嵌入式系统及应用实验报告

<table>
<tr><td>学号</td><td>202018526</td><td>姓名</td><td>高树林</td><td>专业年级</td><td>人工智能 2020 级</td></tr>
<tr><td>实验地点</td><td colspan="2">南楼中栋 108 室</td><td>实验时间</td><td colspan="2">2023-05-11</td></tr>
<tr><td>实验名称</td><td colspan="5">实验二 串口通信及中断实验</td></tr>
<tr><td>实验学时</td><td>2</td><td>实验类型</td><td colspan="3">■验证型 □设计型 ■创新型 ■综合型</td></tr>
<tr><td>实验目的</td><td colspan="5">1. 以串行接收中断为例，掌握中断的基本编程步骤。
2. 通过接收单字节/多个字节指令，掌握上位机控制下位机的串口通信编程方法。
3. UART 通信编程方法。</td></tr>
<tr><td>实验内容</td><td colspan="5">1.验证串行通信编程方法和串口接收中断的处理过程；
2.设计上位机和下位机实时通信程序，实现单字节接收中断和多字节接收中断。</td></tr>
<tr><td>实验任务及实验过程</td><td colspan="5">1.验证
从华水学堂资料中，下载第 6 章源代码，将工程\04-Software\CH06\中的工程导入编译器。理解 UART 构件式的嵌入式编程过程，以及中断处理的过程。请写出程序功能和运行流程：</td></tr>
</table>

```
1. //=============================================================
   ==============
2. //文件名称：main.c（应用工程主函数）
3. //框架提供：SD-Arm（sumcu.suda.edu.cn）
4. //版本更新：20191108-20201106
5. //功能描述：见本工程的..\01_Doc\Readme.txt
6. //移植规则：【固定】
7. //=============================================================
   ==============
8. #define GLOBLE_VAR
9. #include "includes.h"     //包含总头文件
10.
11.
12. //-------------------------------------------------------------------
13. //声明使用到的内部函数
14. //main.c 使用的内部函数声明处
15.
16. //-------------------------------------------------------------------
17. //主函数，一般情况下可以认为程序从此开始运行（实际上有启动过程，
    参见书稿）
18. int main(void)
19. {
20. //（1）=======启动部分（开头）
    ========================================
21. //（1.1）声明 main 函数使用的局部变量
22.     uint32_t mMainLoopCount; //主循环次数变量
23.     //uint8_t  mFlag;         //灯的状态标志
24.     uint8_t  mTest;
```

```
25.     uint32_t mLightCount;    //灯的状态切换次数
26.     uint32_t number=202018601;
27.     uint8_t * name="han";
28.
29.  //（1.2）【不变】关总中断
30.     DISABLE_INTERRUPTS;
31.
32.  //（1.3）给主函数使用的局部变量赋初值
33.     mMainLoopCount=0;   //主循环次数变量
34.     //mFlag='A';          //灯的状态标志
35.     mLightCount=0;      //灯的闪烁次数
36.     //mTest='T';
37.  //（1.4）给全局变量赋初值
38.
39.  //（1.5）用户外设模块初始化
40.     gpio_init(LIGHT_BLUE,GPIO_OUTPUT,LIGHT_ON); //初始化蓝灯
41.     uart_init(UART_User,115200);    //初始化串口模块
42.
43.  //（1.6）使能模块中断
44.   uart_enable_re_int(UART_User); //使能 UART_USER 模块接收中断功能
45.  //（1.7）【不变】开总中断
46.     ENABLE_INTERRUPTS;
47.     printf("%d %s\n",number,name);
48.
49.  //（1）=======启动部分（结尾）
     ==========================================
50.
51.  //（2）=======主循环部分（开头）
     ========================================
52.     for(;;)  //for(;;)（开头）
53.     {
54.  //（2.1）主循环次数变量+1
55.        mMainLoopCount++;
56.  //（2.2）未达到主循环次数设定值，继续循环
57.        if (mMainLoopCount<=3000000)  continue;
58.  //（2.3）达到主循环次数设定值，执行下列语句，进行灯的亮暗处理
59.  //（2.3.1）清除循环次数变量
60.        mMainLoopCount=0;
61.  //（2.3.2）如灯状态标志 mFlag 为'L'，灯的闪烁次数+1 并显示，改变灯状
     态及标志
62.        if (mFlag=='1')              //判断灯的状态标志
63.        {
64.           mLightCount++;
65.
```

```
            //mFlag='A';                    //灯的状态标志
            gpio_set(LIGHT_BLUE,LIGHT_ON);  //灯“亮”
            printf(" 高树林\n");  //串口输出灯的状态
        }
//（2.3.3）如灯状态标志 mFlag 为'A'，改变灯状态及标志
        else if (mFlag=='0')
        {
            //mFlag='L';                    //灯的状态标志
            gpio_set(LIGHT_BLUE,LIGHT_OFF); //灯“暗”
            printf(" 202018526\n");  //串口输出灯的状态
        }
        else
            printf("invalid command!\n");  //串口输出灯的状态
    }  //for(;;)结尾


//（2）=======主循环部分（结尾）
    ========================================

}  //main 函数（结尾）


//=======以下为主函数调用的子函数
    ============================================

//*void USART_Test(){

//}*/

//=============================================================
    ==================
/*
 知识要素：
 （1）main.c 是一个模板，该文件所有代码均不涉及具体的硬件和环境，通过调用构件
    实现对硬件的干预。
 （2）本文件中对宏 GLOBLE_VAR 进行了定义，所以在包含"includes.h"头文件时，会定
    义全局变量，在其他文件中包含"includes.h"头文件时，
    编译时会自动增加 extern
*/
```

