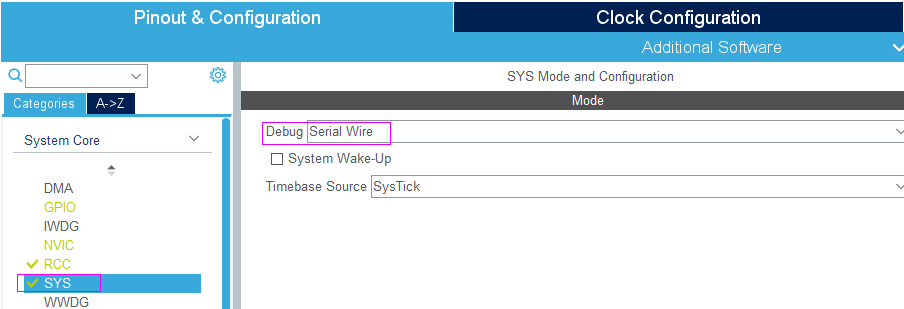
有块STM32的板子，想试试HAL库的功能，经典的点灯程序试验完毕，说明板子没问题。于是想试试串口通信如何，  
    通过CUBEMX配置后，简单测试后发现从STM32串口乱码，即从STM32串口发送出来的数据，在PC端的串口调试助手一直乱码。  
    第一反应是板子晶振问题，这确实是乱码产生的一个因素。  
  
    但是，经过仔细确认，板子不存在这个问题。经仔细查看了原理图，是MINI板的问题，其串口的RX,TX确实引出来了，  
    但是没有232的电平转换，因此直接用杜邦线将排针上的RX,TX直接与PC的串口对应线连接是不能正确的，必须乱码！  
      
    总得通信起来吧，那就USB虚拟串口方式了，于是就有了这篇笔记，详细过程看笔记吧...

