# 第二讲 源源不绝

俗气的马克思主义者认为：“世界是物质的，物质是运动的，运动是有规律的，规律是由事物内部的矛盾决定的”。这句话里有两点对于本讲是很重要的，一点是“物质是运动的”，一点是“运动是有规律的”。STM32 也是一样，他的运动源源不绝，且有着他特定的规律。

## 本讲学习目标：

1、有的放矢，力无虚发。

2、把握重点，审时度势。

## A：万法因缘生

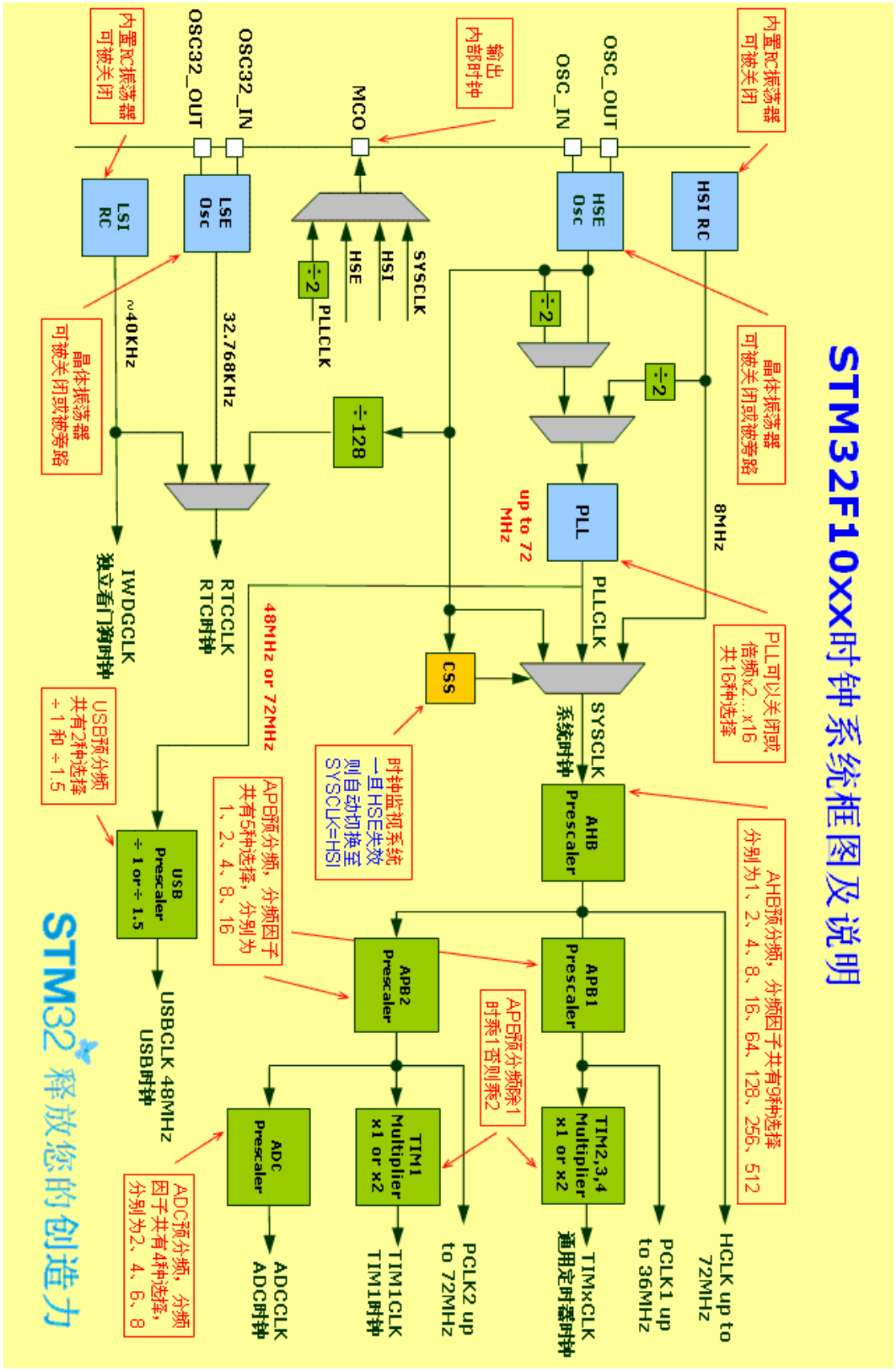
STM32 有着完备的时钟的体系，在上一讲中，列出了整套的硬件结构及库文件结构，今天将围绕着 RCC 复位与时钟控制，完成对 STM32 的初步调戏。