这段代码是一个嵌入式系统的主函数，它包含了启动部分和主循环部分。以下是程序的功能和运行流程：

启动部分（开头）

(1.1) 声明 main 函数使用的局部变量

(1.2) 关总中断

(1.3) 给主函数使用的局部变量赋初值

(1.4) 给全局变量赋初值

(1.5) 用户外设模块初始化

(1.6) 使能模块中断

(1.7) 开总中断

主循环部分（开头）

(2.1) 主循环次数变量+1

(2.2) 未达到主循环次数设定值，继续循环

(2.3) 达到主循环次数设定值，执行下列语句，进行灯的亮暗处理

(2.3.1) 清除循环次数变量

(2.3.2) 如灯状态标志 mFlag 为'L'，灯的闪烁次数+1 并显示，改变灯状态及标志

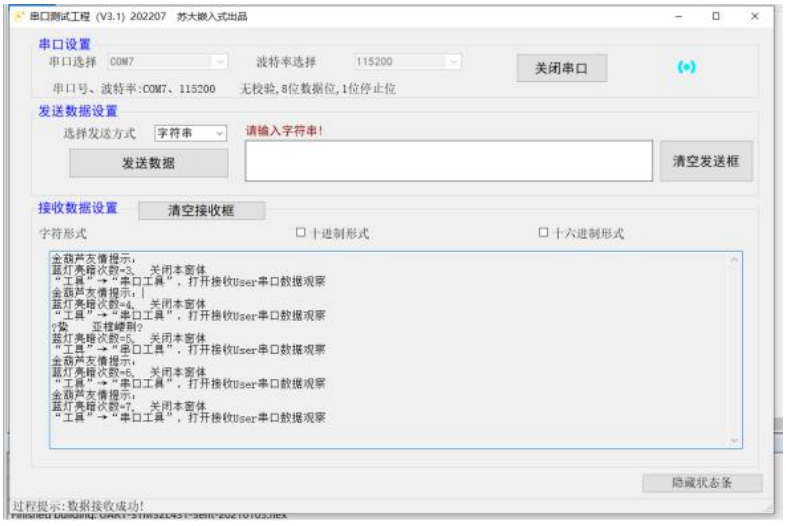
(2.3.3) 如灯状态标志 mFlag 为'A'，改变灯状态及标志如果都不是，则输出 "invalid command!"。