**注：**

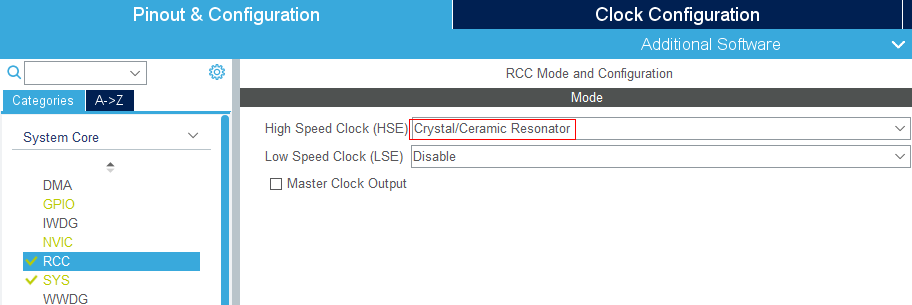
**板子是用张飞的mini板**

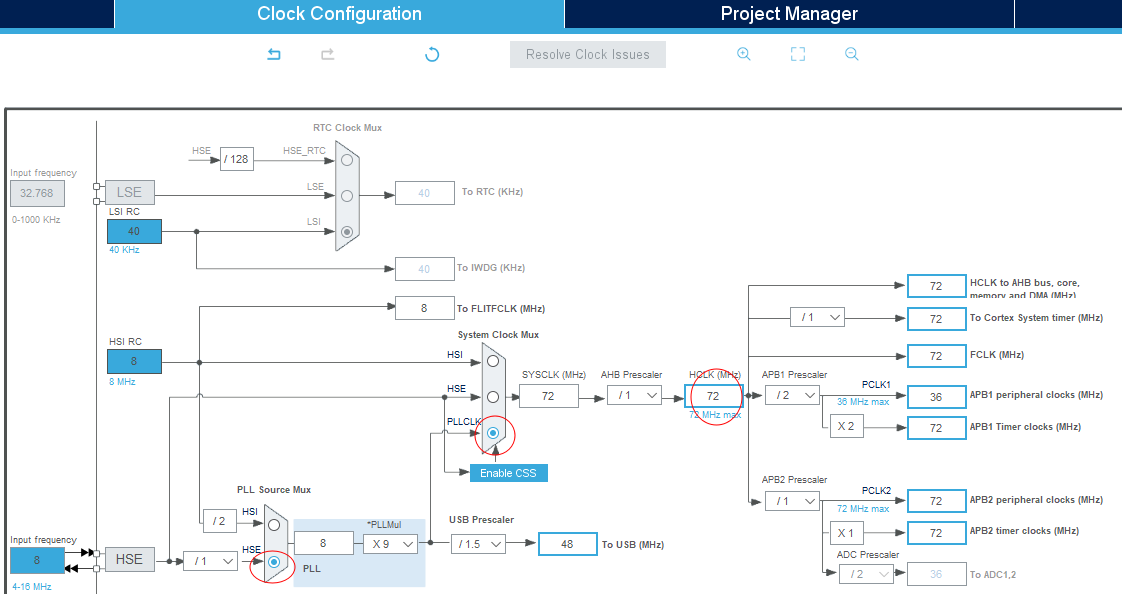
**CubeMX配置：**

根据选择的芯片，逐步完成配置即可



