**单片机（STM32）**

**InnoTechTens电路组**

置顶提醒：关于可能涉及的MOSFET及基础的电路知识，请参考《硬件培训之降压板》中的相关内容。并且，使用单片机需要使用C语言。考虑到部分成员无数电模电基础，本文档主要提供相关知识以配合B站上的STM32教程，帮助大家理解。关于B站教程，可以参考江协科技

(https://www.bilibili.com/video/BV1th411z7sn/?spm\_id\_from=333.337.search-card.all.click&vd\_source=d6f60c0881ac9f5de2c1c470e6b377c7)

或者正点原子

(<https://www.bilibili.com/video/BV1bv4y1R7dp/?spm_id_from=333.337.search-card.all.click&vd_source=d6f60c0881ac9f5de2c1c470e6b377c7>)

对于STM32F4，建议看这一个教程（[第2讲 开发板入门\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1Rx411R75t?spm_id_from=333.788.videopod.episodes&vd_source=d6f60c0881ac9f5de2c1c470e6b377c7&p=2)）

个人建议零基础的同学从前者开始学习，有一定开发经验的可以选择后者。

**基础**

**这一部分是为了在大家学习单片机之前普及一些必要或常见的知识和名词，以免在学习的过程中因为知识盲区而耽误学习进程。部分名词可能在表述上不甚专业，但出于方便大家入门理解的目的，这是一种折中的办法。不过仅仅对于单片机的学习而言已经足够。**

**关于二进制**

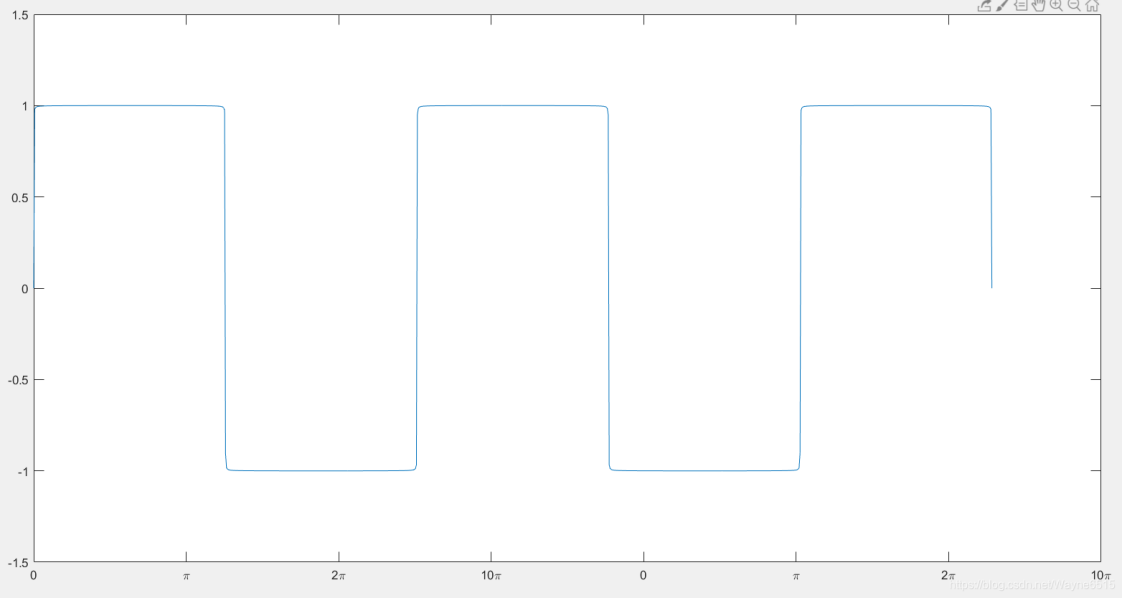
在单片机中，和计算机类似，所有的程序都是经过内部操作系统的转换最终变成不同的二进制数据来对硬件进行操作。使用二进制的原因，在硬件层面来看，可以这样理解：二进制数的每一位上只可能有0和1中的一者。而这对应到硬件电路中其实可以对应为两种电平状态：高电平（常由1对应）和低电平（常由0对应）。当一个电路端口输入/输出高电平时我们在程序中可以用1来表征，相应的低电平则用0表征。若此处无法明显的感知到这种对应关系的用处，在下面我在介绍非门时会用它举例来加强大家的理解。

