**STM32串口实验**

本次实验采用STM32F103系列芯片USART串口的异步功能，实现两个设备之间的通讯，具体表现为：主机通过按键控制从机的LED灯亮灭。

串口设置的一般步骤可以总结为如下几个步骤：

1) 串口时钟使能和GPIO时钟使能

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1)；

2) GPIO端口模式设置

根据数据手册得：配置全双工的串口1，需要将TX(PA9)管脚需要配置为推挽复用输出，RX(PA10)管脚配置为浮空输入或者带上拉输入；

3) 开启中断并且初始化NVIC中断控制器设置优先级

配置NVIC\_InitTypeDef结构体变量：NVIC\_IRQChannel、NVIC\_IRQChannel、NVIC\_IRQChannelSubPriority和NVIC\_IRQChannelCmd

具体设置在前期报告中有具体阐述，此处不在赘述。

4) 串口参数初始化

配置USART\_InitTypeDef结构体变量：USART\_BaudRate、USART\_WordLength、USART\_StopBits、USART\_Parity、USART\_HardwareFlowControl、USART\_Mode

此处串口数据格式设置为：串口波特率：115200、8位数据、1位停止位、无奇偶校验位、无硬件数据流控制、收发模式共同使能。

5) 使能串口功能和串口接收中断

USART\_Cmd(USART1, ENABLE);

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);

6) 编写中断处理函数（开启串口接收中断）

USART1\_IRQHandle；

通过USART\_GetITStatus(USART1, USART\_IT\_RXNE)函数判断中断源。

RX端usart.c代码

1. **void** USART1\_IRQHandler(**void**)                //串口1中断服务程序
2. {
3. u8 Res;
4. **if**(USART\_GetITStatus(USART1, USART\_IT\_RXNE) != RESET)
5. //接收中断(接收到的数据必须是0x0d 0x0a结尾，0x0d 回车键  0x0a 换行符)
6. {
7. Res =USART\_ReceiveData(USART1);      //读取接收到的数据
8. **if**((USART\_RX\_STA&0x8000)==0)         //接受未完成
9. {
10. **if**(USART\_RX\_STA&0x4000)          //表示已接收到了0x0d
11. {
12. **if**(Res!=0x0a)                //判断0x0d紧跟后续不是0x0a
13. USART\_RX\_STA=0;   //如果不是则判定为结束格式错误,重新开始
14. **else**
15. USART\_RX\_STA|=0x8000;    //如果是，则判定为接收完成
16. }
17. **else**
18. {
19. **if**(Res==0x0d)
20. USART\_RX\_STA|=0x4000;    //接收0x0d时
21. **else**                    //0x0d和0x0a（结束标志）未接收到时
22. {
23. USART\_RX\_BUF[USART\_RX\_STA&0X3FFF]=Res ;
24. USART\_RX\_STA++;
25. **if**(USART\_RX\_STA>(USART\_REC\_LEN-1))
26. USART\_RX\_STA=0;//接收数据错误,重新开始接收
27. }
28. }
29. }
30. }
31. }

USART支持10个带标志的中断源：CTS改变、LIN断开符检测、发送数据寄存器空、发送完成、接收数据寄存器满、检测到总线为空闲、溢出错误、帧错误、噪音错误、校验错误。

当上述任意中断源响应时都会执行USART1\_IRQHandle函数，所以此时需要使用USART\_GetITStatus(USART1, USART\_IT\_RXNE)函数判断接收数据中断源。该实验使能串口的接收数据中断，即当有数据传输过来时，本设备立刻执行中断服务函数，将USART\_DR寄存器（由两个寄存器组成的，TDR用于发送，RDR用于接收）中的数据保存下来。

此处还添加了判断数据传输结束的协议：在每次传输的数据结尾需附加发送一个回车键符（0x0d）和一个换行符（0x0a）表示次此数据传输结束。该结束标志的使用可使得下次数据到来时覆盖上一次保存的数据同时对于便于检测当前所保存的数据内容。

RX端main.c代码：

1. **if**(USART\_RX\_STA&0x8000)   //USART\_RX\_STA清0后不再进入接收数据判断
2. {
3. **if**(USART\_RX\_BUF[0] == '0')
4. {
5. LED0 = 0;
6. USART\_RX\_STA = 0;      //USART\_RX\_STA清0，USART\_RX\_BUF[0]不变
7. }
8. **if**(USART\_RX\_BUF[0] == '1')
9. {
10. LED0 = 1;
11. USART\_RX\_STA = 0;
12. }
13. }