主循环部分（结尾）

UART 构件用于实现串行通信，因此在该程序中使用了 UART 构件。在启动部分，UART_User 被初始化为波特率为 115200 的 UART 模块，并且接收中断功能被使能。在主循环部分中，当接收到 UART 数据时，中断处理程序将被调用，从而执行特定的操作。

中断处理程序的具体过程未在该代码中给出。一般来说，中断处理程序会检查中断源，并根据中断源的类型执行相应的处理操作。在 UART 接收中断的情况下，中断处理程序可能会读取 UART 缓冲区中的数据，并执行特定的操作。

运行结果截图：（请贴上你的运行截图）

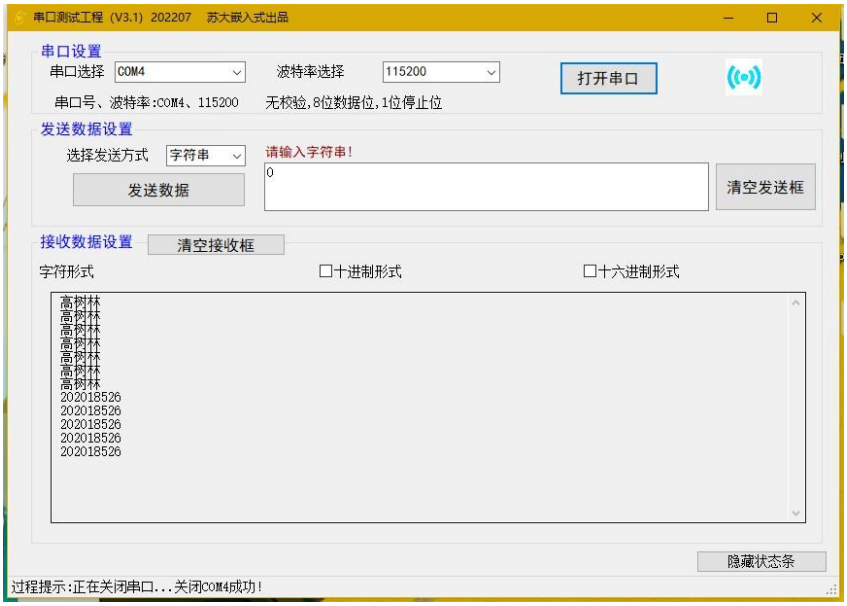


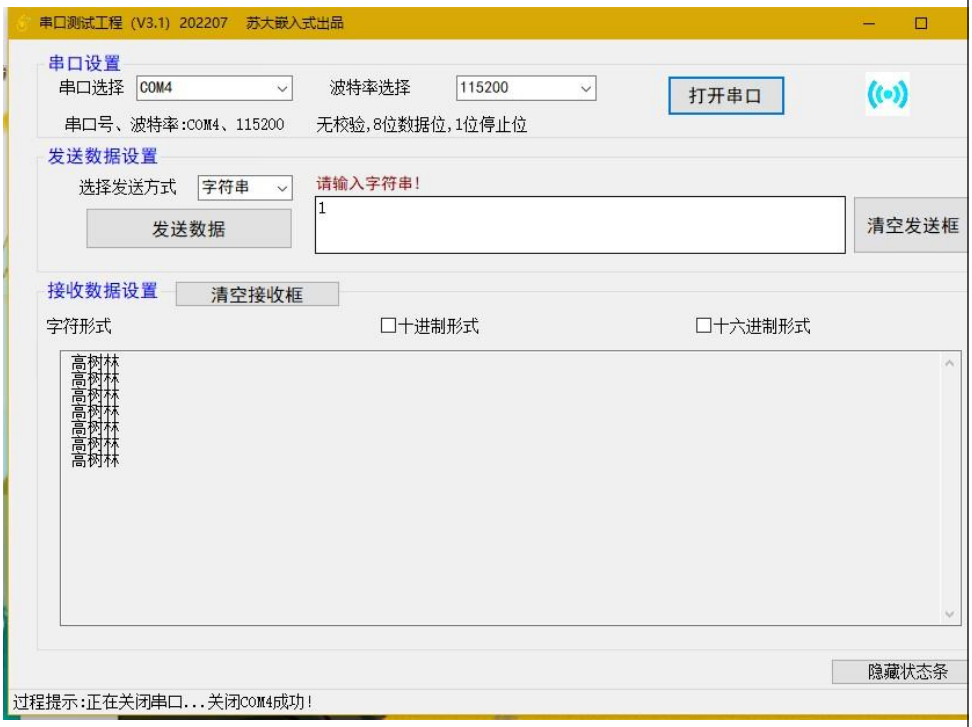
2.设计

请参考已有的工程，设计上位机（PC 机）和下位机（MCU）之间的通信程序。

（1）PC 机发送字符‘1’时，单片机收到后，回送自己的真实姓名，并点亮

	一盏灯，PC 机发送字符‘0’时，单片机收到后，回送自己的真实学号，并将该盏灯熄灭。 **核心代码解析和运行结果截图：** 这段代码是一个嵌入式系统的主函数，它包含了启动部分和主循环部分。