这部分的内容实现的功能是将时间传输到上位机。

第一：串口的配置，前面已经详细的讲过，这里不再讲解。

第二：中断的配置，由于我们需要时间每秒自加一次，一次需要用到中断，在中断服务程序中实现秒的自加，前面也已讲过中断的配置，这里就不详细讲解，只给出代码

**/\***

**\* 函数名：NVIC\_Configuration**

**\* 描述 ：配置RTC秒中断的主中断优先级为1，次优先级为0**

**\* 输入 ：无**

**\* 输出 ：无**

**\* 调用 ：外部调用**

**\*/**

**void NVIC\_Configuration(void)**

**{**

**NVIC\_InitTypeDef NVIC\_InitStructure;**

**/\*设置先占优先级1位，从占优先级3位\*/**

**NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_1);**

**/\*选择RTC的IRQ通道\*/**

**NVIC\_InitStructure.NVIC\_IRQChannel =RTC\_IRQn;**

**/\*设置中断先占优先级为1\*/**

**NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority =1;**

**/\*设置中断从占优先级为1\*/**

**NVIC\_InitStructure.NVIC\_IRQChannelSubPriority =0;**

**/\*使能RTC的IRQ通道\*/**

**NVIC\_InitStructure.NVIC\_IRQChannelCmd =ENABLE;**

**NVIC\_Init(&NVIC\_InitStructure);**

**}**

第三：对于RTC的使用，首先我们要判断是否是第一次使用RTC，如果是第一次使用那么肯定要设置时间的初始值以及对RTC进行相应的配置，而如果不是第一次使用，那么我们就无需再设置时间的初始值以及对RTC进行相应的配置，只需让RTC计数器继续计数就可以了。那么我们如何才能判断RTC是否为第一次使用呢？STM32中有一个后备寄存器，寄存器中的值不会因为掉电而改变，既然如此那我们肯定会这样想，当我第一次使用RTC时，往后备寄存器中写入一个值，下次再使用RTC时，我只要判断后备寄存器中的值是否为我第一次用RTC时写入的值，如果相等，说明我以前已经用过RTC了，现在我无需再对RTC进行配置了，因为第一次都配置好了（RTC和后备寄存器一样，RTC寄存器中设置的值不会因为掉电而改变），但要注意的是RTC的允许中断这一位在每次复位后会回到默认值，所以每次复位后我们都要再次设置允许RTC中断。下面我们就看看RTC的配置程序：

