**《USART串口通信》实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**： **嵌入式系统设计** | **年级**：23级 | **上机实践成绩**： |
| **指导教师**：郭建 | **姓名**： **张建夫** |  |
| **上机实践名称**： **USART串口通信** | **学号：10235101477** | **上机实践日期**：2025/05/06 |
| **上机实践编号**： | **组号**： | **上机实践时间**： 14：50~16：30 |

1. **目的与要求**

简单了解串口通信

了解STM32的 USART外设

熟悉使用USART进行设备间通信的方法

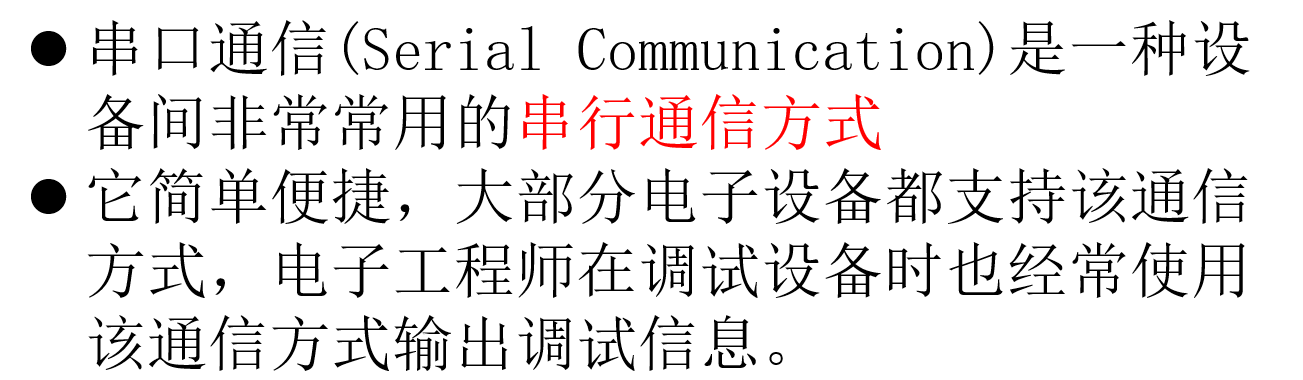
1. **内容与实验原理**
2. **实验内容：**

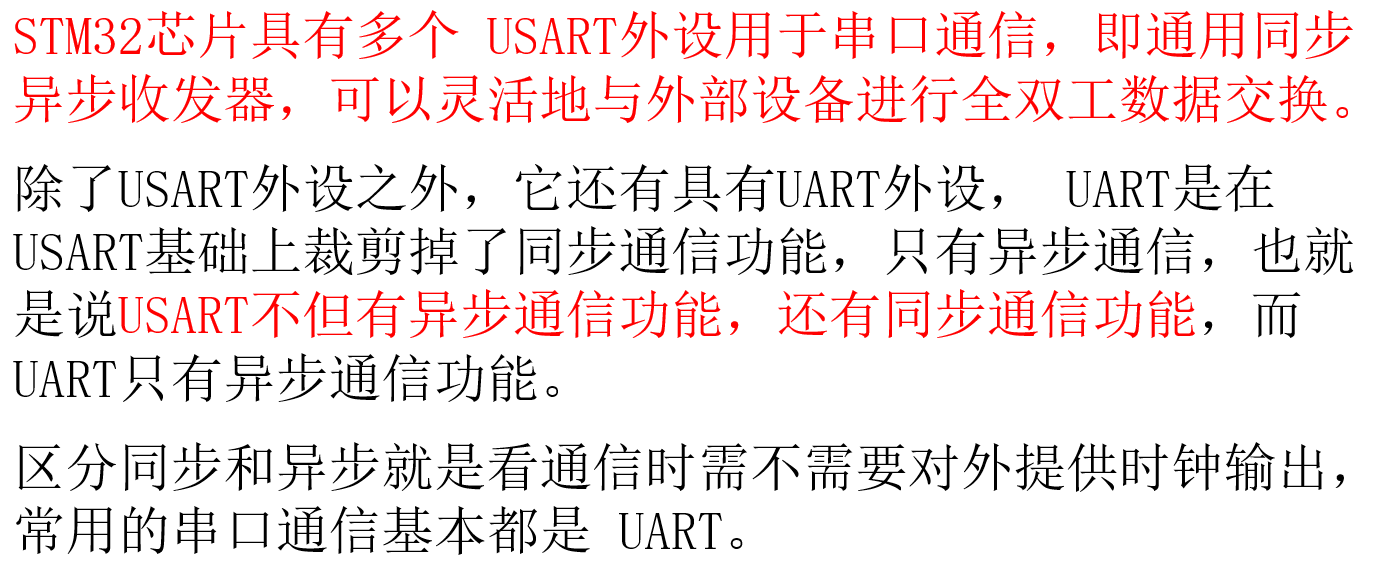
了解串口通信和STM32的USART外设

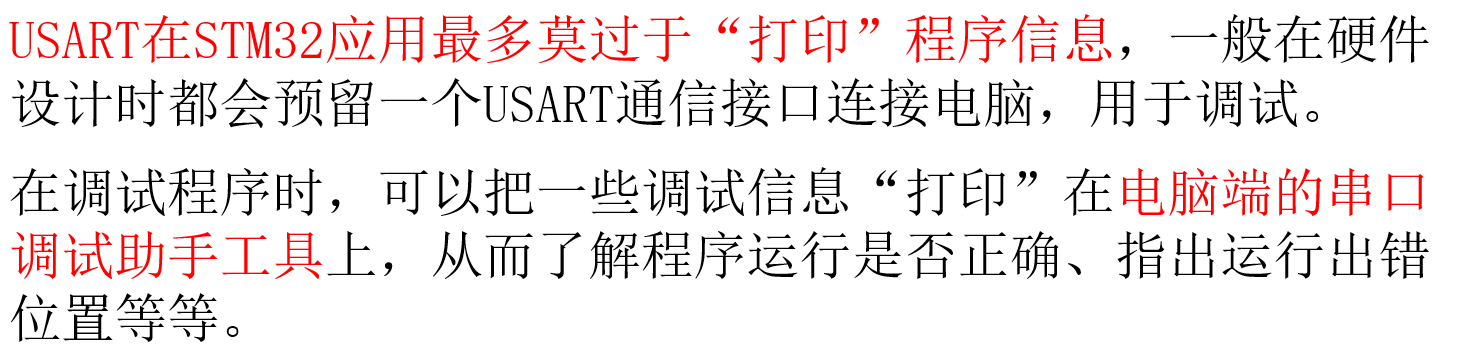
学习使用STM32的USART和电脑进行通信

**2.实验原理：**

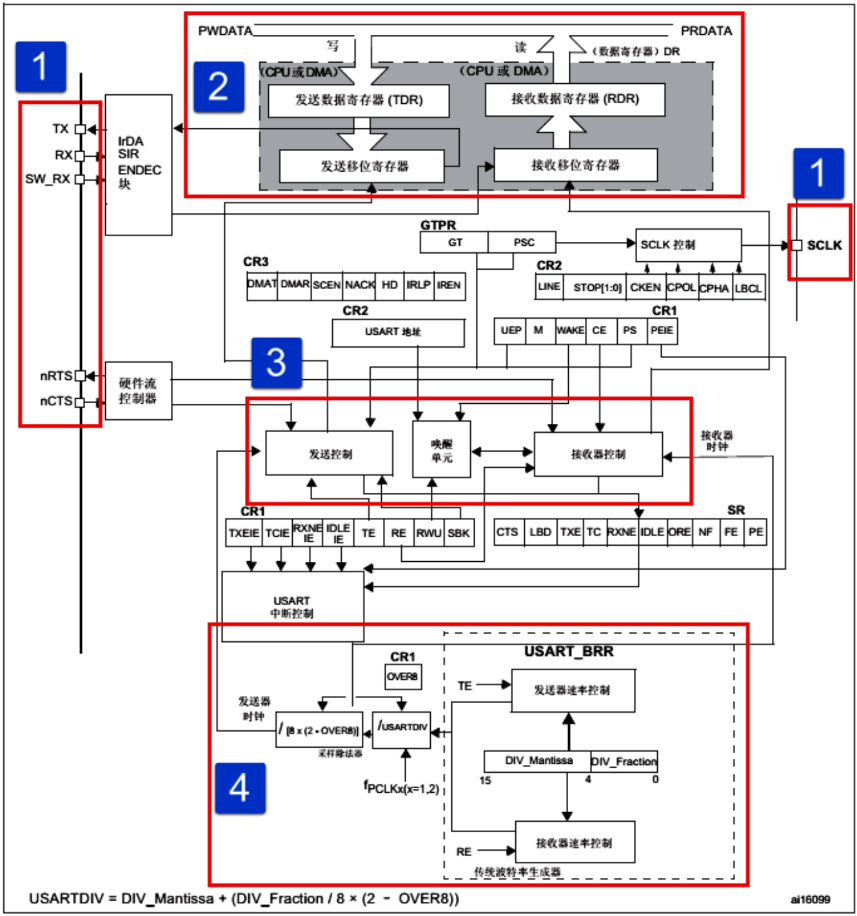
**（1）串口基本知识：**

****

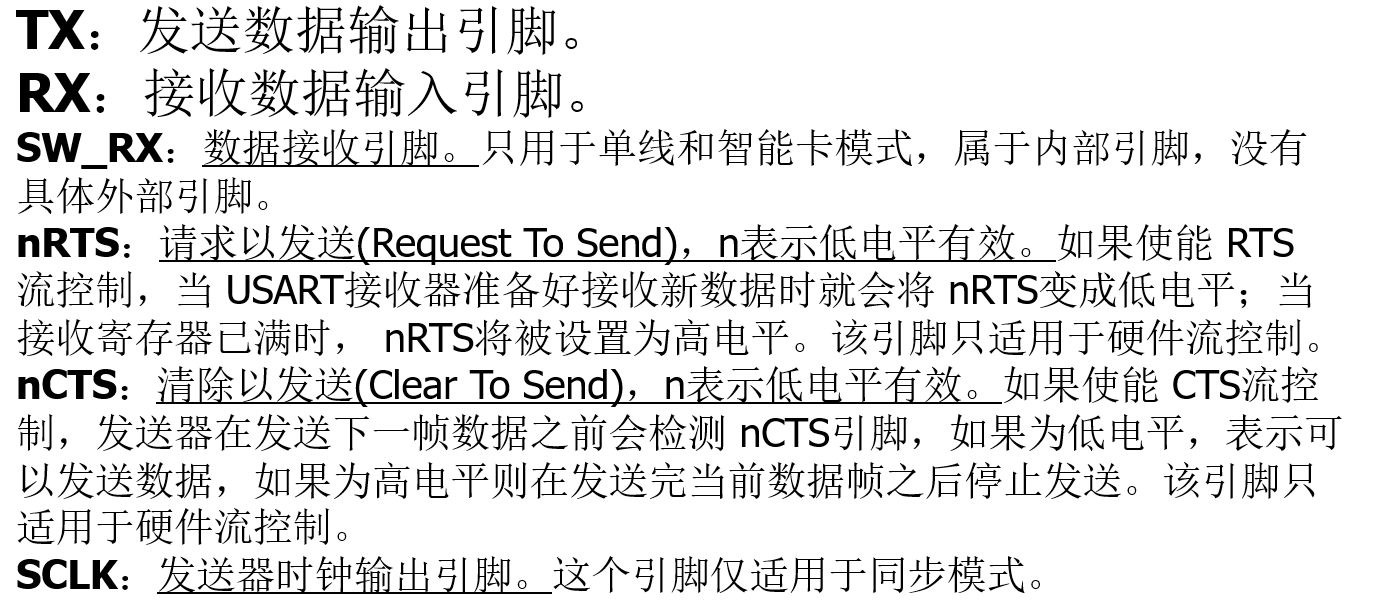
****

****

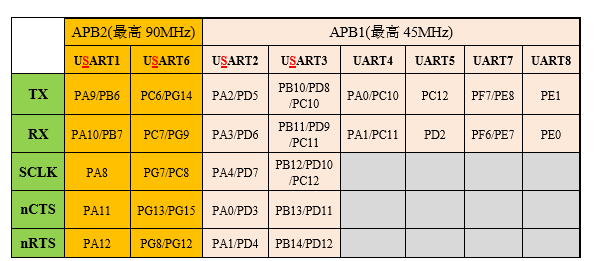
**（2）USART功能框图：**

****

**注意到左上角的五个引脚，本次实验只用到了接收和发送引脚（TX和RX）：**

****

**具体的引脚和USART的引脚对应关系如下图所示：**

****

由于ppt上的内容太多了，全解释的话篇幅过长而且实验报告的核心不在于复述ppt，