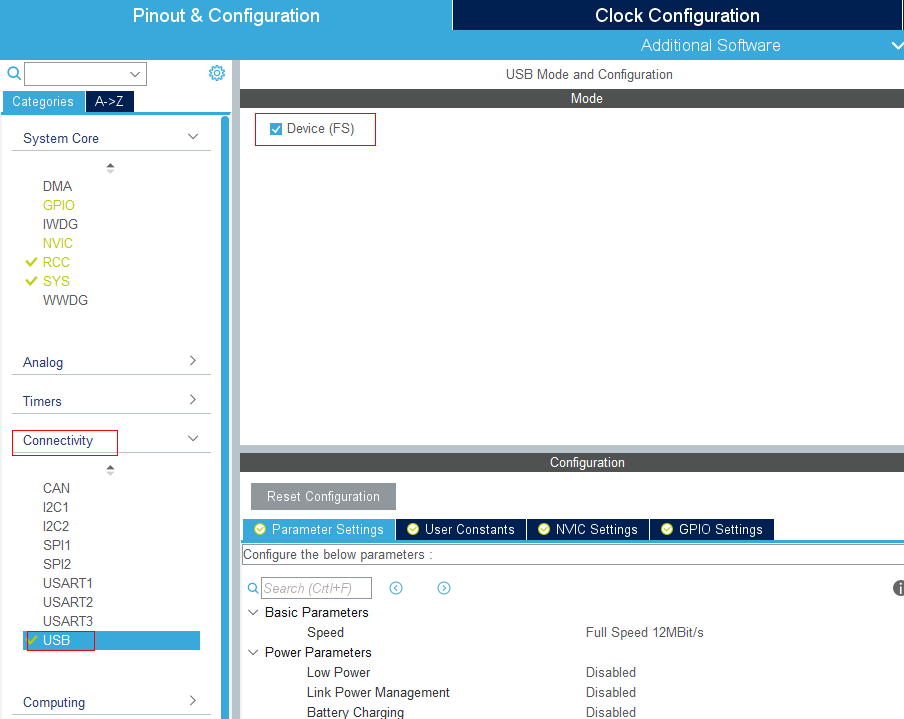
**时钟：**

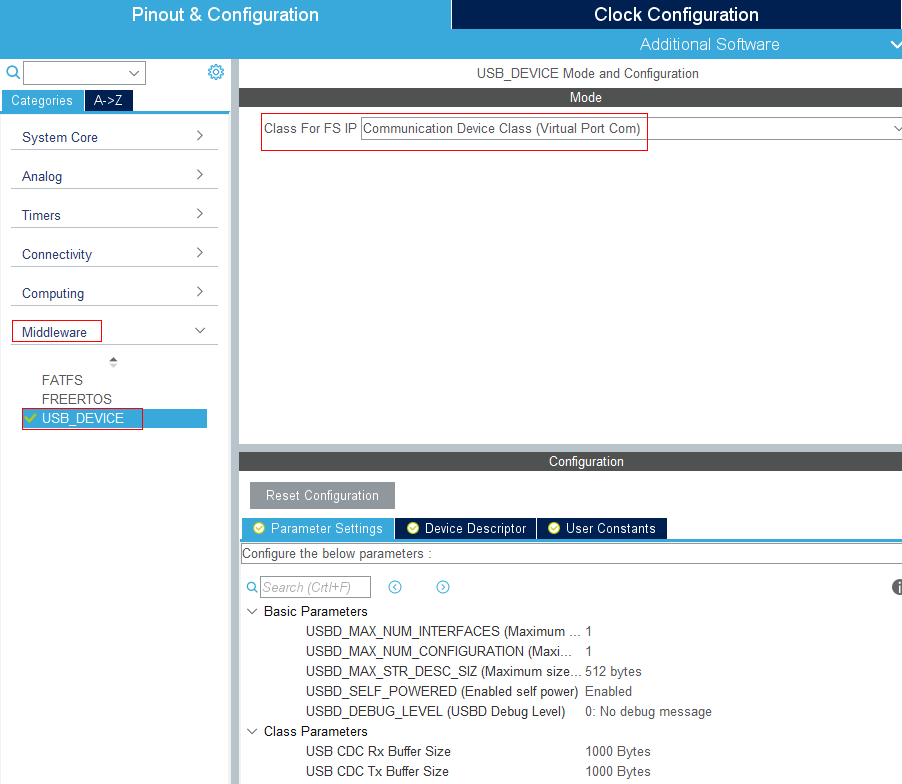




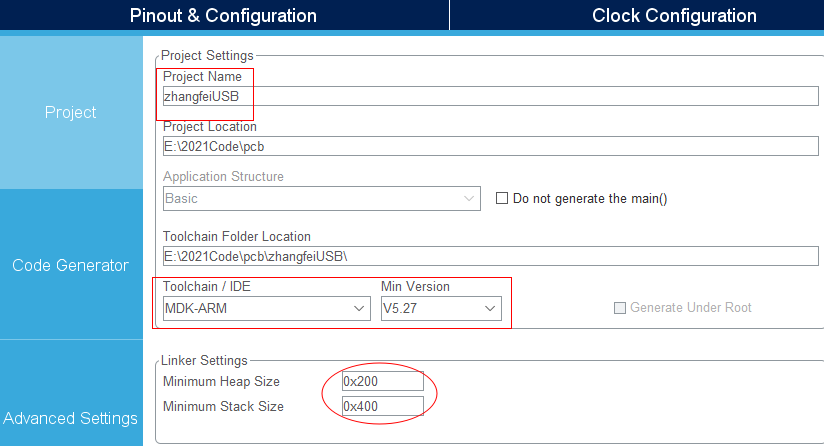
选中前两个圈后，在第三个圈直接填写板子支持的最大时钟：72MHz即可自动完成配置。

