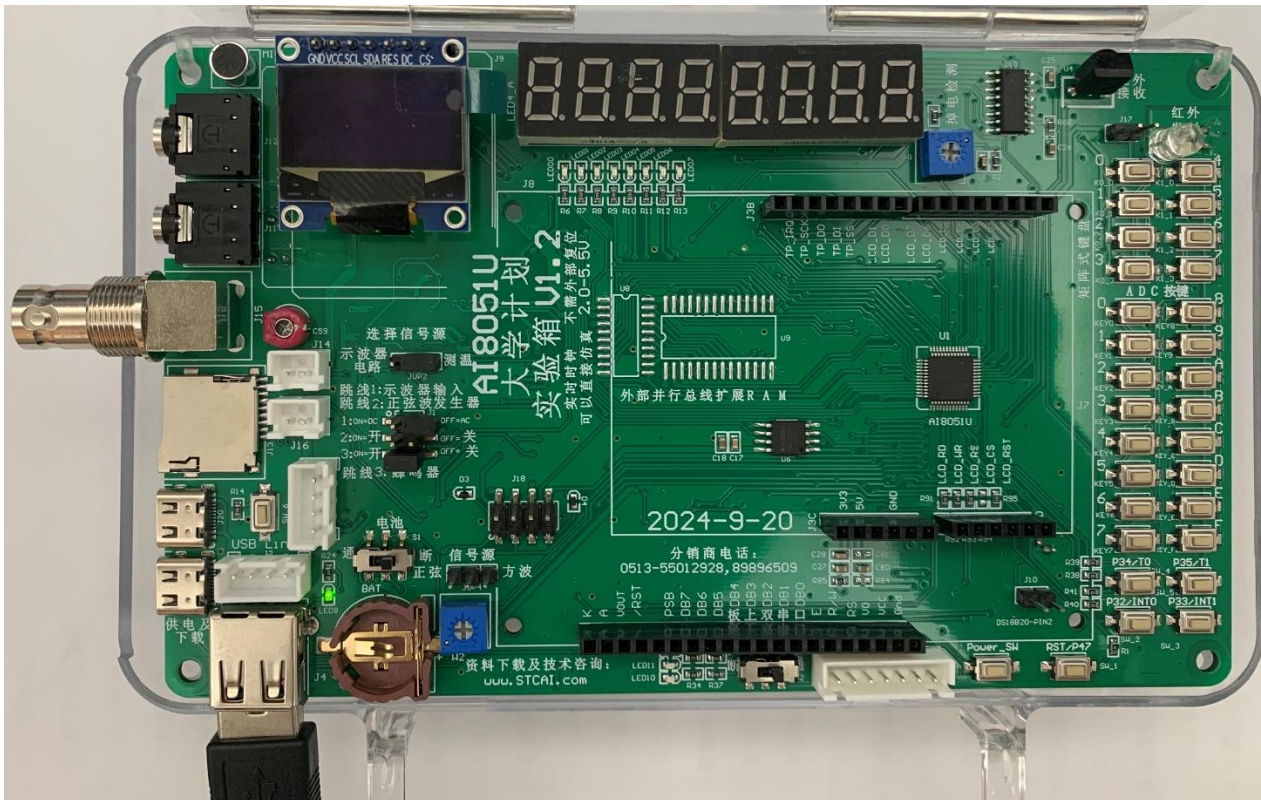
void Timer0_Init(void);       //函数声明

extern bit fls;                //外部变量声明

#endif
```

## 19.14.4. 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：



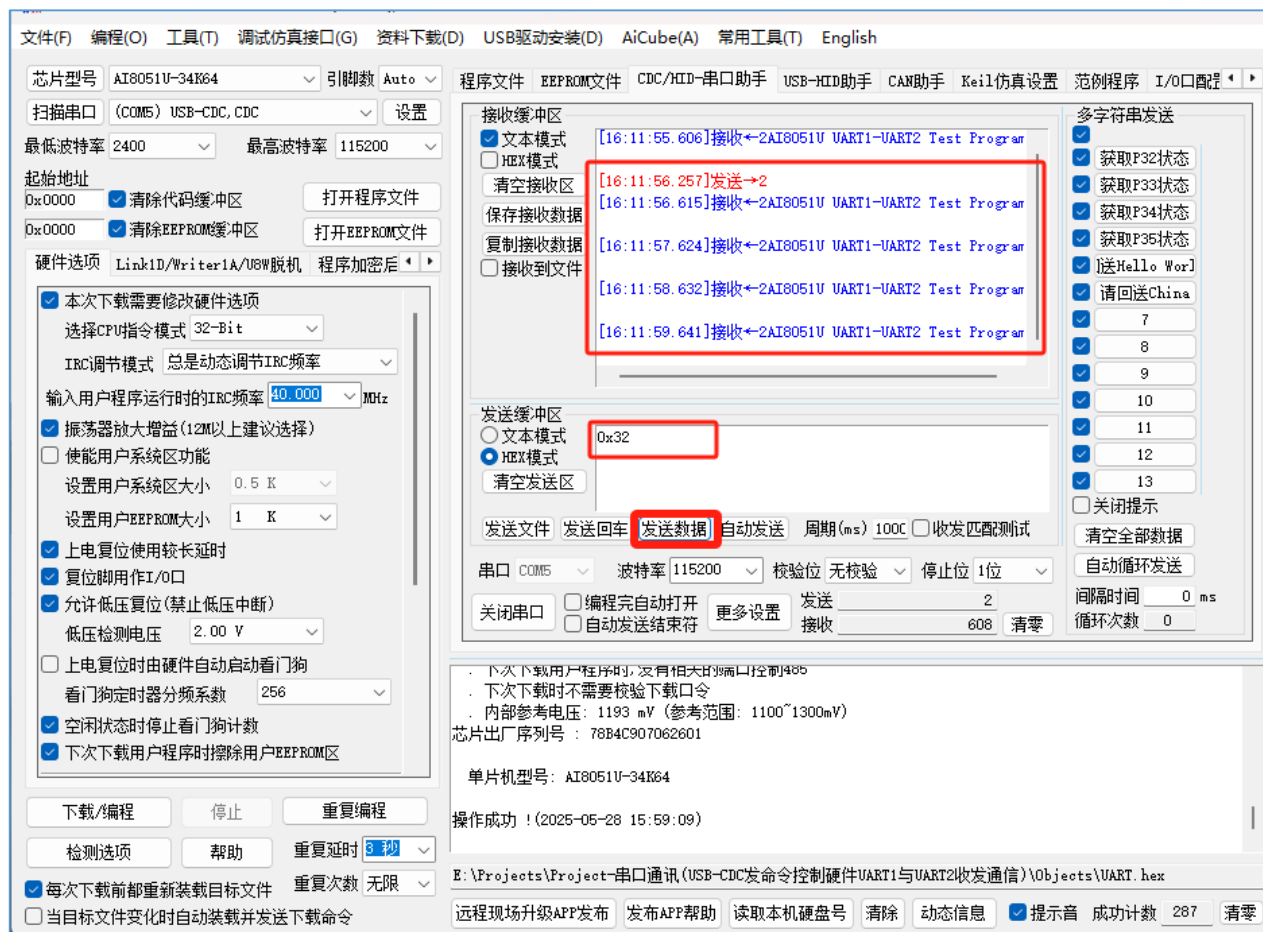


## 19.14.5. 查看实验现象, 验证实验效果

- 点击“打开串口”按钮;
- 因在程序代码中设置 UART2 设备(从机)地址为 0x32, 则在发送缓冲区窗口输入 0x32, 选择“HEX 模式”;
- 点击“发送数据”按钮

我们看到:

- 每秒钟, USB-CDC 串口助手就能输出串口 1(主机)每秒钟发送给串口 2(从机)的数据“AI8051U UART1-UART2 Test Programme!”, 显示在接收缓冲区, 如下图:



- 观察 Ai8051U 实验箱, 我们会看到 S2 开关旁边的 LED11 在闪烁。

完成实验验证。