USART\_RX\_STA变量用于保存是否接收到数据传输结束标志。使用USART\_RX\_STA&0x8000的结果判断当前传输数据帧是否结束，当为1时，可进行所保存接受到的数据的验证。 USART\_RX\_STA = 0表示不再进行接收数据的判断，同时下次数据到来时将覆盖上次已接收的数据。该方法可避免在直接判断接收数据的相关位但是下次数据还未到来的这时间段内条件一直为真，连续执行if程序块内代码的情况。

TX端代码：

1. u8 zero[]= {0x30,0x0d,0x0a};
2. u8 one[] = {0x31,0x0d,0x0a};
4. key=KEY\_Scan(0);    //得到键值
5. **if**(key)
6. {
7. **switch**(key)
8. {
9. **case** KEY1\_PRES: //控制LED1翻转
10. **for**(t=0;t<3;t++)
11. {
12. USART\_SendData(USART1, zero[t]);//向串口1发送数据
13. **while**(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)!=SET);
14. //等待发送结束
15. }
16. **break**;
17. **case** KEY0\_PRES: //同时控制LED0,LED1翻转
18. **for**(t=0;t<3;t++)
19. {
20. USART\_SendData(USART1, one[t]);//向串口1发送数据
21. **while**(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)!=SET);
22. //等待发送结束
23. }
24. **break**;
25. }
26. }

根据ASCII码存储格式，0x30为字符’0’，0x31为字符’1’，0x0d为回车键符、0x0a为换行符。

使用USART\_SendData函数发送数组，接收端可判断USART\_RX\_BUF接收数据数组的第零位进行条件选择。

根据数据手册可得，USART\_SR寄存器的TC位置1时表示发送完成，因此可用USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)函数判断当前数据的传输是否完成，等待此处传输结束后再执行其他操作。

**SEP8000验证**

* 测试条件和目的：

在不使用PLL的情况下，验证从Normal模式进入Sleep1模式流程和从Sleep1模式退出至Normal模式流程的正确性。

* 测试步骤：

1、设置PMU\_RCCFG、PMU\_CRYCFG寄存器，分别配置32KHz和20MHz时钟源的稳定时间。用户手册中指出，Sleep1模式将系统时钟切换为32KHz晶振，所以在进入Sleep1模式之前需开启RC振荡器。同时在不使用PLL的前提下，系统时钟默认选择20MHz。

2、不使用PLL，配置PMCR寄存器不使能PLL。

3、配置PMU\_MaskWakeUp寄存器为0x02，屏蔽GPIO唤醒。

4、Sleep1模式支持GPIO和Sleeptimer唤醒，此次针对于SleepTimer唤醒进行验证，配置PMU\_SleepTimerSetValue寄存器，设置SleepTimer定时时间。

5、配置PMU\_SleepTimerCfg寄存器使能SleepTimer，并设置SleepTimer分频系数。

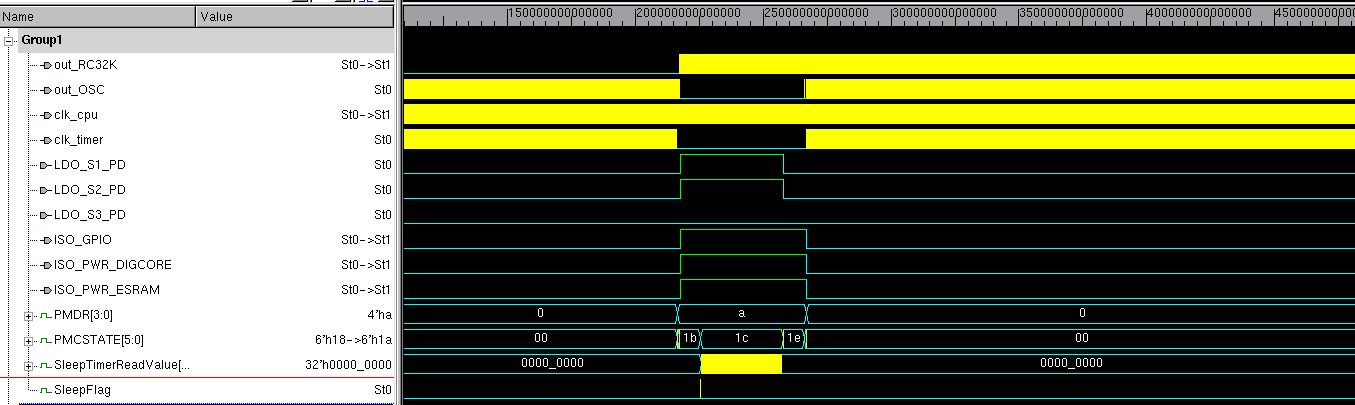
6、配置PMU\_PWROFFCFG 和PMU\_PWRONCFG寄存器 ，设置进入低功耗模式时，数字核心区的断电等待时间和从低功耗模式下退出时，数字核心区的上电稳定的等待时间。