此处简介stm32F4系列的USART和UART：

1. STM32F42xxx系统控制器有四个 USART和四个 UART，其中 USART1和 USART6的时钟来源于APB2总线时钟，其最大频率为 90MHz，其他六个的时钟来源于 APB1总线时钟，其最大频率为 45MHz。

2. USART有同步功能，支持DMA（Direct Memory Access）传输，本次实验并不使用其同步功能。

3. USART数据寄存器(USART\_DR)的第九位数据是否有效取决于USART控制寄存器 1(USART\_CR1)的 M位设置，当 M位为 0时表示 8位数据字长，当M 位为 1表示 9位数据字长。

4. 使用 USART之前需要将 USART\_CR1寄存器的UE位置1使能 USART。

**（3）stm32F4中的串行通信流程：**

第一步：

当 USART\_CR1 寄存器的发送使能位 TE置 1时，启动数据发送，发送移位寄存器的数据会在 TX 引脚输出，一个字符帧发送需要三个部分：起始位+数据帧+停止位。

第二步（发送方）：

发送器开始会先发送一个空闲帧(一个数据帧长度的高电平)，接下来就可以往 USART\_DR寄存器写入要发送的数据。然后等待 USART状态寄存器(USART\_SR)的 TC 位为 1，表示数据传输完成，如果 此时USART\_CR1寄存器的 TCIE位置 1，将产生中断。

第三步（接收方）：

如果将 USART\_CR1 寄存器的 RE 位置 1，使能 USART接收，使得接收器在 RX 线开始搜索起始位。在确定到起始位后就根据 RX 线电平状态把数据存放在接收移位寄存器内。接收完成后就把接收移位寄存器数据移到 RDR内，并把 USART\_SR 寄存器的 RXNE位置 1，同时如果 USART\_CR2寄存器的 RXNEIE置 1的话可以产生中断。

