《微控制器原理》实验报告

学院： 计算科学与人工智能学院 学号： 2017494095 姓名： 李启佳 日期： 2023/5/5

# 实验三 定时计数在PC与MCU间通信实验

## 一．实验目的

1．加强串口通信（UART）基本原理及编程原理的理解。

2．理解用定时器实现计数与定时的基本工作原理以及实现方法。

3．理解UART通信原理图（如图1）。

4．理解并掌握定时器构件的应用，实现时钟在PC上的显示。

## 二．实验内容

使用STM32 MCU为核心的实验板，另一个是PC机，通过UART相连，PC机的软件界面发送“信息”控制实验板小灯的亮灭，测试UART的通信。

实现定时器1秒钟中断一次，通过UART串口连接PC机，让PC显示计数的“时”、“分”和“秒”，各占两位，之间用“：”分隔。要求利用教材例程提供的对应模块文件，按照功能要求编写主程序和中断。综合实现不同模块共同运行的编程方法，提高编程技巧，加强编程的规范性。

## 三．实验过程

## 3.1 硬件原理图

系统分两部分：一个是使用STM32 MCU为核心的实验板，另一个是PC机，通过UART相连，PC机的软件界面发送“信息”控制实验板小灯的亮灭。掌握UART的编程知识。

（1）用 B 口的 7、9 引脚连接小灯；

（2）用 A 口的 2 、 3 引脚分别连接在TTL-USB线通向至PC。

具体连线方式见图1。

|  |
| --- |
| +5V  8  LCD  R1  MCU  4  LED  TTL电平  4  PC  图1 原理图 |

定时计数的原理图，略去小灯部分即可。

## 3.2 软件结构图

软件分MCU方程序和PC方程序。

## 3.2.1 MCU端软件结构

|  |
| --- |
| 禁止总中断  变量初始化  接小灯的引脚初始化  小灯暗  UART的初始化  否  小灯亮  是  是  否  MCU的UART接收成功  UART接收=’1’  for(;;)  图2 MCU端测试流程图 |