佛说：万法因缘生，万法因缘灭。我说：还个换成“源”好一点。开头说过，物质是运动的，运动是有规律的。这一规律在 uP 中非常容易被体现出来。“源”即时钟，STM32 的时钟无法想象的复杂，见下页中的图。

---

在这里说明下 BOOT 的选择(具体软件启动 startup.s 解释见讲义附件)：

| 启动模式选择引脚 | | 启动模式 | 说明 |
|---|---|---|---|
| BOOT1 | BOOT0 | | |
| X | 0 | 主闪存存储器 | 主闪存存储器被选为启动区域 |
| 0 | 1 | 系统存储器 | 系统存储器被选为启动区域 |
| 1 | 1 | 内置SRAM | 内置SRAM被选为启动区域 |


STM32F10XX时钟系统框图及说明
内置RC振荡器 可被关闭
输出 内部时钟
OSC_OUT
OSC_IN
OSC32_IN
OSC32_OUT
内置RC振荡器 可被关闭
MCO
HSI RC
HSE Osc
LSE Osc
LSI RC
SYSCLK
HSI
HSE
PLLCLK
÷2
÷128
÷2
÷2
晶体振荡器 可被关闭或被旁路
~40KHz
32.768KHz
晶体振荡器 可被关闭或被旁路
PLL
up to 72 MHz
8MHz
PLL可以关闭或 倍频x2...x16 共16种选择
PLLCLK
CSS
SYSCLK 系统时钟
AHB Prescaler
AHB预分频，分频因子共有9种选择 分别为1、2、4、8、16、64、128、256、512
HCLK up to 72MHz
IWDGCLK 独立看门狗时钟
RTCCLK RTC时钟
48MHz or 72MHz
时钟监视系统 一旦HSE失效 则自动切换至 SYSCLK=HSI
USB预分频 共有2种选择 ÷1 和 ÷1.5
USB Prescaler ÷1 or ÷1.5
APB预分频，分频因子 共有5种选择，分别为 1、2、4、8、16
APB2 Prescaler
APB1 Prescaler
APB预分频除1 时乘1否则乘2
USBCLK 48MHz USB时钟
TIM1 Multiplier x1 or x2
TIM2,3,4 Multiplier x1 or x2
PCLK2 up to 72MHz
PCLK1 up to 36MHz
TIM1CLK TIM1时钟
TIMxCLK 通用定时器时钟
ADC Prescaler
ADC预分频，分频 因子共有4种选择， 分别为2、4、6、8
ADCCLK ADC时钟
STM32 释放您的创造力

现在简单介绍下几个时钟：

HSE：外部晶体/陶瓷谐振器(一般选用 8M 手册中提到范围为 4~16Mz)

用户外部时钟 （最高可达 25MHz 从 SOC_IN 引脚输入，并保证 OSC_OUT 悬空）

HSI：此时钟信号由内部 8MHz 的 RC 振荡器产生，可直接作为系统时钟或在 2 分频后作为 PLL 输入

---

LSE：是一个 32.768kHz 的低速外部晶体或陶瓷谐振器。它为实时时钟或者其他定时功能提供一个低功耗且精确的时钟源

LSI RC：担当一个低功耗时钟源的角色，它可以在停机和待机模式下保持运行，为独立看门狗和自动唤醒单元提供时钟。LSI 时钟频率大约 40kHz(在 30kHz 和 60kHz 之间)。