另外一个知识点是USART的发送器和接收器使用相同的波特率，波特率等价于比特率，波特率越大，传输速率越快。

1. **使用环境**

调用dxdiag工具：

Operating System: Windows 11 家庭中文版 64-bit (10.0, Build 22621) (22621.ni\_release.220506-1250)

Language: Chinese (Simplified) (Regional Setting: Chinese (Simplified))

System Manufacturer: HP

System Model: HP Pavilion Aero Laptop 13-be2xxx

BIOS: F.13 (type: UEFI)

Processor: AMD Ryzen 5 7535U with Radeon Graphics (12 CPUs), ~2.9GHz

Memory: 16384MB RAM

Available OS Memory: 15574MB RAM

Page File: 27604MB used, 5685MB available

Windows Dir: C:\WINDOWS

DirectX Version: DirectX 12

DX Setup Parameters: Not found

User DPI Setting: 144 DPI (150 percent)

System DPI Setting: 192 DPI (200 percent)

DWM DPI Scaling: UnKnown

Miracast: Available, with HDCP

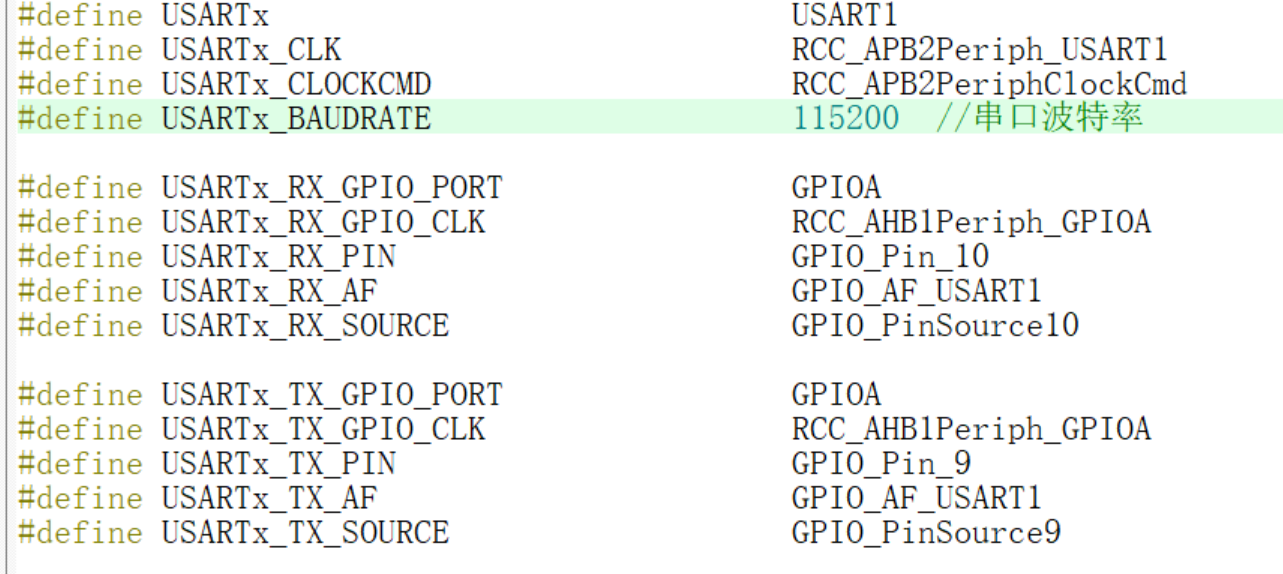
Microsoft Graphics Hybrid: Not Supported