**USB：**

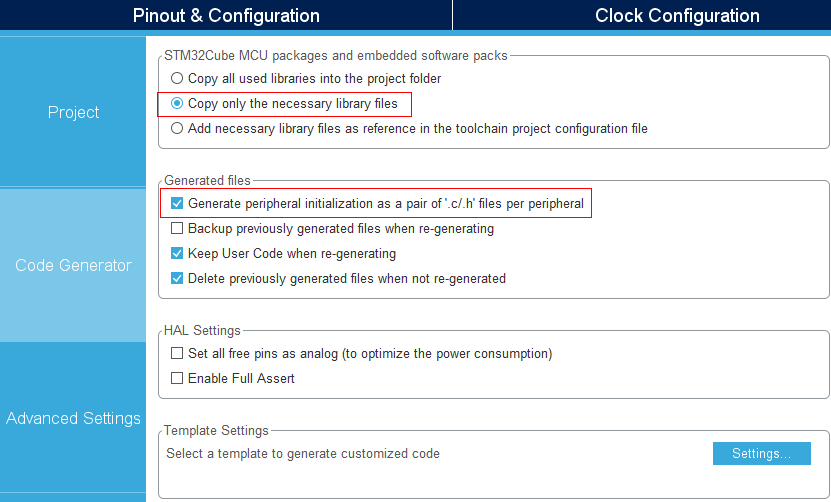








这个地方可以都是设置0x400，也可以到代码工程里修改。



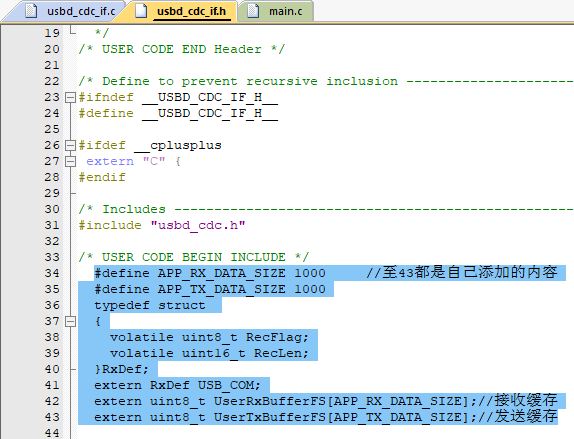


生成工程即可。

打开工程后，开始添加代码。

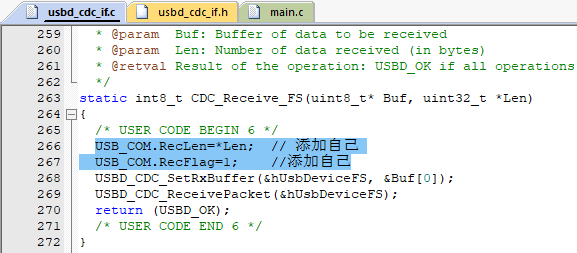
首先，在usbd\_cdc\_if.h 文件添加如下代码：

|  |
| --- |
| #define APP\_RX\_DATA\_SIZE 1000 //至43都是自己添加的内容  #define APP\_TX\_DATA\_SIZE 1000  typedef struct  {  volatile uint8\_t RecFlag;  volatile uint16\_t RecLen;  }RxDef;  extern RxDef USB\_COM;  extern uint8\_t UserRxBufferFS[APP\_RX\_DATA\_SIZE];//接收缓存  extern uint8\_t UserTxBufferFS[APP\_TX\_DATA\_SIZE];//发送缓存 |



接着，在usbd\_cdc\_if.c 文件添加如下代码：

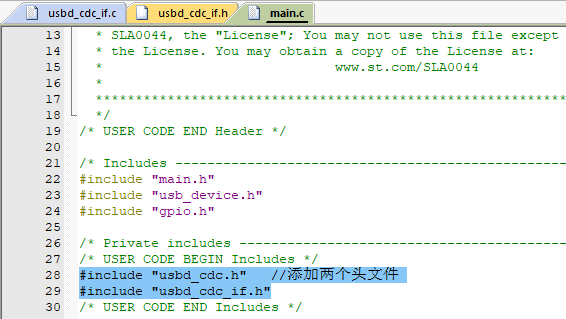
|  |
| --- |
| USB\_COM.RecLen=\*Len; // 添加自己  USB\_COM.RecFlag=1; //添加自己 |