以下是程序的功能和运行流程： 启动部分（开头） (1.1) 声明 main 函数使用的局部变量 (1.2) 关总中断 (1.3) 给主函数使用的局部变量赋初值 (1.4) 给全局变量赋初值 (1.5) 用户外设模块初始化 (1.6) 使能模块中断 (1.7) 开总中断 主循环部分（开头） (2.1) 主循环次数变量+1 (2.2) 未达到主循环次数设定值，继续循环 (2.3) 达到主循环次数设定值，执行下列语句，进行灯的亮暗处理 (2.3.1) 清除循环次数变量 (2.3.2) 如灯状态标志 mFlag 为'L'，灯的闪烁次数+1 并显示，改变灯状态及标志，并输出学号 (2.3.3) 如灯状态标志 mFlag 为'A'，改变灯状态及标志如果都不是，则输出 "姓名!"。 主循环部分（结尾） UART 构件用于实现串行通信，因此在该程序中使用了 UART 构件。在启动部分，UART_User 被初始化为波特率为 115200 的 UART 模块，并且接收中断功能被使能。在主循环部分中，当接收到 UART 数据时，中断处理程序将被调用，从而执行特定的操作。 中断处理程序的具体过程未在该代码中给出。一般来说，中断处理程序会检查中断源，并根据中断源的类型执行相应的处理操作。在 UART 接收中断的情况下，中断处理程序可能会读取 UART 缓冲区中的数据，并执行特定的操作。 运行结果：





（2）PC 机发送字符‘OPEN’时，单片机收到后，回送自己的真实姓名，并点亮一盏灯，PC 机发送字符‘CLOS’时，单片机收到后，回送自己的真实学号，并将该盏灯熄灭。