1. **主要实验内容和结果展示**

本次实验使用的是之前提供的模板，需要将bsp\_key的头文件和源文件均变为bsp\_usart的文件内容。除此之外，本次实验较为简单（虽然中间因为一些头大的驱动问题和电脑的硬件问题卡了一段时间），于是想结合之前的led灯控制和中断控制做一些有意思的项目，而这个学期选了一门通识课讲到使用灯光表示摩斯码来传递信息，便做了一个小项目：用户通过串口输入一个字符串，板载led就将该字符串使用摩斯码表示出来（红色表示点，白色表示横线，灯灭表示当前字符结束）。

1. **示例实验：**

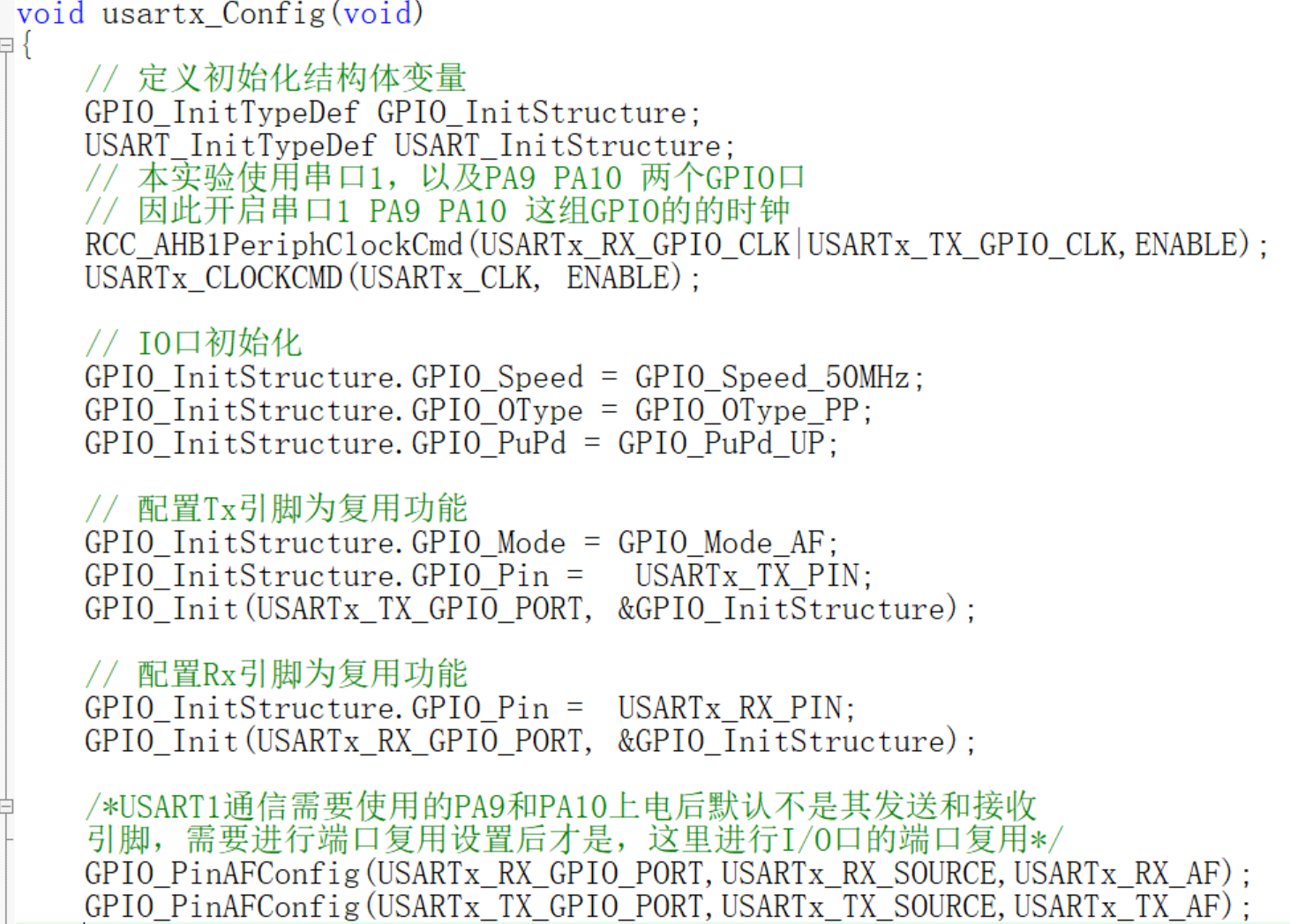
按照ppt上的教程，先要在头文件（bsp\_usart.h）对串口的变量进行映射（宏定义）：