最后，在main.c文件添加代码：

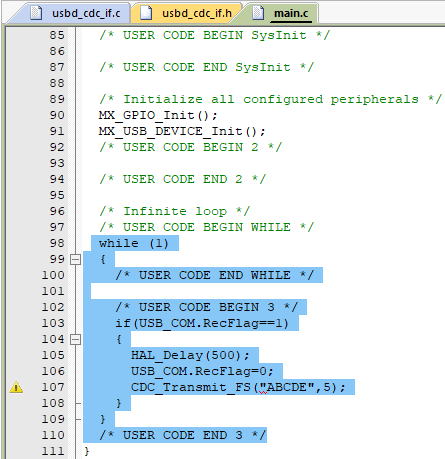
先添加2个头文件：

|  |
| --- |
| #include "usbd\_cdc.h" //添加两个头文件  #include "usbd\_cdc\_if.h" |

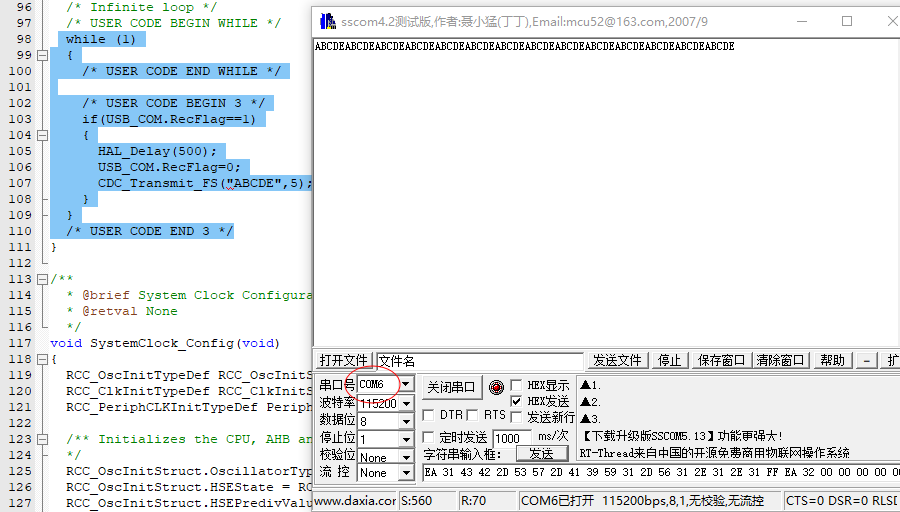


再添加发送数据代码

|  |
| --- |
| while (1)  {  /\* USER CODE END WHILE \*/  /\* USER CODE BEGIN 3 \*/  if(USB\_COM.RecFlag==1)  {  HAL\_Delay(500);  USB\_COM.RecFlag=0;  CDC\_Transmit\_FS("ABCDE",5);  }  }  /\* USER CODE END 3 \*/ |



**测试：**



PC端随意发数据，每发一次则SMT32回应固定的字符串“ABCDE”,说明通信OK。

当然，可以实现自定义协议，来控制板子上做一些操作。比如PC发一串命令包下来，板子端可以解析PC来的命令包，控制关LED,开LED等等。

**备注问题：**

USB无法识别：

即cubeMX配置完成时，且未添加任何代码，直接编译，下载到板子后，接USB线后，无法识别。

1、查一下说堆和栈太小了，要大一点，两种 方法：

1、MX时，直接都搞成0x400

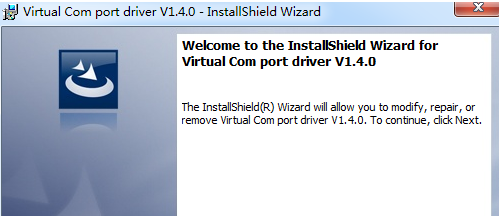
2、修改启动汇编代码startup\_stm32f103xb.s文件中直接修改

|  |
| --- |
| Stack\_Size EQU 0x400  Heap\_Size EQU 0x400 |

1. PC安装虚拟USB串口

<https://pan.baidu.com/s/1ecc3Qo24bqVoE66zDL6Gow>

密码：368w

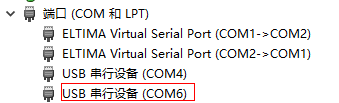


C:\Program Files (x86)\STMicroelectronics\Software\Virtual comport driver\Win8

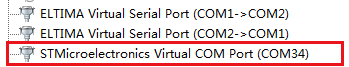
电脑为 64 位系统的用户，找到 dpinst\_amd64.exe；电脑为 32 位系统的用户，找到 dpinst\_x86.exe