**除此之外**，值得一提的是 SYSCLK/HSI/HSE/2 分频的 PLL 时钟，可由时钟配置寄存器(RCC_CFGR)中的 MCO[2:0]位控制外部 MCO 引脚输出。作为时钟同步信号。

---

知道了这几个时钟，我们来认识一下相关的几个寄存器：

| | |
|---|---|
| 时钟控制寄存器(RCC_CR) | 时钟配置寄存器(RCC_CFGR) |
| APB2 外设复位寄存器(RCC_APB2RSTR) | APB1 外设复位寄存器(RCC_APB1RSTR) |
| AHB 外设时钟使能寄存器(RCC_AHBENR) | APB2 外设时钟使能寄存器(RCC_APB2ENR) |
| APB1 外设时钟使能寄存器(RCC_APB1ENR) | AHB 外设时钟复位寄存器(RCC_AHBRSTR) |

部分对应库函数：

| 函数名 | rcc.c | 函数作用 |
|---|---|---|
| RCC_DeInit | 00216 | 将外设 RCC 寄存器重设为缺省值 |
| RCC_HSEConfig | 00269 | 设置外部高速晶振（HSE） |
| RCC_HSICmd | 00353 | 使能或者失能内部高速晶振（HSI） |
| RCC_PLLConfig | 00377 | 设置 PLL 时钟源及倍频系数 |
| RCC_PLLCmd | 00400 | 使能或者失能 PLL |
| RCC_SYSCLKConfig | 00563 | 设置系统时钟（SYSCLK） |
| RCC_HCLKConfig | 00607 | 设置 AHB 时钟（HCLK） |

| | | |
|---|---|---|
| RCC_PCLK1Config | 00633 | 设置低速 AHB 时钟（PCLK1） |
| RCC_PCLK2Config | 00659 | 设置高速 AHB 时钟（PCLK2） |
| RCC_LSEConfig | 00828 | 设置外部低速晶振（LSE） |
| RCC_LSICmd | 00861 | 使能或者失能内部低速晶振（LSI） |
| RCC_GetClocksFreq | 00907 | 返回不同片上时钟的频率 |
| RCC_AHBPeriphClockCmd | 01063 | 使能或者失能 AHB 外设时钟 |
| RCC_APB2PeriphClockCmd | 01094 | 使能或者失能 APB2 外设时钟 |
| RCC_APB1PeriphClockCmd | 01125 | 使能或者失能 APB1 外设时钟 |
| RCC_MCOConfig | 01281 | 选择在 MCO 管脚上输出的时钟源 |

以上的这些是会常用到的 RCC 库函数，当然有些函数是包括输入参量的，具体可以参看 rcc.h 中的一些宏。

---

笔者的话：

如果你想快速的配置好 RCC，则可使用一个比较偷工减料的办法，笔者一直使用。

试用一下这个函数“SystemInit();”他位于 system_stm32f10x.c 下 162 行。

注意：开放 system_stm32f10x.c 下 051 行设置为 72MHz 模式，以及仔细观察 108 行。

这个方式很管用吧？呵呵。

---

Cortex-M3 滴答时钟

Cortex-M3 的内核中包含一个 SysTick 时钟。SysTick 为一个 24 位递减计数器，SysTick 设定初值并使能后，每经过 1 个系统时钟周期，计数值就减 1。计数到 0 时，SysTick 计数器自动重装初值并继续计数，同时内部的 COUNTFLAG 标志会置位，触发中断(如果中断使能)。

在 STM32 的应用中，使用 Cortex-M3 内核的 SysTick 作为定时时钟，设定每一毫秒产生一次中断，在中断处理函数里对 N 减一，在 Delay(N)函数中循环检测 N 是否为 0，不为 0 则进行循环等待；若为 0 则关闭 SysTick 时钟，退出函数。延迟时间将不随系统时钟频率改变。