本次使用的串口为USART1，其中根据前一部分的引脚对应关系表，USART1的发送引脚对应的GPIO引脚为9，端口为A， 波特率设为115200。

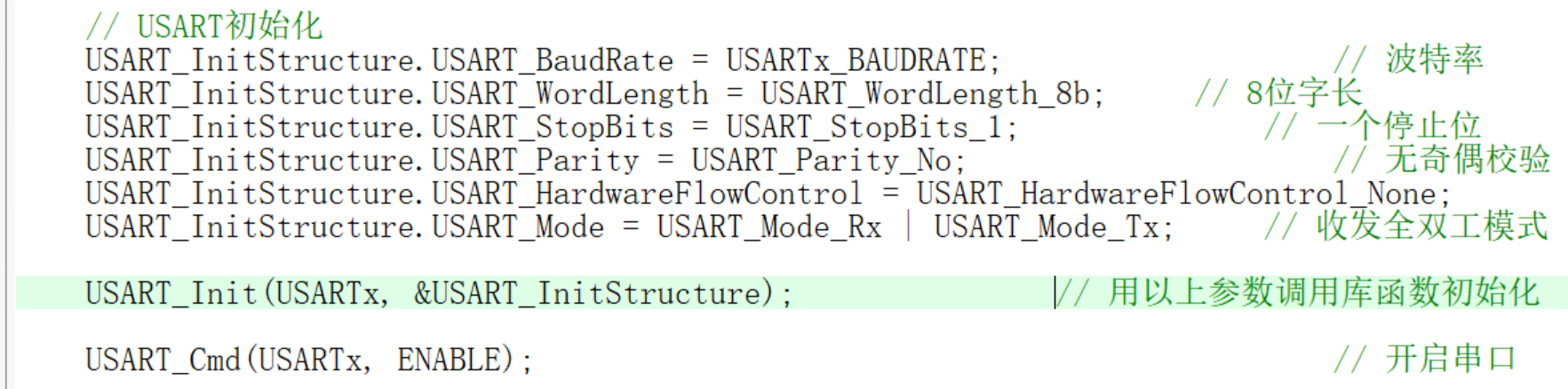
该处我在理解代码的时候产生了一个疑惑，可以从图上看出，串口使用的时钟是APB总线上的时钟，而接收端和发送端的接口时钟使用的是AHB总线上的时钟，这两者不会有冲突吗？后面我查阅了相关资料，发现TX/RX 的信号发出依靠 AHB1 上的 GPIO 来完成（二者挂载在AHB1总线上），而控制逻辑和数据流处理都在 USART 模块本身，二者并不冲突，在实际实现中，USART 是用 APB 时钟驱动 + 与 AHB 上的 GPIO 配合实现物理通信。

接着编写串口初始化函数：



此处GPIO初始化加上了USART初始化特色：分别为Tx和Rx引脚设置复用功能，并且要对这两个引脚进行端口复用（对应底部的两行代码）

接下来是USART的初始化：



此处首先设置了波特率为宏定义里的115200，设置了字长（数据位+校验位）为8，一个停止位，不使用奇偶校验（ppt里面的注释有问题，应该是无奇偶校验，而不是偶校验，偶校验的宏应为USART\_Parity\_Even），不使用硬件流控制，使用收发全双工模式（使能接收和发送），最后初始化USART并将其使能。

接下来，由于为了成功实现串口通信（为了键盘上的输入能够到达开发板，我们要“重定向”io流），需要“重写”C库函数fgetc和fputc，原因是C库里面的其他输入输出函数的实现都是调用这两个函数实现的，下面是“重写”代码：