## 3.2.2 主程序main.c

主程序是完成系统各模块的初始化，通过主循环完成各模块测量控制工作。

**#include "Includes.h" //总头文件**

**//主函数**

**int main(void)**

**{**

**//（1）======启动部分（开头）======**

**//（1.1）声明main函数使用的局部变量**

**uint8\_t mFlag; //主循环使用的临时变量**

**uint8\_t mSec;**

**uint8\_t gT[10]; //记当前秒的值(加的）**

**//（1.2）【不变】关总中断**

**DISABLE\_INTERRUPTS;**

**wdog\_stop();**

**//（1.3）给主函数使用的局部变量赋初值**

**mFlag='A'; //主循环使用的临时变量：蓝灯状态标志**

**//（1.4）给全局变量赋初值**

**//"时分秒"缓存初始化(00:00:00)**

**gTime[0] = 0; //时**

**gTime[1] = 0; //分**

**gTime[2] = 0; //秒**

**mSec = gTime[2]; //记住当前秒的值**

**//（1.5）用户外设模块初始化**

**gpio\_init(LIGHT\_BLUE,GPIO\_OUTPUT,LIGHT\_OFF); //初始化蓝灯**

**timer\_init(TIMER\_USER,20);**

**uart\_init(UART\_User,115200); //设置TIMER\_USER为20ms中断（加的）**

**//（1.6）使能模块中断**

**timer\_enable\_int(TIMER\_USER);**

**uart\_enable\_re\_int(UART\_User); //（加的）**

**//（1.7）【不变】开总中断**

**ENABLE\_INTERRUPTS;**

**printf("------------------------------------------------------\n");**

**printf("金葫芦提示： \n");**

**printf(" （1）蓝灯闪烁\n");**

**printf(" （2）每20ms中断触发Timer定时器中断处理程序一次。 \n");**

**printf(" （3）进入Timer定时器中断处理程序后，静态变量20ms单元+1， \n");**

**printf(" （4）达到一秒时，调用秒+1，程序，计算“时、分、秒”。 \n");**

**printf(" （5）使用全局变量字节型数组gTime[3]，分别存储“时、分、秒”。 \n");**

**//printf(" （6）可通过时间测试程序C#2019测试30秒的时间间隔来校准Timer \n");**

**//printf(" （7）注意其中静态变量的使用 \n");**

**printf("------------------------------------------------------\n");**

**//for(;;) { } //在此打桩，理解蓝色发光二极管为何亮起来了？**

**//（1）======启动部分（结尾）====**

**//（2）======主循环部分（开头）===**

**for(;;) //for(;;)（开头）**

**{**

**if (gTime[2] == mSec) continue;**

**mSec=gTime[2];**

**//切换灯状态**

**if (mFlag=='A') //若灯状态标志为'A'**

**{**

**gpio\_set(LIGHT\_BLUE,LIGHT\_ON); //设置灯“亮”**

**printf("%d:%d:%d\n",gTime[0],gTime[1],gTime[2]);**

**mFlag='L'; //改变状态标志**

**}**

**else //否则,若灯状态标志不为'A'**

**{**

**gpio\_set(LIGHT\_BLUE,LIGHT\_OFF); //设置灯“暗”**