---

V3.4 版本库函数 滴答时钟举例 (注意：全局变量 TimingDelay 必须定义为 volatile)

```
void RCC_Configuration(void)
{
        SystemInit();

        RCC_GetClocksFreq(&RCC_ClockFreq);

        //SYSTICK分频--1ms的系统时钟中断

        if (SysTick_Config(SystemFrequency / 1000))
        {
                /* Capture error */
                while (1);
        }
}
```

```
volatile u16 Timer1;

void SysTickDelay(u16 dly_ms)
{
        Timer1=dly_ms;

        while(Timer1);
}


void SysTick_Handler(void)
{
        if(Timer1)Timer1--;
}
```

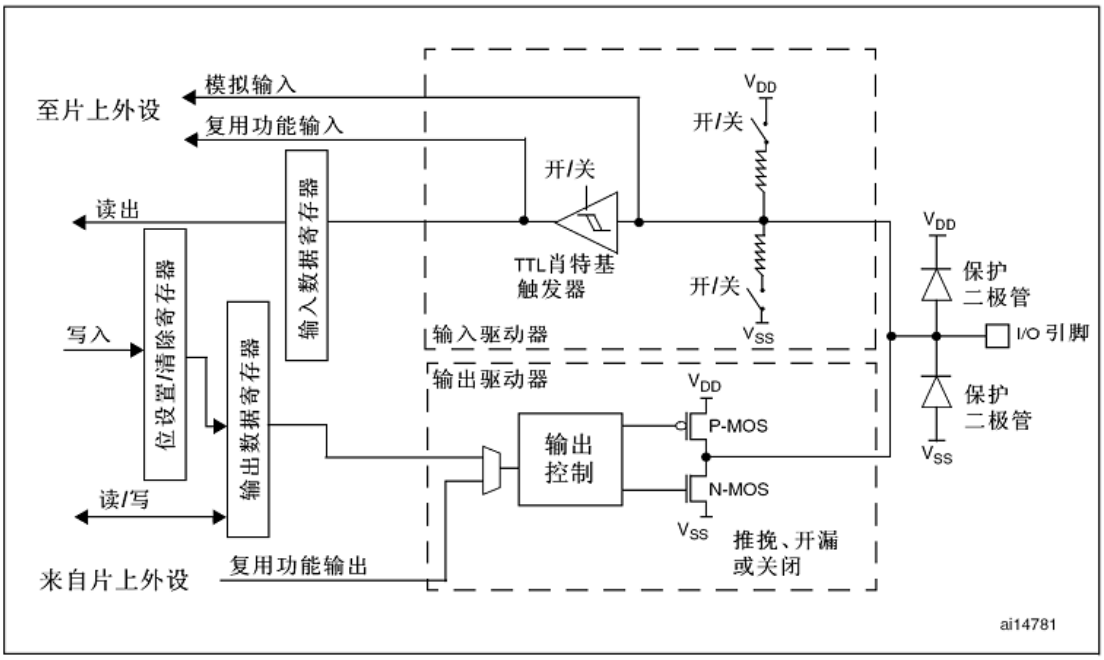
**应用举例：SysTickDelay(500);**

---

## B：矢---> GPIO

所有的微控制器都有输入输出端口，简称 IO，而 GPIO 是 General Purpose Input Output (通用输入/输出)的简称，所谓有的放矢，找到控制目标进行控制，最基本的控制除了复位时钟控制器 RCC 之外，GPIO 是最直观的外部控制，其中 STM32 的 GPIO，有功能配置及速度配置位，在 STM32 中一般有以下几种模式，请注意下以下结构参考图与库函数结构体中定义（GPIO.H 第 70 行）：

```
typedef enum
{ GPIO_Mode_AIN = 0x0,            //模拟输入
  GPIO_Mode_IN_FLOATING = 0x04,   //浮空输入
  GPIO_Mode_IPD = 0x28,           //下拉输入
  GPIO_Mode_IPU = 0x48,           //上拉输入
  GPIO_Mode_Out_OD = 0x14,        //开漏输出
  GPIO_Mode_Out_PP = 0x10,        //推挽输出
  GPIO_Mode_AF_OD = 0x1C,         //开漏复用功能
  GPIO_Mode_AF_PP = 0x18          //推挽复用功能
}GPIOMode_TypeDef;
```