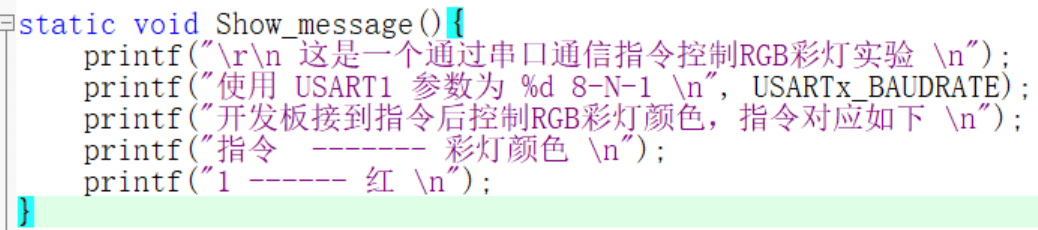


该处代码的实现采用了忙等的方式，不断等待串口输入/输出数据，直到TXE标志位（发送寄存器为空）或RXNE标志位（读数据寄存器非空）被设置为reset。

最后，就是main函数的实现：

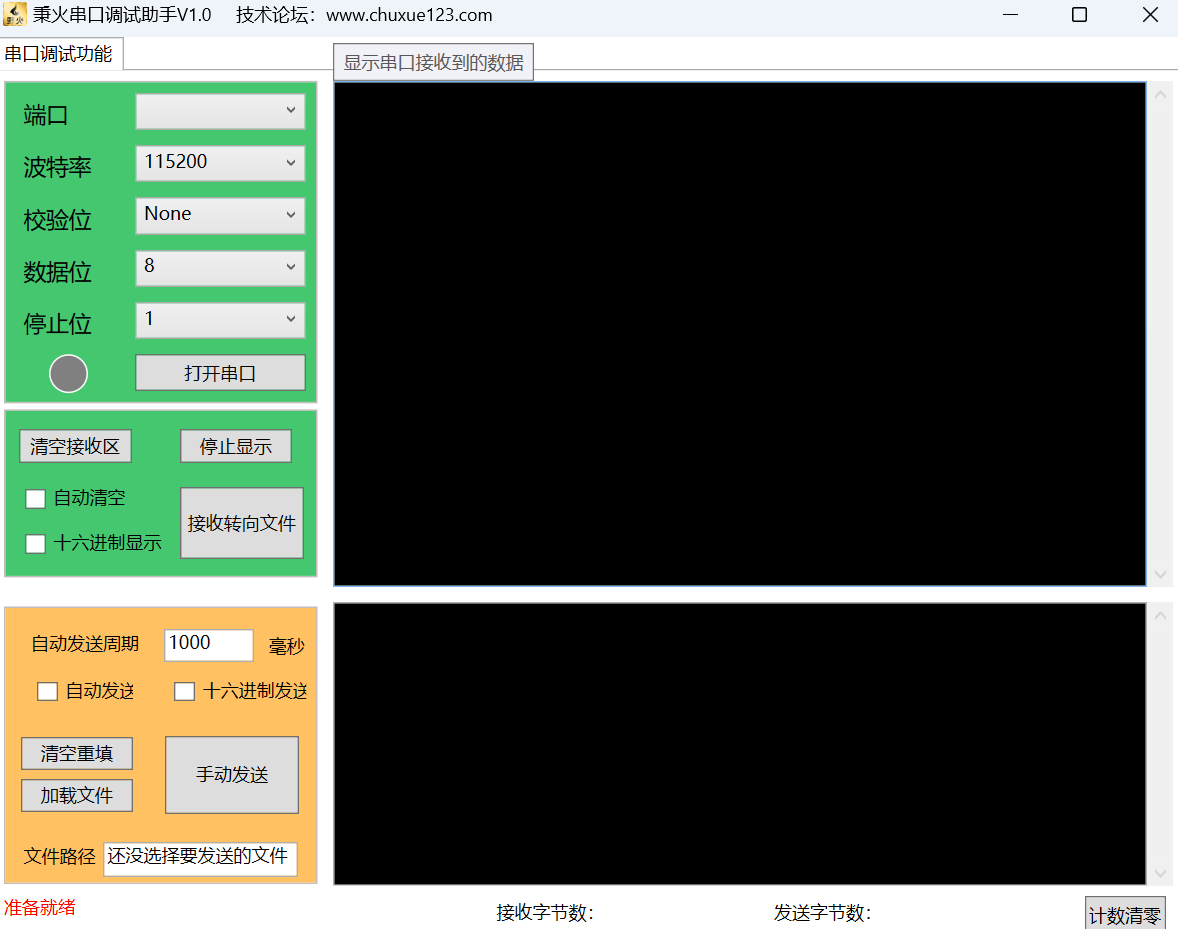


首先对所有变量初始化，然后在串口上打印提示信息，此处使用了一个show\_message函数将提示信息进行了封装，打印内容如下：



在main函数中，我做了一些小小的改变，相比于ppt上的代码：我在else语句中加入了LED\_RGBOFF，也就是说，如果用户输入的不是1，那么就会关闭红灯，这样就不会在输入1后红灯一直亮着了。

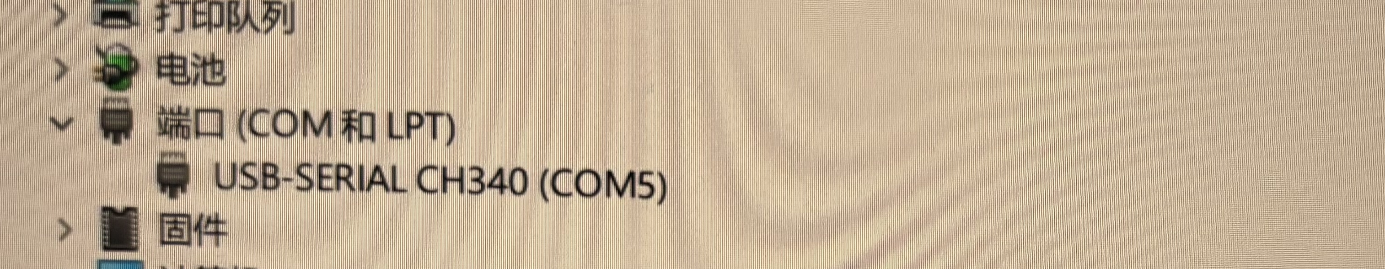
到了这里，软件的部分算是做完，只要ppt仔细阅读，代码都不难理解，但是，硬件，才是接下来使我比较伤脑筋的部分，在烧录完代码，按照指令连接usb，打开了老师提供的串口调试助手后，发现没有任何反应：



端口处无法进行识别，其他同学此处都是自动识别的，而我这个愣是没反应，在排除了软件问题后，决定换成学校电脑试一次，这次端口处有反映了，但是发送数据后没有任何的打印信息，板载的led也没有亮：



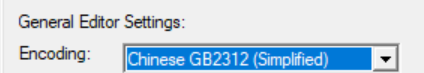
后面我又在自己电脑和学校电脑试了几次，打开设备管理器看了又看，发现当我插上usb时，我个人pc的设备管理器没有任何反应，学校电脑的也是（这里有个伏笔，后面解释，可以说明我用的那台学校电脑硬件也有点问题），下课后求助老师和助教，老师的诊断结果是串口没有成功接上，导致通信没有反应，但是原因不清楚，当时有一位同学（实名感谢洪宇翔同学，他让我立马找到了解决方案，还帮我解决了一个字符编码的小问题）说可以用他的电脑试一下，看是否是开发板的问题。在接上我的开发板后，他的串口上成功显示了我的打印信息，这一点让我确认了我的开发板和程序是没有问题的，问题只可能是我用的电脑，接下来，最振奋人心的时候到了，我让他打开他的设备管理器，看到了这个：