核心代码解析和运行结果截图：

这段代码是一个嵌入式系统的主函数，它包含了启动部分和主循环部分。以下是程序的功能和运行流程：

启动部分（开头）

(1.1) 声明 main 函数使用的局部变量

(1.2) 关总中断

(1.3) 给主函数使用的局部变量赋初值

(1.4) 给全局变量赋初值

(1.5) 用户外设模块初始化

(1.6) 使能模块中断

(1.7) 开总中断

主循环部分（开头）

(2.1) 主循环次数变量+1

(2.2) 未达到主循环次数设定值，继续循环

(2.3) 达到主循环次数设定值，执行下列语句，进行灯的亮暗处理

(2.3.1) 清除循环次数变量

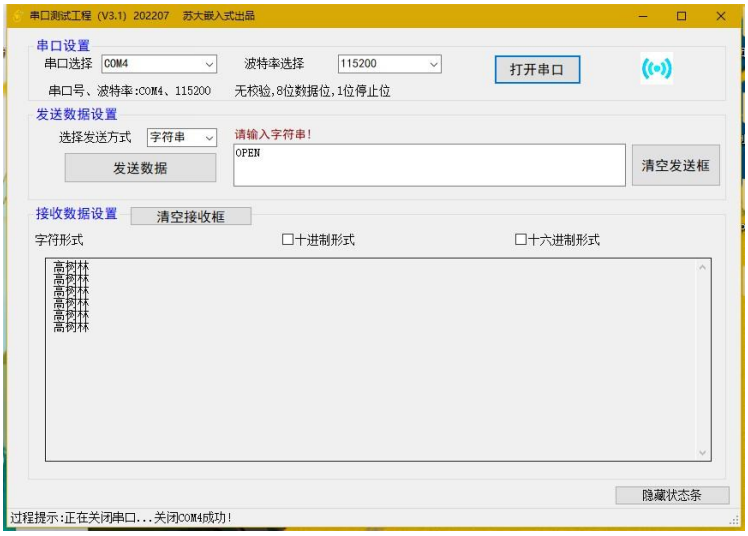
(2.3.2) 如灯状态标志 mFlag 为'L'，灯的闪烁次数+1 并显示，改变灯状态及标志，并输出学号

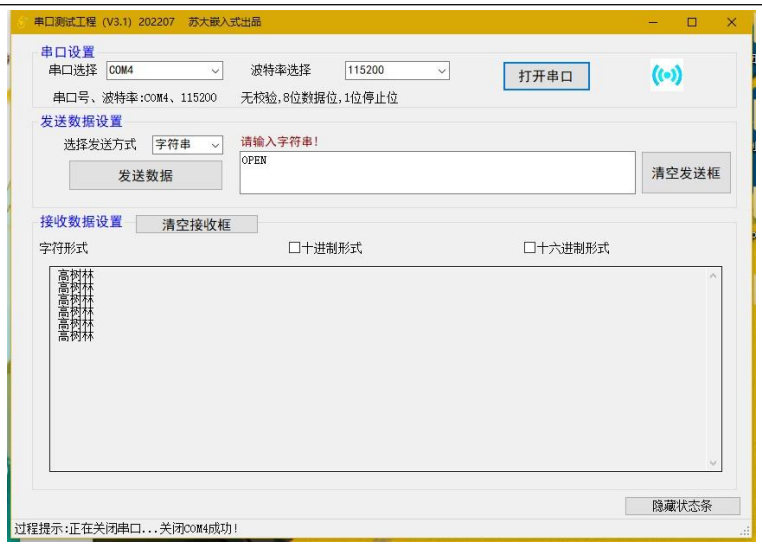
(2.3.3) 如灯状态标志 mFlag 为'A'，改变灯状态及标志如果都不是，则输出 "姓名!"。

主循环部分（结尾）

UART 构件用于实现串行通信，因此在该程序中使用了 UART 构件。在启动部分，UART_User 被初始化为波特率为 115200 的 UART 模块，并且接收中断功能被使能。在主循环部分中，当接收到 UART 数据时，中断处理程序将被调用，从而执行特定的操作。

中断处理程序的具体过程未在该代码中给出。一般来说，中断处理程序会检查中断源，并根据中断源的类型执行相应的处理操作。在 UART 接收中断的情况下，中断处理程序可能会读取 UART 缓冲区中的数据，并执行特定的操作。





3.实践性问答题

(1) UART 接收中断处理函数 UART_User_Handler（void）和 UART 构件 uart_re1(uint8_t uartNo,uint8_t *fp)均能接收一个字节，两者的区别是什么？

UART 接收中断处理函数和 UART 构件函数都可以接收一个字节，但它们的功能和使用方式不同。

UART 接收中断处理函数是一种中断服务程序，用于在 UART 接收数据时触发中断，一旦中断被触发，CPU 会跳转到该函数中执行特定的代码。这个函数通常被用来处理 UART 接收缓冲区的数据，例如将数据存储到一个缓冲区中以供后续处理。由于这个函数是在中断上下文中执行的，因此它需要快速地完成任务，以便不会影响系统的实时性能。

