STM32 入门系列教程

串口的编程

Revision 0.01

(2010-04-21)

很多书籍或者教程，在介绍串口的时候，总会先介绍 UART 的功能多么强大。在这里，笔者认为，功能强大，可以红外，可以流控，那跟我有啥关系？我只要让串口成为我的工具即可。我们对串口编程，熟悉串口，只是想利用串口来调试信息等。

先别忙着看书，大家对串口编程前，应该明确如下几个问题：

1、串口的作用：用在 STM32 板子和 PC 机通信的。我们调试的时候，无法知道是否正确，就可以用 STM32 的 cpu，给串口输出一些信息给 PC，我们通过屏幕（实际上是终端串口软件），可以看到这些信息，从而知道当前程序的错误可能出现的位置。当然，也可以在 PC 的键盘敲打命令，让串口帮传递给 STM32 板子，来执行这些命令。

2、串口到底如何工作的？一般有两种方式：查询和中断。

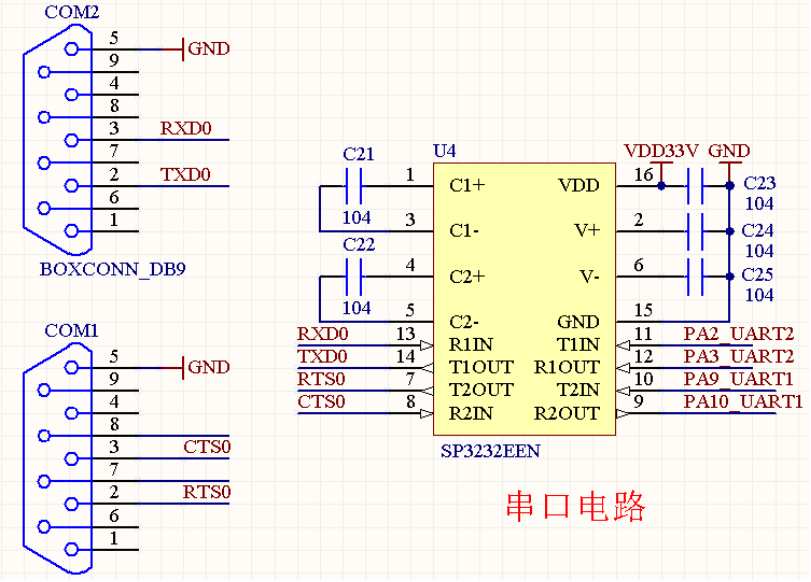
（1）查询：串口程序不断地循环查询，看看当前有没有数据要它传送。如果有，就帮助传送（可以从 PC 到 STM32 板子，也可以从 STM32 板子到 PC）。

（2）中断：平时串口只要打开中断即可。如果发现有一个中断来，则意味着要它帮助传输数据——它就马上进行数据的传送。同样，可以从 PC 到 STM32 板子，也可以从 STM32 板子到 PC。

3、如何通过编程，让串口工作在查询或中断方式下？由于现实项目中，一般采用中断方式来处理串口，为方便大家编程，我们在例程目录中，给出了中断和查询两种方式的代码，仅供参考。

OK，明白以上三个问题后，让我们开始串口的学习吧！我们分几个步骤来学习串口。

步骤一　从硬件开始学习。大家先打开芯达 STM32 开发板附带的原理图。找到串口部分。笔者把它截图如下。我们发现，串口模块的电路是这样的：STM32 的 CPU 引脚，通过两个 PA 端口的引脚 PA10 和 PA9，连接到一个 SP3232 芯片，或者 MAX232 芯片。然后再连接到 DB9 串口座上。由于 232 芯片可以允许走两路信号，因此，我们扩展了一个串口 COM2，请注意，如无特别说明，我们都将使用 COM1。