他的串口连接是有一个ch340的驱动的，而我插上去时并没有装这个驱动，随即我开始尝试这个办法。装完驱动后我成功了：



（这里屏幕上显示乱码是因为字符编码的问题，要将Edit菜单的configuration里编码格式改为如下即可：



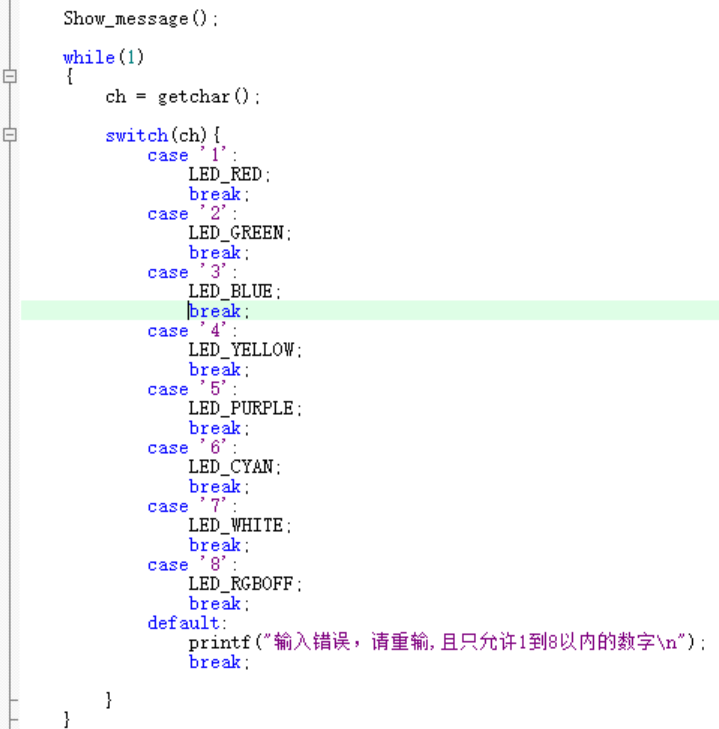
该问题也是由洪宇翔同学帮忙解决）

但是这个成功是暂时的，当时我的仿真器（就是用于烧录程序的那个）连的是学校电脑，用于串口通信的usb接口连的是我个人电脑右侧的usb接口（我电脑有左右两个usb接口），回去后，我用自己电脑又试了一遍，这次失败了，后面查看设备管理器，发现只有当usb接上我电脑右侧的接口的时候，设备管理器才会自动刷新，结合之前那个“伏笔”，我的推测就是，1.我的电脑左侧的usb和学校电脑的usb接口有问题。2.我的电脑要安装CH340驱动才能识别串口。

由于我的左侧usb有问题，演示视频中采用了拓展坞将两个usb接到电脑上。

1. **参考PPT， 使用USART1指令控制 RGB彩灯，输入数字1~8：** “**红**-**绿**-**蓝**-**黄**-**紫**-**青**-白–灭”**点亮相应的彩灯，其它字符错误。**

在解决硬件问题后，这个问题就显得异常简单了，只需要将示例实验里面的主函数的核心代码变为一个switch语句，对于符合要求的输入，展示相应的led灯，对于其他情况（default），输出错误信息，并让用户重新输入：