那么为什么不使用其他进制，这和以下几点有关

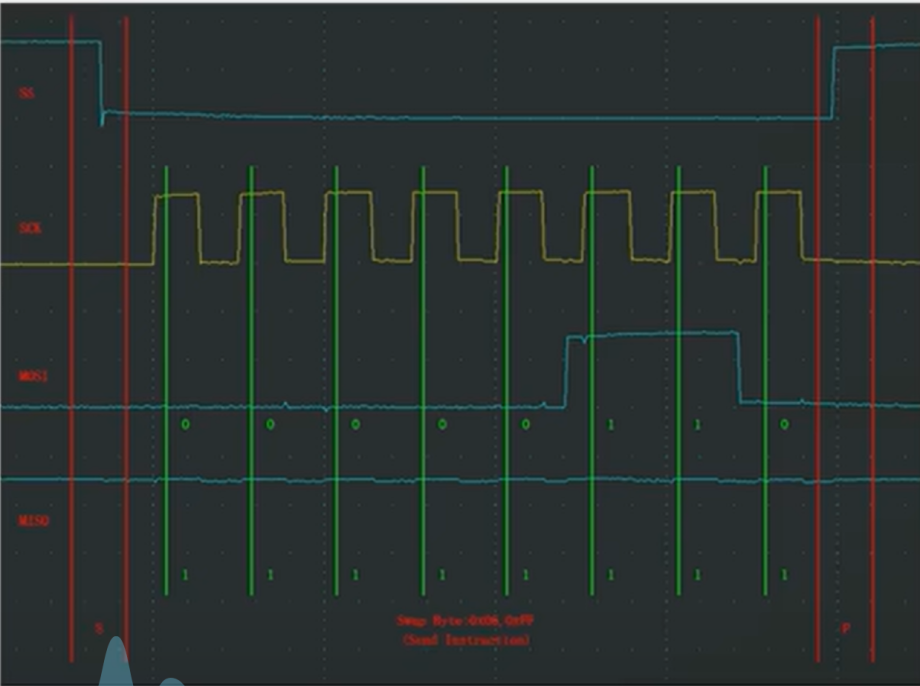
1. **电子电路的简单性**：计算机硬件，特别是早期的电子电路，是基于开关的，它们只有两种状态：开（1）和关（0）。这种二元性使得二进制成为表示和处理数据的自然选择。
2. **逻辑运算的便利性**：二进制非常适合逻辑运算，如AND、OR和NOT，这些运算是构建更复杂计算操作的基础。在二进制中，这些运算可以直接映射到电子电路的物理状态。
3. **错误检测和纠正**：二进制系统便于实现错误检测和纠正机制，如奇偶校验和汉明码等，这些机制对于确保数据传输和存储的可靠性至关重要。
4. **存储和传输效率**：二进制与电子设备的物理特性相匹配，使得数据存储和传输更加高效。例如，磁盘和内存都是以二进制形式存储数据的。
5. **历史和兼容性**：计算机科学的发展历史中，二进制已经成为标准。改变进制将需要重新设计和制造大量的硬件和软件，这在经济和技术上都是不切实际的。
6. **易于实现**：二进制的实现相对简单，因为它只需要区分两种状态，这在电子和机械系统中都很容易实现。

**关于信号**

在STM32或其他信号的单片机中，几乎所有我们能接触到的信号都是方波信号或不同的数据传输信号。方波信号如下图所示，是一种占空比为50%（即高电平和低电平持续时间均为信号周期的一半）的信号，且高电平固定在某一个值。



STM32中数据传输信号和方波信号在形状上都是方形，但区别在于数据传输信号高低电平的持续时间是根据数据而不断变化的。



上图是STM32进行SPI通信时的一段波形图。其中黄色波形便是进行传输时为了保证发送端发送数据和接收端读取数据的一致性而产生的时钟信号，可看到这是一段方波信号。方波信号在数字电路中是最常见的时钟信号。它的低电平持续时间段、高电平持续时间段、上升沿（信号由低电平向高电平转换的时刻）、下降沿（信号由高电平向低电平转换的时刻）都有可能发生不同的电路操作。但一般是以上升沿和下降沿作为操作的“行动时机”。