**printf("%d:%d:%d\n",gTime[0],gTime[1],gTime[2]);**

**mFlag='A'; //改变状态标志**

**}**

**uart\_sendN(UART\_User,4,"095=");**

**gT[0]=gTime[0]/10+'0';**

**gT[1]=gTime[0]%10+'0';**

**gT[2]=':';**

**gT[3]=gTime[1]/10+'0';**

**gT[4]=gTime[1]%10+'0';**

**gT[5]=':';**

**gT[6]=gTime[2]/10+'0';**

**gT[7]=gTime[2]%10+'0';**

**uart\_sendN(UART\_User,8,gT);**

**uart\_send1(UART\_User,'\r');**

**uart\_send1(UART\_User,'\n');**

**}**

**//for(;;)结尾**

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

**return 0;**

**}**

## 填写UART及定时中断程序

**void TIMER\_USER\_Handler(void)**

**{**

**DISABLE\_INTERRUPTS; //关总中断**

**static uint8\_t TimerCount = 0; //申请一个静态的计数器值**

**if(timer\_get\_int(TIMER\_USER))**

**{ //获取当前时钟溢出标志位**

**TimerCount++; //计数器累加**

**if (TimerCount >= 50){**

**TimerCount = 0; //时钟计数器清零**

**SecAdd1(gTime); //时间显示累加**

**}**

**timer\_clear\_int(TIMER\_USER); //清时钟溢出标志位**

**}**

**ENABLE\_INTERRUPTS; //开总中断**

**}**

**void SecAdd1(uint8\_t \*p)**

**{**

**\*(p+2)+=1; //秒+1**

**if(\*(p+2)>=60)**

**{ //秒溢出**

**\*(p+2)=0; //清秒**

**\*(p+1)+=1; //分+1**

**if(\*(p+1)>=60)**

**{ //分溢出**

**\*(p+1)=0; //清分**

**\*p+=1; //时+1**

**if(\*p>=24)**

**{ //时溢出**

**\*p=0; //清时**

**}**

**}**

**}**

**}**

**#include "includes.h"**

**uint8\_t g\_recvDate[20]; //串口接收字符数组**

**uint8\_t CreateFrame(uint8\_t Data,uint8\_t \* buffer); //组帧函数声明**

**void UART\_User\_Handler(void){**

**uint8\_t ch;**

**uint8\_t flag;**

**DISABLE\_INTERRUPTS; //关总中断**

**//接收一个字节**

**ch = uart\_re1(UART\_User, &flag); //调用接收一个字节的函数，清接收中断位**

**if(flag){**

**//判断组帧是否成功**

**if(CreateFrame(ch,gcRec)){**

**gTime[0]=gcRec[1];**

**gTime[1]=gcRec[2];**

**gTime[2]=gcRec[3];**

**}**

**}**

**ENABLE\_INTERRUPTS; //开总中断**

**}**

**#define FrameHead (0x3A) //帧头 ASCII码对应冒号**

**#define FrameTail1 (0x0D) //帧尾1**

**#define FrameTail2 (0x0A) //帧尾2**

**uint8\_t