7、配置PMDR寄存器，进入Sleep1模式；

8、SleepTimer计时结束后，Sleep1模式退出。在不使用PLL情况下，系统时钟为20M时钟，程序从头重新开始执行，配置PMU\_Int寄存器为0x7，写1清除所有中断信号；读PMU\_WakeUpRecord寄存器，可验证唤醒源是否为SleepTimer；

* 测试结果

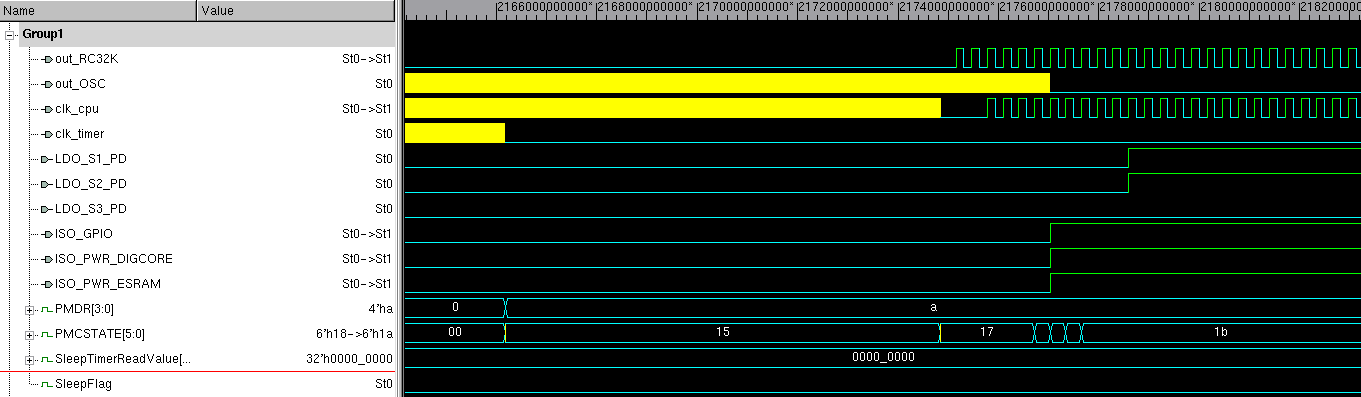
根据测试总图可以直观的看到在Sleep1模式期间系统时钟的切换和关键信号的变化。