而 UART 构件函数是一种 API 函数，用于通过 UART 接收数据。这个函数通常被用于在主程序中读取 UART 接收缓冲区中的数据。使用这个函数时，可以将函数指针传递给它，以便在读取到数据时回调指定的函数。相比于中断处理函数，UART 构件函数更加灵活，可以在主程序中方便地进行调用，并且可以针对不同的应用场景进行自定义配置。

UART 接收中断处理函数和 UART 构件函数虽然都可以接收一个字节，但是它们的功能和使用方式是不同的。中断处理函数是在中断上下文中执行，主要用于处理 UART 接收缓冲区中的数据；而 UART 构件函数则是在主程序中执行，主要用于读取 UART 接收缓冲区中的数据。

(2)在 UART 通信时，波特率设置为 9600Baud 和 115200Baud 时，发送 1 个字节需要的时间分别是多少。

ART 通信中，发送 1 个字节需要的时间取决于波特率。波特率是指每秒钟传输的比特数，通常以 Baud 为单位表示。由于 UART 通信是异步通信，每个字节都包含一个起始位、数据位、可选的校验位和一个或多个停止位。

当波特率为 9600Baud 时，发送 1 个字节需要 10 个比特，因此发送 1 个字节的时间为 1.04ms（10÷9600 秒）。

当波特率为 115200Baud 时，发送 1 个字节需要约 0.09ms（10÷115200 秒）。

(3)利用 PC 机的 USB 口与 MCU 之间进行串行通信，为什么要进行电平转换？AHL-STM32L431 开发板中是如何实现这种电平转换的？

	在使用 PC 机的 USB 口与 MCU 之间进行串行通信时，需要进行电平转换，主要是因为 PC 机的 USB 接口使用的是 TTL（Transistor-Transistor Logic）电平，而 MCU 的串行口使用的是 CMOS（Complementary Metal-Oxide-Semiconductor）电平。这两种电平的电压值和电平定义不同，因此需要进行电平转换才能进行正常的串行通信。 在 AHL-STM32L431 开发板中，实现电平转换是通过 MAX3232 芯片来完成的。MAX3232 是一款 RS232 转 TTL 芯片，可以将 RS232 电平转换为 TTL 电平，从而实现 MCU 与 PC 机的串行通信。在开发板中，MCU 的串行口通过引脚连接到 MAX3232 芯片上，MAX3232 芯片的 TX、RX 引脚则分别连接到 USB 转串口模块的 TX、RX 引脚上，从而实现了 PC 机和 MCU 之间的串行通信，并完成了电平转换的过程。
实验总结（遇到的问题、解决过程、收获）	本次实验主要涉及到串口通信和中断。串口通信是一种常用的通信方式，它可以在不同的设备之间传输数据。而中断则是一种计算机处理器在执行程序的过程中，根据外部事件发生而中断当前程序执行的机制。本次实验通过使用串口通信和中断，实现了两个设备之间的数据传输。 在本次实验中，我们使用了两个设备，分别为 PC 机和单片机。通过使用串口通信，PC 机可以向单片机发送数据，并且单片机也可以向 PC 机发送数据。为了实现这一过程，我们需要编写两份程序，分别为 PC 机端和单片机端的程序。在单片机端，我们需要设置串口通信的参数，以及设置中断服务函数，在接收到数据时中断程序执行，从而实现数据的实时接收。而在 PC 机端，则需要使用串口通信相关的 API 函数，发送数据到单片机。 通过本次实验，我们学习了如何使用串口通信和中断，以及如何编写相应的程序。同时，本次实验也让我们更深入地了解了计算机系统中的中断机制，以及它在实际应用中的作用。这些知识对于我们理解计算机系统的工作原理和应用开发都具有重要的意义。

备注：此表格，可以根据自己的实验情况，进行格式调整。