SP3232EEN 芯片能帮助把数据信号转换成电脑 232 接口能识别的信号。其转换是自动进行的。因此，我们只需要把要发送的数据送给引脚 PA9，然后再串口

座的引脚 3 上去接收数据即可。反之，接收数据也是一样。这里您可能有个疑问，为啥使用 PA9 和 PA10？

大家下载 STM32 的 datasheet 文档后，在 GPIO 的复用功能章节，即可找到原因：原来这两个引脚是 USART 复用的，呵呵。另外，固件函数中的串口例子，使用的也是 PA9 和 PA10。当然还有其他复用的 GPIO 引脚，比如 PA2，PA3 等。

步骤二　初始化串口。

请您打开《STM32F103xxx 参考手册》与《STM32 固件库使用手册》。我们的思路和之前一样，根据固件库使用手册中给出的步骤来配置串口。

1、要声明一个结构：

```
GPIO_InitTypeDef  GPIO_InitStructure;
USART_InitTypeDef USART_InitStructure;
```

这里顺便也声明了 GPIO 的结构。原因是：串口是需要使用 IO 口来进行发送和接收的。

2、设置该结构中的成员：

串口的结构成员设置如下：

```
USART_StructInit(&USART_InitStructure);
USART_InitStructure.USART_BaudRate = 115200;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
```

GPIO 结构的成员设置如下：

```
GPIO_StructInit(&GPIO_InitStructure);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA , &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

3、调用函数 xxx_Init()来初始化外设（包括 IO 和 USART）：

USART 的初始化函数：`USART_Init(USART1 , &USART_InitStructure);`

由于 GPIO 的外设初始化已经放在成员设置后面，因此这里没有列出来。

4、调用 xxx_Cmd(xxx , ENABLE);函数来使能外设。

这里只需要使能 USART 即可。GPIO 的固件中，没有使能即可使用。

```
USART_Cmd(USART1 , ENABLE);
```

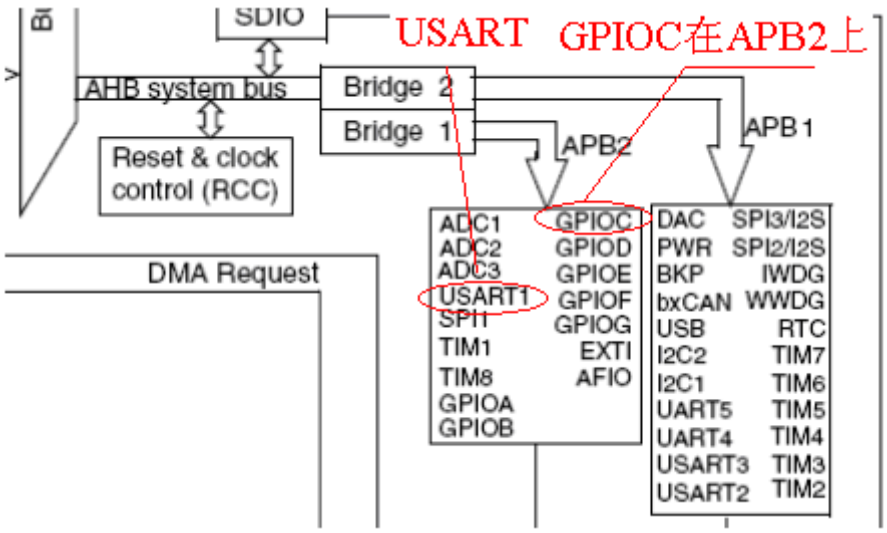
由于本文档使用的是中断方式来触发串口收发数据，因此，我们在使能串口之前，也把发送和接收的中断使能打开：

```
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
USART_ITConfig(USART1, USART_IT_TXE, ENABLE);
```

5、别忘了加上外设的时钟使能哈~

从下页的截图，可以看出，GPIOA 和 USART 都在 APB2 上。因此我们调用的函数如下：