**/\***

**\* 函数名：RTC\_Configuration**

**\* 描述 ：配置RTC**

**\* 输入 ：无**

**\* 输出 ：无**

**\* 调用 ：外部调用**

**\*/**

**void RTC\_Configuration(void)**

**{**

**/\* 使能PWR和BKP时钟 \*/**

**RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_PWR | RCC\_APB1Periph\_BKP, ENABLE);**

**/\*取消后备区域的写保护，因为后备寄存器中放的是重要的数据，默认是不允许往里面写入值的 \*/**

**PWR\_BackupAccessCmd(ENABLE);**

**/\* 将后背寄存器的寄存器值设为默认值 \*/**

**BKP\_DeInit();**

**/\* 打开外部低速晶振，RTC可以选择的时钟源是外部和内部低速晶振及外部高速晶振，这里我们选择外部低速晶振32768HZ \*/**

**RCC\_LSEConfig(RCC\_LSE\_ON);**

**/\*等待外部低速晶振准备就序\*/**

**while(RCC\_GetFlagStatus(RCC\_FLAG\_LSERDY) == RESET)**

**{}**

**/\*选择外部低速晶振为RTC的时钟源\*/**

**RCC\_RTCCLKConfig(RCC\_RTCCLKSource\_LSE);**

**/\* Enable RTC Clock \*/**

**RCC\_RTCCLKCmd(ENABLE);**

**/\* 等待RTC寄存器与RTC的APB时钟同步 \*/**

**RTC\_WaitForSynchro();**

**/\* 等待上次对RTC寄存器配置完成 \*/**

**RTC\_WaitForLastTask();**

**/\* 使能RTC中断 \*/**

**RTC\_ITConfig(RTC\_IT\_SEC, ENABLE);**

**/\* 等待上次对RTC寄存器配置完成 \*/**

**RTC\_WaitForLastTask();**

**/\* 设置RTC的预分频值，因为外部低速晶振是32768，所以选择 \*/**

**/\* RTC计数器计数频率= RTCCLK/RTC\_PR = (32.768 KHz)/(32767+1) \*/**

**RTC\_SetPrescaler(32767);**

**/\* 等待上次对RTC寄存器配置完成 \*/**

**RTC\_WaitForLastTask();**

**}**

**下面再来看看在RTC中断中写了哪些程序：**

**/\***

**\* 函数名：RTC\_IRQHandler(void)**

**\* 描述 ：RTC中断服务函数**

**\* 输入 ：无**

**\* 输出 ：无**

**\* 调用 ：无**

**\*/**

**void RTC\_IRQHandler(void)**

**{ /\*判断中断标志位是否被置位\*/**

**if (RTC\_GetITStatus(RTC\_IT\_SEC) != RESET)**

**{**

**/\* 清除中断标志位 \*/**

**RTC\_ClearITPendingBit(RTC\_IT\_SEC);**

**/\* TimeDisplay是一个标志位，只有等于1时才让串口发送时间数据，即让串口一秒发一次时间值 \*/**

**TimeDisplay = 1;**

**/\* Wait until last write operation on RTC registers has finished \*/**

**RTC\_WaitForLastTask();**

**/\* 当时间走到23:59:59秒时RTC计数器中的值清零，0x00015180=23\*3600+56\*60+59\*/**

**if (RTC\_GetCounter() == 0x00015180)**

**{**

**RTC\_SetCounter(0x0);**

**/\* Wait until last write operation on RTC registers has finished \*/**

**RTC\_WaitForLastTask();**

**}**

**}**

**}**

**第四：RTC配置好后肯定要对初始时间的设置进行相应的配置了，首先是如何从串口中接受设置的时间初始值：**

**/\***

**\* 函数名：USART\_Scanf**

**\* 描述 ：串口从超级终端中获取数值**

**\* 输入 ：- value 用户在超级终端中输入的数值**

**\* 输出 ：无**

**\* 调用 ：被Time\_Regulate(void)调用**

**\*/**

**uint8\_t USART\_Scanf(uint32\_t value)**

**{**

**uint32\_t index = 0;**

**uint32\_t tmp[2] = {0, 0};**

**while (index < 2)**

**{**

**/\* 等待数据接受完成 \*/**

**while (USART\_GetFlagStatus(USART1, USART\_FLAG\_RXNE) == RESET)**

**{}**

**/\*将数据给数组tmp, 从串口终端里面输进去的数是ASCII码值**

**\*/**

**tmp[index++] = (USART\_ReceiveData(USART1));**

**/\*判断接受到的数据是否在0到9之间\*/**

**if ((tmp[index - 1] < 0x30) || (tmp[index - 1] > 0x39))**

**{**

**printf("\n\rPlease enter valid number between 0 and 9");**

**index--;**

**}**

**}**

**/\*将接受的两个数据组成一个两位数 \*/**

**index = (tmp[1] - 0x30) + ((tmp[0] - 0x30) \* 10);**

**/\* 判断组成后的值是否有无，比如组成的小时不能超过23，分秒不能超过59，value 即为23，或59\*/**

**if (index > value)**

**{**

**printf("\n\rPlease enter valid number between 0 and %d", value);**

**return 0xFF;**

**}**

**return index;**

**}**

**上面的函数是被下面的Time\_Regulate(void)函数调用的。**

**/\***

**\* 函数名：Time\_Regulate**

**\* 描述 ：返回用户在超级终端中输入的时间值，并将值储存在**

**\* RTC 计数寄存器中。