提示安装成功，插入相关设备，我的电脑>>属性>>设备管理器里，在“端口（COM 和 LPT）”一栏，可以看到 COMx。至此，ST 虚拟串口驱动安装完毕。完成驱动安装后，请重启电脑。

我试验过程中没有安装，插好USB线后就识别一个COM6出来：



而安装虚拟USB串口在设备管理上识别出来的是：



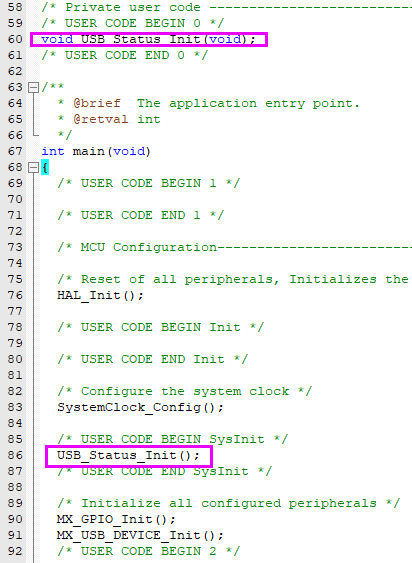
**USB无法识别的第三种情况：**

当每次下载完程序后，需要重新插拔USB线，电脑端才能识别到，反复这样的操作很烦人。其实解决办法也很简单，就是让USB的两个管脚都保持几个毫秒的低电平即可。

在main.c文件增加代码：

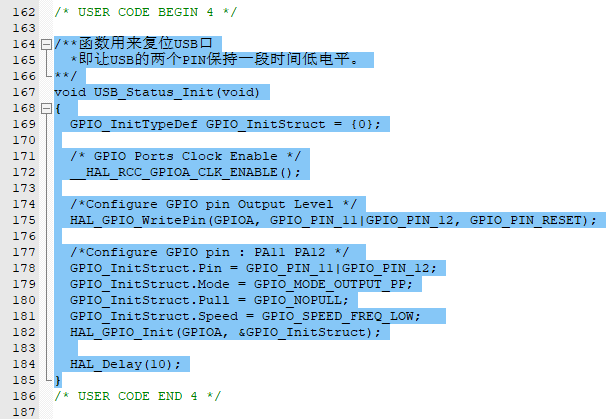
首先，声明和调用如下：

|  |
| --- |
| /\* USER CODE BEGIN 0 \*/  void USB\_Status\_Init(void);  /\* USER CODE END 0 \*/  /\* USER CODE BEGIN SysInit \*/  USB\_Status\_Init();  /\* USER CODE END SysInit \*/ |



其次，函数USB\_Status\_Init()实现：

|  |
| --- |
| /\*\*函数用来复位USB口  \*即让USB的两个PIN保持一段时间低电平。  \*\*/  void USB\_Status\_Init(void)  {  GPIO\_InitTypeDef GPIO\_InitStruct = {0};  /\* GPIO Ports Clock Enable \*/  \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  /\*Configure GPIO pin Output Level \*/  HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11|GPIO\_PIN\_12, GPIO\_PIN\_RESET);  /\*Configure GPIO pin : PA11 PA12 \*/  GPIO\_InitStruct.Pin = GPIO\_PIN\_11|GPIO\_PIN\_12;  GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;  GPIO\_InitStruct.Pull = GPIO\_NOPULL;  GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;  HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);    HAL\_Delay(10);  } |



编译，下载后测试OK