Normal-Sleep1-Normal模式转换信号总图

1. 进入Sleep1模式期间信号变化

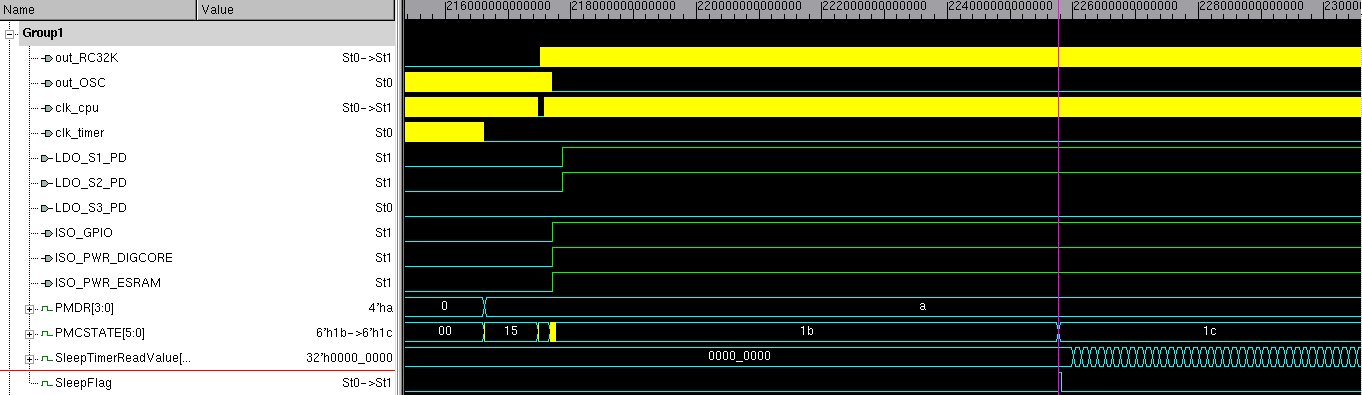
在进入Sleep1模式期间，会关闭所有外设的时钟的门控。此处以clk\_timer信号为例，在进入Normal2Sleep1模式过度期间后，受PMU的影响，时钟信号保持为0。PMCSTATE信号为0x15时，根据状态机描述，表示此时状态为WaitRCReady，等待外部RC32KHz时钟稳定后，clk\_cpu将切换为32KHz时钟。系统时钟切换为32KHz后关闭OSC20M时钟，同时LDO控制信号与ISO信号变化正常。



Normal模式进入Sleep1模式期间信号图

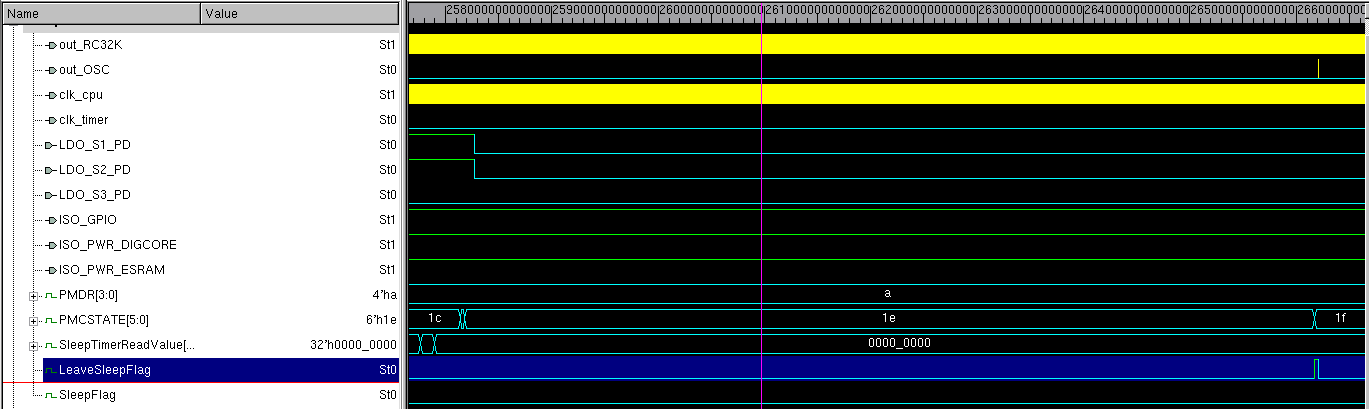
1. 进入Sleep1模式后信号变化

PMCSTATE信号为0x1B时，状态为No2Sl\_PowerOff，等待系统掉电计时（由用户配置PMU\_PWROFFCFG寄存器设定）结束后正式进入Sleep1模式。PMCSTATE信号为0x1C时，状态为Sleep。进入Sleep模式后，SleepFlag信号变化正常，SleepTimer开始计数。