**

**\* 输入 ：无**

**\* 输出 ：用户在超级终端中输入的时间值，单位为 s**

**\* 调用 ：被Time\_Adjust调用，调用USART\_Scanf**

**\*/**

**uint32\_t Time\_Regulate(void)**

**{**

**uint32\_t Tmp\_HH = 0xFF, Tmp\_MM = 0xFF, Tmp\_SS = 0xFF;**

**printf("\r\n==============Time Settings=====================================");**

**printf("\r\n Please Set Hours");**

**while (Tmp\_HH == 0xFF)**

**{**

**/\*将串口接受的数据给Tmp\_HH ，23便是传递给USART\_Scanf（value）中的形参value的\*/**

**Tmp\_HH = USART\_Scanf(23);**

**}**

**printf(": %d", Tmp\_HH);**

**printf("\r\n Please Set Minutes");**

**while (Tmp\_MM == 0xFF)**

**{**

**Tmp\_MM = USART\_Scanf(59);**

**}**

**printf(": %d", Tmp\_MM);**

**printf("\r\n Please Set Seconds");**

**while (Tmp\_SS == 0xFF)**

**{**

**Tmp\_SS = USART\_Scanf(59);**

**}**

**printf(": %d", Tmp\_SS);**

**/\* 将时分秒转换成秒放入RTC计数器中\*/**

**return((Tmp\_HH\*3600 + Tmp\_MM\*60 + Tmp\_SS));**

**}**

**/\***

**\* 函数名：Time\_Adjust**

**\* 描述 ：时间调节**

**\* 输入 ：无**

**\* 输出 ：无**

**\* 调用 ：主函数调用**

**\*/**

**void Time\_Adjust(void)**

**{**

**/\* Wait until last write operation on RTC registers has finished \*/**

**RTC\_WaitForLastTask();**

**/\* 将设置的初始时间值装入RTC计数器，RTC开始运行时，计数器里面的值会在初始值的基础上自动一秒加1次\*/**

**RTC\_SetCounter(Time\_Regulate());**

**/\* Wait until last write operation on RTC registers has finished \*/**

**RTC\_WaitForLastTask();**

**}**

**时间的初始值设置好之后肯定是让时间正常的走起来，同时在上位机上显示时间**

**/\***

**\* 函数名：Time\_Show**

**\* 描述 ：在超级终端中显示当前时间值**

**\* 输入 ：无**

**\* 输出 ：无**

**\* 调用 ：主函数调用，调用函数Time\_Display(uint32\_t TimeVar)**

**\*/**

**void Time\_Show(void)**

**{**

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

**while (1)**

**{**

**/\* 只有TimeDisplay 为1时才在上位机上显示时间，TimeDisplay是用户自己定义的标示符，在RTC中断服务函数中被置1，即1秒被置1一次，也就是1秒才让串口向上位机发一次时间\*/**

**if (TimeDisplay == 1)**

**{**

**/\* 将RTC计数器中的值传给函数 Time\_Display \*/**

**Time\_Display(RTC\_GetCounter());**

**TimeDisplay = 0;**

**}**

**}**

**}**

**/\***

**\* 函数名：Time\_Display**

**\* 描述 ：显示当前时间值**

**\* 输入 ：-TimeVar RTC计数值，单位为 s**

**\* 输出 ：无**

**\* 调用 ：被Time\_Show调用**

**\*/**

**void Time\_Display(uint32\_t TimeVar)**

**{**

**uint32\_t THH = 0, TMM = 0, TSS = 0;**

**/\* 将极速器中的值换成时分秒分别赋给THH ，TMM，TSS \*/**

**THH = TimeVar / 3600;**

**TMM = (TimeVar % 3600) / 60;**

**TSS = (TimeVar % 3600) % 60;**

**/\*将时间传到上位机上显示\*/**

**printf(" Time: %0.2d:%0.2d:%0.2d\r", THH, TMM, TSS);**

**}**

**到此程序全部写完了，是不是看的有点晕头转向啊，那就正常了，哥可是弄了一天才理顺的，好好多看几遍吧。**

**好的下面我们再看看主函数是按什么顺序调用上面的函数的**

**#include "stm32f10x.h"**

**#include "usart1.h"**

**#include "rtc.h"**

**/\***

**\* 函数名：main**

**\* 描述 ：主函数**

**\* 输入 ：无**

**\* 输出 ：无**

**\* 调用 ：调用外部函数USART1\_Config()，ADC1\_Init()，printf**

**\*/**

**int main(void)**

**{**

**/\* USART1 config \*/**

**USART1\_Config();**

**/\* 配置RTC秒中断优先级 \*/**

**NVIC\_Configuration();**

**/\*通过读取后备寄存器中的值来判断是否为第一次使用RTC，0xA4A5只是我们自己随便规定的一个数\*/**

**if(BKP\_ReadBackupRegister(BKP\_DR1) !=0xA4A5)**

**{**

**/\* RTC Configuration \*/**

**RTC\_Configuration();**

**Time\_Adjust();**

**BKP\_WriteBackupRegister(BKP\_DR1, 0xA4A5);**

**}**

**/\*这句语句千千万万不可以丢，因为掉电或复位后RTC的中断允许位为会被恢复到默认值，而不会像RTC其他的配置位掉电后也会保持不变，如果我们不是第一次使用RTC，上电后系统不会运行上面的if句，如果再没有下面的开RTC中断语句，那么RTC不会进入中断，当然也就不会执行中断中的中断服务函数，所以这句不可少，亲！\*/**

**RTC\_ITConfig(RTC\_IT\_SEC, ENABLE);**

**Time\_Show();**

**while (1)**

**{**

**}**

**}**

**聪明名的孩子肯定要疑问，如果再掉电的期间RTC计时到23:59:59秒咋办，因为掉电的情况下是不执行中断服务函数的，也就是说无法实现走到23:59:59RTC自动清0，哎，STM32对此也无能为力了。所以RTC时钟跟DS1302时钟芯片还是有不小的差距的。**