| 结构体单元名 | 中文释义 | 实际结构与特点 |
|---|---|---|
| AIN | 模拟输入 | 实际控制中，关闭了上端的施密特触发器(自己的翻译，图中 TTL 肖特基触发器恕我知识浅薄看不懂)，以及输出驱动器即模拟信号仅能通过管脚流入片内模拟输入端。(有可能有下拉电阻，起到阻抗匹配作用)。 |
| IN_FLOATING | 浮空输入 | 理解起来最简单，即什么都不接，手触摸的时候人体感应引脚，可能高可能低，这个就不好说啦，STM32 结构不明，更何况人体是个大电容啊。一般不用。 |
| IPD | 下拉输入 | D 即 DOWN，下拉，笔者认为起到衰减及稳定信号作用，如有雷同理解，不胜荣幸。 |
| IPU | 上拉输入 | U 即 UP，上拉，此时在输入端，上拉电阻开关使能，芯片或接口的内部或外部有上拉,保证输入不接信号时为 1。 |
| Out_OD | 开漏输出 | 相当于三极管集电极，要得到输出高电平需上拉电阻，若输出低电平则吸电流能力强，可参看达林顿管 UNL2003，思考该芯片可否输出电平 1。 |

| | | |
|---|---|---|
| Out_PP | 推挽输出 | 笔者钻研开关电源，（数字电路很没劲啦，一下一下就搞出来了的，开关电源环节多，健脑 O(∩_∩)O~~）习惯称为图腾柱输出，Totem Pole，或半桥，两个三极管或 FET 受互补信号控制，使得在一个导通时另一个总是截止。一般用来用来匹配电压，或者提高 IO 口的驱动能力。 在 STM32 中当然是为了提高其驱动能力而设定的。 |
| AF_OD | 开漏复用 | AF 是复用的标号，是对于 STM32 的 GPIO 管脚的第二功能而言的，具体用到再说。比如使用串口就是要设为复用哦。具体配置可以参看参考手册 V10：8.1.11 外设的 GPIO 配置。 |
| AF_PP | 推挽复用 | |

| 配置模式 | | CNF1 | CNF0 | MODE1/0 | PxODR寄存器 |
|---|---|---|---|---|---|
| 通用输出 | 推挽(Push-Pull) | 0 | 0 | 01<br>10<br>11<br>见表18 | 0 或 1 |
| | 开漏(Open-Drain) | | 1 | | 0 或 1 |
| 复用功能输出 | 推挽(Push-Pull) | 1 | 0 | | 不使用 |
| | 开漏(Open-Drain) | | 1 | | 不使用 |
| 输入 | 模拟输入 | 0 | 0 | 00 | 不使用 |
| | 浮空输入 | | 1 | | 不使用 |
| | 下拉输入 | 1 | 0 | | 0 |
| | 上拉输入 | | | | 1 |