SleepFlag和SleepTimer信号变化图

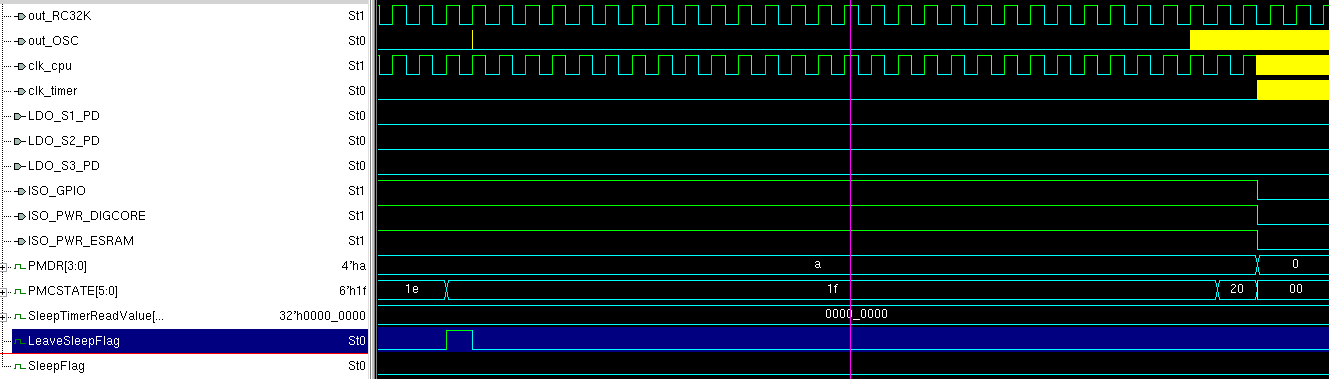
当SleepTimer计数结束后LeaveSleepFlag和LDO信号变化正常，进入退出Sleep1模式流程。



LeaveSleepFlag和SleepTimer信号变化图

1. 退出Sleep1模式期间信号变化

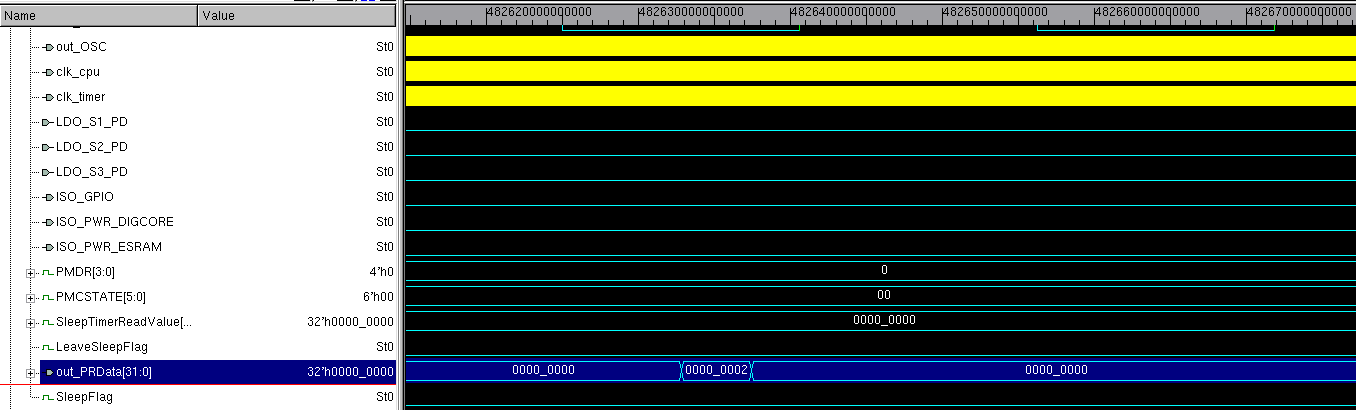
PMCSTATE信号为0x1E时，状态为Sl2No\_PowerOn。在等待系统上电计时（由用户配置PMU\_PWRONCFG寄存器设定）结束后重启OSC20M时钟源，并将系统时钟切换回20M时钟。同时ISO信号变化正常，并在回到Normal模式后将PMDR[3:0]置0表示当前状态为Normal模式。



Sleep1模式返回Normal模式期间信号图

1. 验证唤醒源

根据用户手册，从低功耗模式被唤醒返回Normal模式后，内核从第一条指令开始执行。所以在被唤醒之后通过读取PMU\_WakeUpRecord寄存器的值可验证唤醒源，此时out\_PRData信号为0x02，对比用户手册可得唤醒源为SleepTimer，与验证相符。



读取PMU\_WakeUpRecord寄存器信号