此处对应数字的颜色由题目决定。

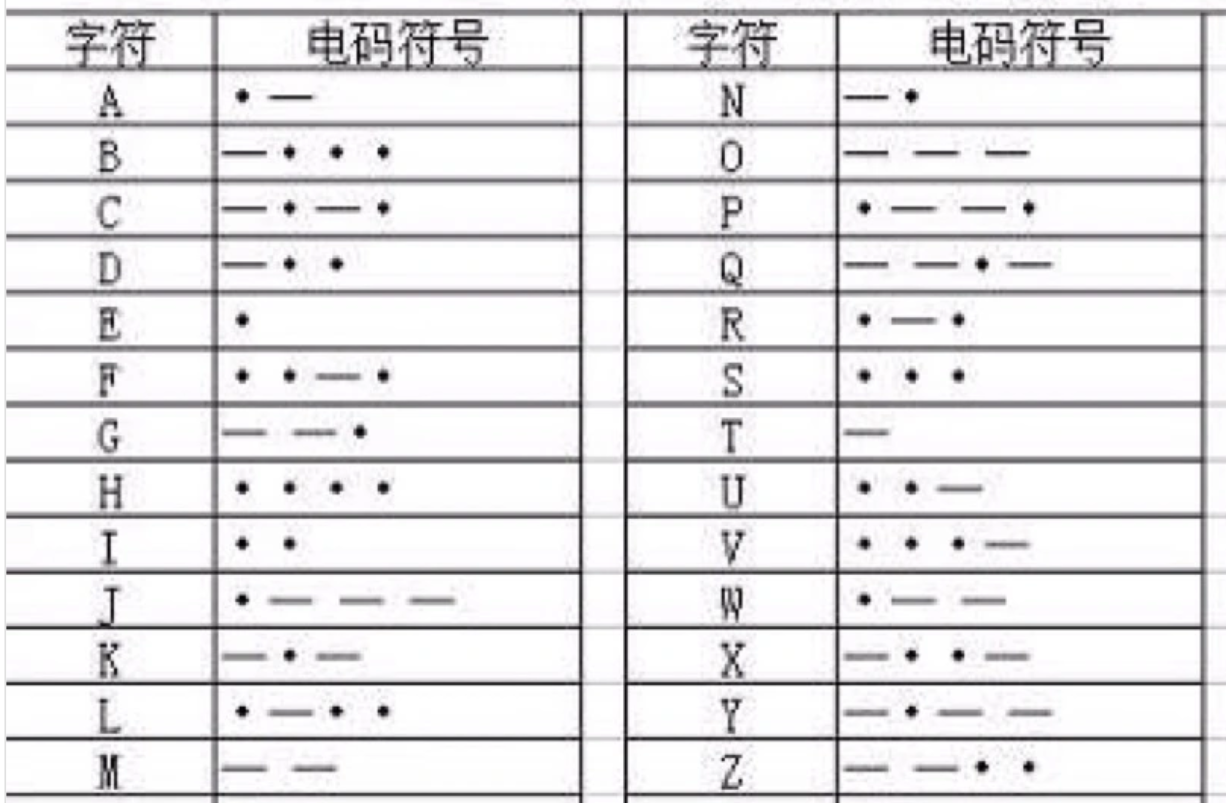
演示视频见提交文件。

1. 课外项目：使用led表示摩斯码，字符串由串口给出：

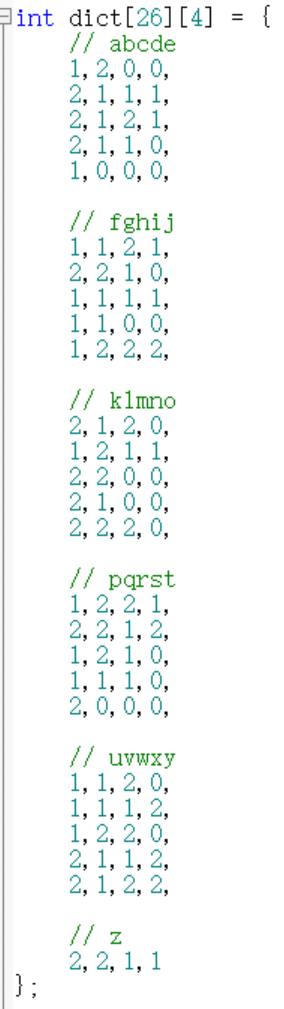
本项目的具体需求如下：

* 用户从串口输入一个仅包含英文字符（不论大小写）的字符串，发送后，板载led灯应该展现这个字符串的摩斯码（红色表示点，白色表示横线，持续时间为红色的三倍，灯灭表示当前字符结束）

为了使编码较为简单，我打算使用哈希表来存储当前字符的摩斯码，其中，英文字符的摩斯码如下：



可以从图中看出，每个英文字符由4个状态组成（一个状态包括点，横线，空），于是哈希表可以设置为一个二维数组（行表示字母a~b，映射为0~25，列则存储的是各自字母的摩斯码）：

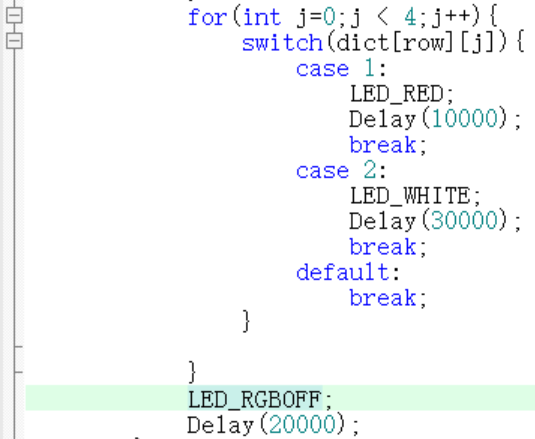


其中，数字1表示点，数字2表示横线，数字0表示空。

同时，由于是二维数组，我们能通过库函数tolower当前字符转化为其小写形式，再求得这个小写形式与ascii码中‘a’的值相减就能得到行数：



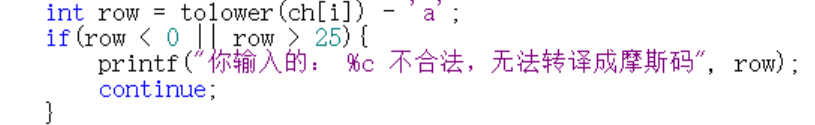
遍历这一行的摩斯码即可完成该字符的摩斯码展示：



（此处使用的Delay函数是第二个实验中模板的延迟函数，通过无效循环来延迟时间，此处不再贴代码，可以在提交的代码文件中查看。）

可以看到各个灯随摩斯码的变化状态，以及每次内循环结束后灯灭的长度，点，横线和灯灭的时间间隔之比为1：3：2.

最后，为了防止用户输入非法字符而导致程序崩溃，要判断一下映射出来的行数是否合法：

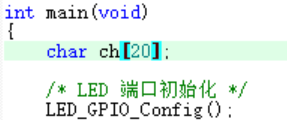


最后展现一下整个程序：

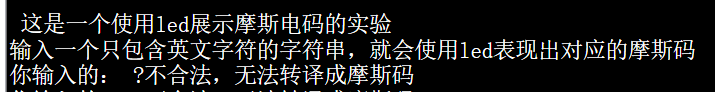


这个版本是初始程序，后面调试的时候发现了如下问题：

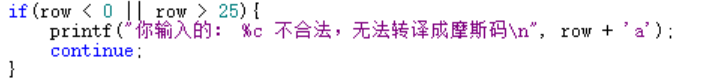
1. ch变量给的缓冲区太大（2k字节），导致将代码烧到板子上时板子没有任何反应，于是将ch的字节数变为20：



1. 在测试非法字符输入的时候，发现提示信息无法打印用户的字符：

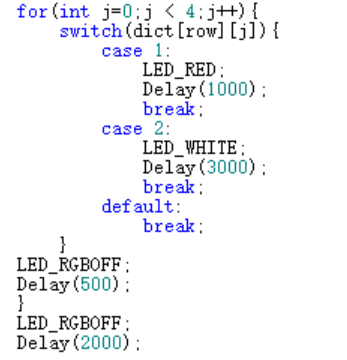


用户输入的字符会出现乱码，于是修改了提示信息的代码：



1. 调试时延时时间太长，摩斯码led不明显，且元字符（横线和点）中间无空格，导致看不出来摩斯码：

我将所有延时时间缩短了10倍，并在元字符之间加上了较短的间隔：



最终的程序见提交的代码文件。

演示视频见提交文件。

**五、实验总结**

本次实验我收获颇多，通过USART串口通信这一个功能，使开发板的潜在能力大大增强，尤其是其开始能够接收用户输入了，而不只是按个按键那么简单。此外，在本次实验中，我发现自己对硬件的认识仍然不够深入，还需要不断地学习，来积累一些硬件相关的解决办法，不然以后遇到了同类问题无法下手。本次实验也很好的激发了我对嵌入式开发探索的兴趣，后面有机会找一些有意思的嵌入式项目做一些实际的开发。