```
RCC_APB2PeriphClockCmd(  RCC_APB2Periph_USART1 |RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOE, ENABLE);
```



步骤三　操作串口收发数据

芯达 STM32 的思路是：首先让 STM32 的 CPU 发送一些欢迎信息，从串口打印出来。然后接收中断，该中断来自键盘输入。只要键盘输入一个字符，即打印出该字符，从而验证串口编程是否成功。

固件库的函数如何让串口发送和接收数据呢？它给我们提供了两个函数：

USART_SendData();　　——省略函数参数

USART_ReceiveData();

先来看发送。我们的程序在初始化串口之后，马上就会打印欢迎信息。也就是从 STM32 的 CPU 发送信息，在电脑屏幕上显示出来，只要如下操作即可：

```
/*====================USART 打印欢迎信息======================*/
    for( i = 0; TxBuf1[i] != '\0'; i++) {
        USART_SendData(USART1 , TxBuf1[i]);
        while(USART_GetFlagStatus(USART1, USART_FLAG_TC)==RESET);
        }
```

很简单吧，呵呵，TxBuf1 是一个发送缓存。我们事先把数据放在这个数组里了：

```
unsigned char TxBuf1[100] = "这里可以自定义需要输出到串口的字符";
```

这里要注意的是，当我们发送一个字符后，必须查看状态标志，这里是发送是否完成的标志：USART_FLAG_TC，如果发送完成了，则才可以发送下一个数据。函数 USART_GetFlagStatus();就是用来做这个判断的。

下面开始另外一个操作：键盘输入什么字符，就得显示什么字符。我们的代码如下：

```
    while (1)
    {
```

```
            GPIO_SetBits(GPIOE, GPIO_Pin_1);
            RX_status = USART_GetFlagStatus(USART1, USART_FLAG_RXNE);
            if(RX_status == SET) {
                USART_SendData(USART1 , USART_ReceiveData(USART1));
                while(USART_GetFlagStatus(USART1,
USART_FLAG_TC)==RESET);
                GPIO_ResetBits(GPIOE, GPIO_Pin_1);
                Delay(0xFFFFF);
            }
        }
```

思路：先判断接收的状态标志 USART_FLAG_RXNE，如果接收的寄存器非空，说明已经接收到键盘发送来的数据，于是就把这个数据从接收缓存中取出来，发送给电脑。这样电脑就可以看到刚才敲入的字符了。

USART_ReceiveData(USART1)；这个函数是从接收缓存取出数据。

我们在这个 while 中，还加入了 GPIO 的函数：

GPIO_SetBits(GPIOC, GPIO_Pin_0);

GPIO_ResetBits(GPIOC, GPIO_Pin_0);

这两个函数，SetBits，表示对 GPIOC 端口的第 0 个引脚置 1。ResetBits，清零。由于芯达 STM32 开发板上，GPIOC 端口上的第 0 引脚连着一个 LED 灯，所以，我们可以通过观察 LED 灯是否闪烁来判断串口是否正在发送。

串口的编程，写到这里，已经进入尾声。如果您对串口操作还有不明白的地方，请直接到我们的官方网站：ARM 技术交流网 www.arm79.com，进行讨论。我们将会尽快给您做出答复。

附：

福州芯达工作室简介

福州芯达工作室成立于 2009 年 9 月，我们专注于嵌入式产品的研发与推广，目前芯达产品涉及 ARM9 系列、STM32 系列。

芯达团队成员均硕士研究生毕业，具有一定研发实力。我们的愿景在于把福州芯达打造成国内一流的嵌入式品牌。或许我们现在做的还不够，但是我们真的努力在做，希望通过我们的努力，能够在您学习和使用芯达产品的过程中带来或多或少的帮助。

这是芯达为了配合 STM32 开发板而推出的入门系列教程。如果您在看了我们的教程后，理清了思路，我们都会倍感欣慰！让我们一起学习，共同进步，在征服嵌入式领域的道路上风雨同行！

官方网站：http://www.arm79.com/

官方淘宝：http://shop36353570.taobao.com/