CreateFrame(uint8\_t Data,uint8\_t \* buffer){**

**static uint8\_t frameLen=0; //帧的计数器**

**uint8\_t frameFlag; //组帧状态**

**frameFlag=0; //组帧状态初始化**

**//根据静态变量frameLen组帧**

**switch(frameLen){**

**case 0: { //第一个数据**

**if (Data==FrameHead) { //收到数据是帧头FrameHead**

**buffer[0]=Data;**

**frameLen++;**

**frameFlag=0; //组帧开始**

**}**

**break;**

**}**

**default: { //其他情况**

**//如果接收到的不是帧尾**

**if(frameLen>=1 && Data!=FrameTail2&&buffer[frameLen-1]!=FrameTail1){**

**buffer[frameLen]=Data;**

**frameLen++;**

**break;**

**}**

**//若是末尾数据则组帧成功**

**if(Data==FrameTail2&&buffer[frameLen-1]==FrameTail1){**

**buffer[frameLen]=Data;**

**frameFlag=1; //组帧成功**

**frameLen=0; //计数清0，准备重新组帧**

**break;**

**}**

**}**

**}**

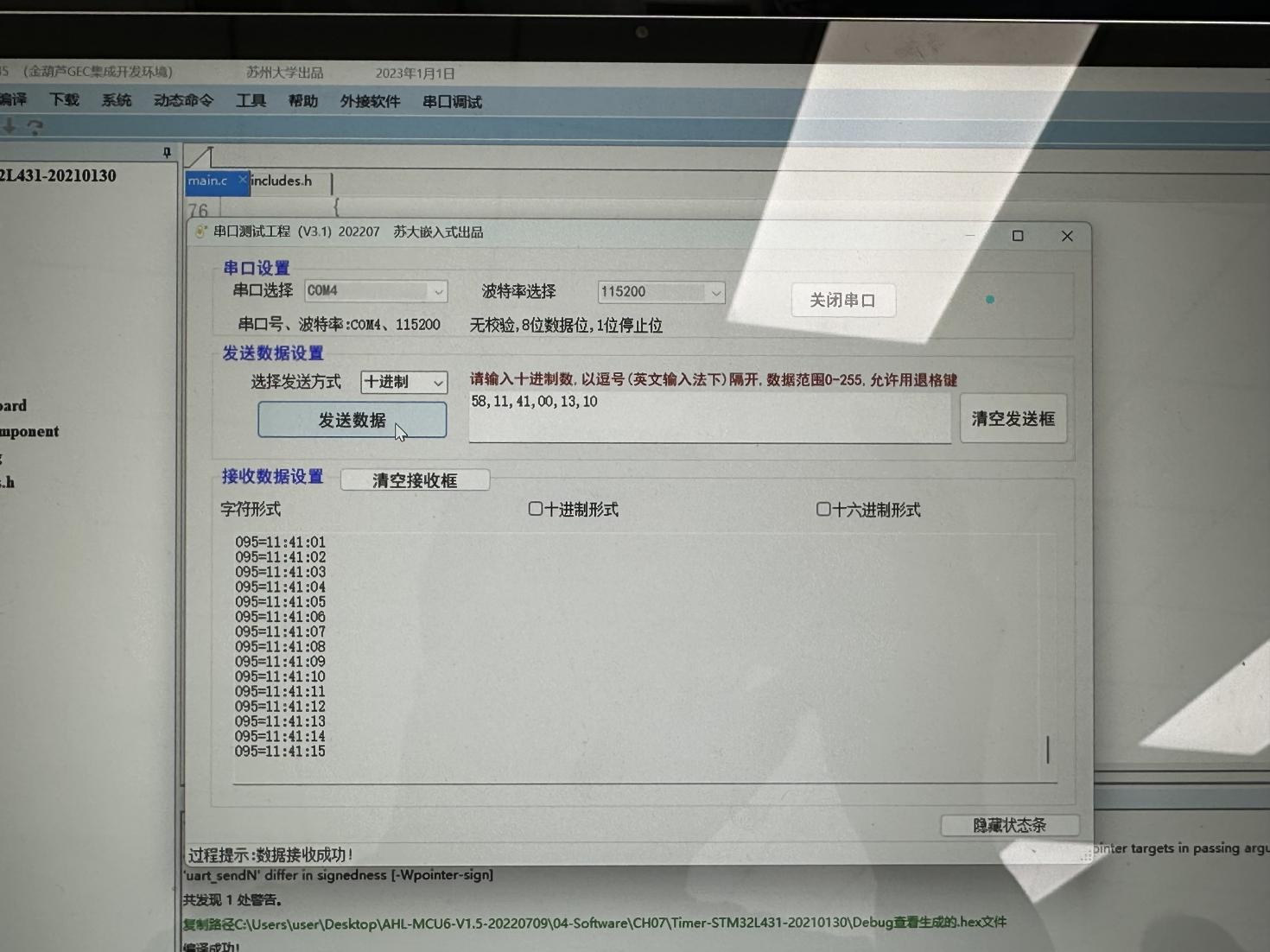
**return frameFlag; //返回组帧状态**

**}**

## 3.2.3 PC端软件

PC端软件为串口工具。

## 运行截图



## 五、回答下列问题

1、串行UART通信有哪些中断？各在什么情况下发生，作用是什么？

* 接收数据寄存器非空中断：当接收数据寄存器中有新的接收数据时，该中断会触发。它的作用是在数据接收到达时通知处理器进行接收数据处理，以避免数据丢失。
* 发送数据寄存器空中断：当发送数据寄存器为空且可以发送新的数据时，该中断会触发。它的作用是在发送缓冲区为空时通知处理器进行发送数据处理，以确保数据的连续发送。
* 帧错误中断：当接收到的数据帧中出现错误时，例如起始位、停止位或校验位错误，该中断会触发。它的作用是通知处理器数据帧存在错误，以便进行相应的错误处理或重新接收数据。
* 奇偶校验错误中断：当接收到的数据帧的奇偶校验位与设定的奇偶校验方式不匹配时，该中断会触发。它的作用是通知处理器数据的奇偶校验错误，以便进行相应的错误处理或重新接收数据。
* 帧中断：当接收线路处于空闲状态一段时间后，该中断会触发。它的作用是通知处理器数据帧的结束，用于进行相应的处理，例如判断数据接收是否完成。

2、本实验中是定时器的间隔为1秒，思考任何获得更小或更大的定时间隔？

* 使用更高/低频率的定时器：选择一个具有更高/低频率的定时器来进行定时操作
* 使用分频器：通过使用分频器来分频定时器的时钟频率，以获得更小/大的定时间隔。
* 使用定时器的预分频功能：某些定时器具有预分频功能，可以将定时器的输入时钟频率进一步分频，以实现更小/大的定时间隔。

## 六．实验小结

小结一下，通过此次实验自己所掌握的知识和感想。

通过本次实验，我对加强串口通信（UART）的基本原理和编程原理有了更深入的理解。我了解了UART通信的基本原理，包括数据的传输方式、波特率的设定以及帧结构的组成等。我也学会了使用定时器来实现计数和定时功能，并将其应用于UART通信中。我成功地实现了时钟在PC上的显示，这让我更加熟悉了定时器的应用和UART通信的实现过程。我也意识到了实验中的改良和创新的重要性。在理解了实验原理的基础上，我尝试着对实验进行改进，以提高性能或满足特定需求。这种创新的精神让我对嵌入式系统开发更加兴趣盎然。

通过这次实验，我不仅加深了对课本理论的理解，还提高了实践操作能力。我学会了如何将理论知识转化为实际应用，并通过实验中的调试和测试不断完善和优化我的代码。这种实践经验对我今后在嵌入式系统开发和通信领域的工作中将起到积极的影响。