第二轮考核笔记

一：相关环境配置

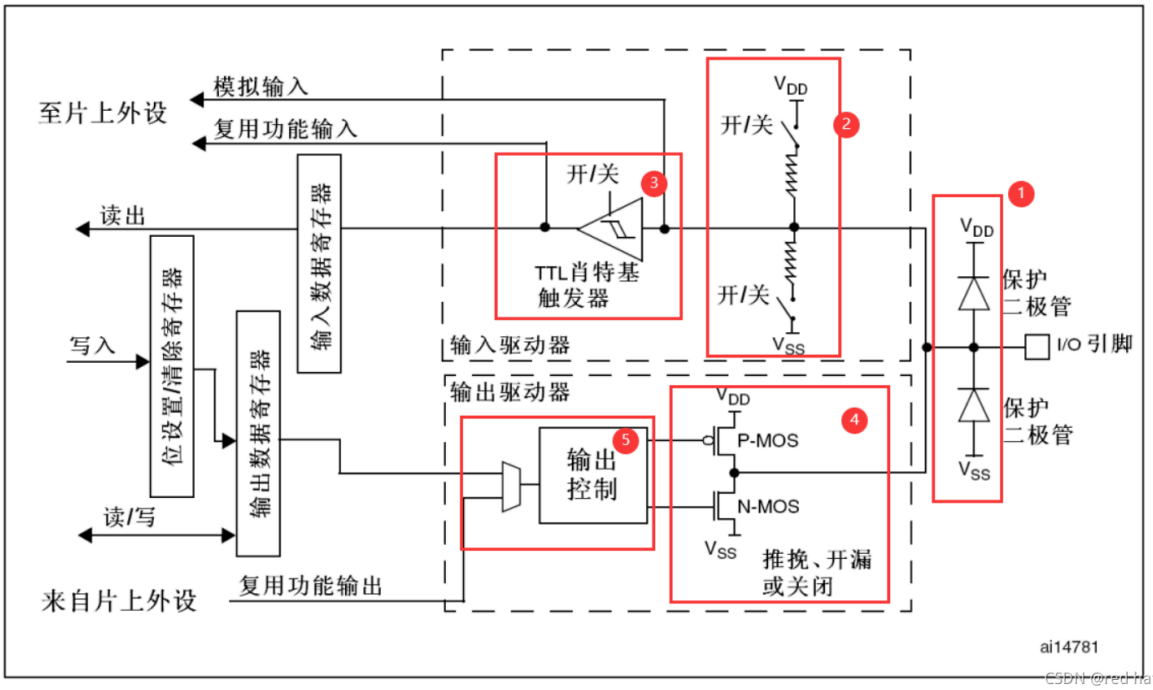
1：在mdk中打不开跑马灯的文件，原因：zip文件没有全部解压导致无法打开。

2：USB串口作用：可以当串口使用；

供电，如果开发板功率过大或者外设较多，可能出现供电不足的情况，可以再连接一个外设电源；

如果连接到stm32的串口1(stm32isp下载只能是串口1），可用来串口下载程序；

二：GPIO



1:不是所有引脚都是GPIO，有ft标识的引脚可容纳5.5v的电压，所有io口都可以作为中断输入；

2:STM32 芯片的 GPIO 被分成七组，每组有 16 个引脚;

3:GPIO有八种工作模式

- 通过配置GPIO的端口配置寄存器，端口可以配置成以下8种模式

模式名称	性质	特征
浮空输入	数字输入	可读取引脚电平，若引脚悬空，则电平不确定
上拉输入	数字输入	可读取引脚电平，内部连接上拉电阻，悬空时默认高电平
下拉输入	数字输入	可读取引脚电平，内部连接下拉电阻，悬空时默认低电平
模拟输入	模拟输入	GPIO无效，引脚直接接入内部ADC
开漏输出	数字输出	可输出引脚电平，高电平为高阻态，低电平接VSS
推挽输出	数字输出	可输出引脚电平，高电平接VDD，低电平接VSS
复用开漏输出	数字输出	由片上外设控制 高电平为高阻态，低电平接VSS
复用推挽输出	数字输出	由片上外设控制，高电平接VDD，低电平接VSS

引脚接连片上外设可以是

usart,i2c,spi...

输出模式下可控制端口的输出高低电平，可用于驱动LED、控制蜂鸣器、模拟通信协议输出时序（SPI I2C 等）等

输入模式下可读取端口的高低电平或电压，用于读取按键输入，外接模块电平信号输入，ADC电压采集、模拟通信协议接收数据（SPI I2C 等）等；

4：大部分引脚除了当GPIO使用外，还可以复用为外设功能引脚；

三：寄存器

1：端口模式寄存器：MODER寄存器每两位控制一个IO，32个位控制一组IO的16个IO；

2：端口输出类型寄存器：OTYPER寄存器每位控制一个IO，低十六位控制一组IO的十六个IO，高十六位保留没有用；

3：端口输出速度寄存器：OSPEEDR每两位控制一个IO，32个位控制一组IO的16个IO；低速（00）2MHZ，中速（01）25MHZ，快速（10）50MHZ，高速（11）100MHZ（30PF）（15PF时为80MHZ）；

4：端口上/下拉寄存器；PUPDR

5：端口输入数据寄存器：IDR

6：端口置位/复位寄存器：

7：端口输出数据寄存器

8：端口位复用功能寄存器

四：跑马灯实验

1：GPIO采用推挽输出

2：GPIO库函数：

一个初始化函数 GOPIO init

两个读取输入电平函数

两个输出电平函数

四个设置输出电平函数

3：**使用GPIO必须先使能相应的GPIO时钟**;

在使用GPIO之前要先初始化GPIO的时钟，因为许多外设都被设计时序逻辑电路，所以必须要为外设提供时钟源，否则外设的电路状态不会被改变；

4:#ifndef #define #endif格式条件编译，作用是避免头文件内容比重复定义；

5：调用任何函数都要引用头文件;

6:invalid警告，原因是这个函数在H文件里面没有声明或者头文件写错了；

1. GPIO初始化：在程序开始时，需要配置GPIO端口为输出模式。这通常涉及选择相应的端口和引脚，然后设置方向寄存器为输出。
2. 定义循环：跑马灯的效果需要通过一个无限循环来实现，确保灯光按特定顺序持续移动。循环的步长和方向（顺时针或逆时针）由程序员设定；

   3：设置时序：为了创建连续移动的效果，需要精确控制每个LED的亮灭时间。这可以通过延时函数实现，例如使用 `delay()` 或 `sleep()` 函数，以毫秒为单位设定间隔。

4：切换LED状态：在每个循环迭代中，需要改变特定LED的电平状态，使其亮起或熄灭。通过写入1（高电平）或0（低电平）到对应的GPIO端口，可以控制LED的开和关。 在实际操作中，LED灯通常并联连接，并通过电阻分压来保护它们免受过高电压的影响。每个LED连接到一个GPIO口，通过程序控制这些GPIO口的状态，从而实现跑马灯效果;

四：Git常用命令

git status：查看的修改的状态（暂存区、工作区）

git add:添加工作到暂存区；

git commit_m:提交暂存区内容到本地仓库；

git log:查看提交日志；