| MODE[1:0] | 意义 | 端口配置寄存器 |
|---|---|---|
| 00 | 保留 | GPIOx_CRL<br>GPIOx_CRH |
| 01 | 最大输出速度为10MHz | |
| 10 | 最大输出速度为2MHz | |
| 11 | 最大输出速度为50MHz | |

```
void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_3|GPIO_Pin_4;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

---

**补充知识：**

在一般集成电路中(如74系列)TTL 悬空等效于高电平，因为 TTL 型逻辑电路的基本构成是晶体三极管电路，逻辑0电平就是将输入端接到信号地，逻辑1电平+5V，为了保证电路可靠工作，设计时就对输入端加了正偏置，所以 TTL 与非门电路输入端悬空时可看做高电平输入。

CMOS 不允许悬空。原因是 CMOS 电路 CMOS 电路悬空可能出现不正常情况，因为悬空会使电位不定，破坏正常的逻辑关系，另外，悬空时输入阻抗高，易受外界噪声干扰，使电路产生误动作，而且也极易使栅极感应静电造成击穿。

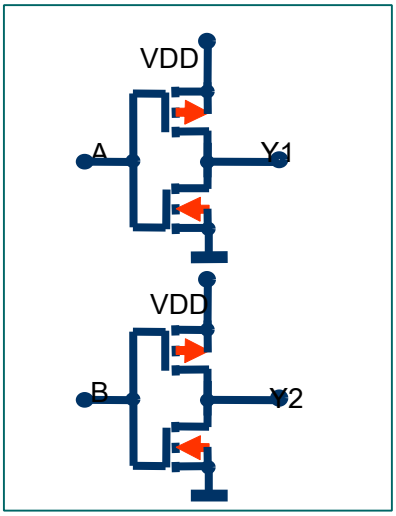
CMOS 的输入端是 FET 的栅极，MOSFET 输入阻抗极高，氧化层很薄，若输入端存在漏电流，就会产生极高的压降，致使 SiO2层被击穿，因此一般都加保护电路。极易被击穿造成永久性的损坏，所以在使用时要注意防止静电打击或其他瞬变电压引起的过压击穿。

1、CMOS 电路的电源电压，切记不能把极性接反，否则会使保护二极管因过流而损坏。

2、电路输出端既不能和电源短接，也不能和地短接，否则输出级的 MOS 管就会因过流而损坏。

3、 除了 OD 门和三态门之外，不同输出端不能并联起来使用，否则容易造成输出级 MOS 管因过流而损坏。

附例图：




假设：A（CMOS）器件导通，B（CMOS）器件截止，Y1/Y2 关系线与：则 A 器件下臂，B 器件上臂容易烧毁。

形成的低阻通路会产生很大的电流，极有可能导致器件的损毁，并且无法确定输出电平的高、低因此无法形成有用的线与逻辑关系。

解决方法：使用 OD 门

这不是数电课：要看请课后自行翻看数字电路基础。

网上资料： TTL 电平与 CMOS 电平的区别

(一)TTL 高电平3.6~5V，低电平0V~2.4V

CMOS 电平 Vcc 可达到12V

CMOS 电路输出高电平约为0.9Vcc，而输出低电平约为0.1Vcc。

CMOS 电路不使用的输入端不能悬空，会造成逻辑混乱。

TTL 电路不使用的输入端悬空为高电平

另外，CMOS 集成电路电源电压可以在较大范围内变化，因而对电源的要求不像 TTL 集成电路那样严格。 用 TTL 电平他们就可以兼容。

(二)TTL 电平是5V，CMOS 电平一般是12V。

因为 TTL 电路电源电压是5V，CMOS 电路电源电压一般是12V。

5V 的电平不能触发 CMOS 电路，12V 的电平会损坏 TTL 电路，因此不能互相兼容匹配。

(三)TTL 电平标准：

输出 L： <0.8V ； H：>2.4V。

输入 L： <1.2V ； H：>2.0V

TTL 器件输出低电平要小于0.8V，高电平要大于2.4V。输入，低于1.2V 就认为是0，高于2.0就认为是1。

(四)CMOS 电平标准： 笔者补充：俗称**1/2**电平翻转

输出 L： <0.1*Vcc ； H：>0.9*Vcc。

输入 L： <0.3*Vcc ； H：>0.7*Vcc.

---

## C：课后练习

第二讲 源源不绝，旨在把握 RCC 的同时，使读者学会使用库函数配置 STM32，将 STM32 的时钟系统深层次的剖析，并能在熟练掌握时钟配置的同时，对 GPIO 进行简单初始化，希望有所帮助。课后练习是根据自身理解，在上次讲座结束后自己建立的工程模板中，设定系统滴答时钟作为延时，使各自开发板上灯被系统滴答时钟控制，有效的亮灭。

D：听课笔记