而第三层和第四层的信号便是数据传输信号。可以看到第三层信号在时钟信号的前五个周期都是低电平，这代表在这段时间内传输的五位数据都是0（一般是一个时钟周期进行一次数据传输）。而到时钟信号的第六个上升沿处，第三层信号变为了高电平。高电平持续了两个时钟信号周期，代表在这段时间内传输的是两位数据1。在最后一个时间周期，第三层信号又变为了低电平，代表最后一个周期内传输的是一位数据0。由此，便传输了一个字节的数据00000110（如果是先传输高位再传输低位，即高位先行，MSB）或者01100000（如果是先传输低位再传输高位，即低位先行，LSB）

**寄存器**

寄存器是CPU内部用来存放数据的一些小型存储区域，用来暂时存放参与运算的数据和运算结果。它是由具有存储功能的触发器组合起来构成的。一个触发器可以存储1位二进制代码，故存放n位二进制代码的寄存器，需用n个触发器来构成。。寄存器是中央处理器内的组成部分。寄存器是有限存储容量的高速存储部件，它们可用来**暂存指令、数据和位址**。

寄存器最起码具备以下4种功能。

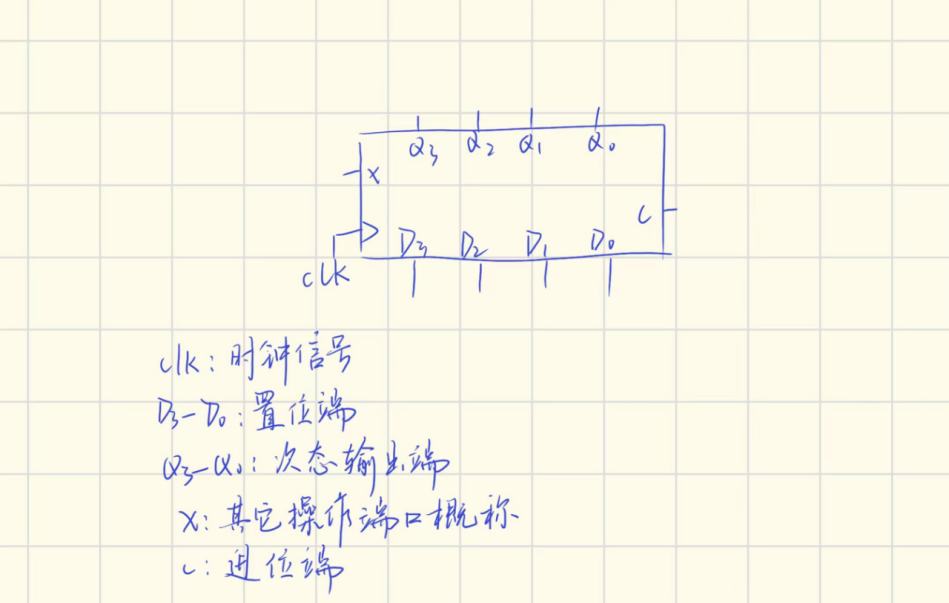
①清除数码：将寄存器里的原有数码（数据）清除。

②接收数码：在接收脉冲信号作用下，将外输入数码存入寄存器中。

③存储数码：在没有新的写入脉冲信号来之前，寄存器能保存原有数码不变。

④输出数码：在输出脉冲信号作用下，才通过电路输出数码。

实际上，寄存器往往是软件和硬件之间连接的桥梁。软件程序的操作转换为二进制数据暂存到寄存器中，再由寄存器的各个输出端口去对对应的不同电路进行相应的电平操作。



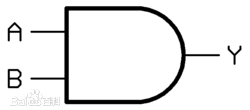
这是一个寄存器的概要图。其中的次态输出端可对应到我们所说的对硬件操作的端口。置位端理解为我们通过软件写数据到寄存器的端口。时钟信号用于保持电路运作的同步性等。举个例子，我们通过D3-D1端口向寄存器写入数据1001，那么寄存器就会在Q3-Q1端口分别输出数据1001，对应输出高电平、低电平、低电平、高电平，从而**对对应的电路端口输入了不同的电位**。这些电位可能使能（可以理解为激活）某些电路或失能某些电路。

**逻辑门**

看作一种黑盒子电路即可。常见的有或门、非门、与门、与非门、或非门、异或门、同或门

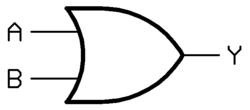
1. 与门

只有当所有输入都为高电平（对应1）时，输出才为高电平（对应1）。



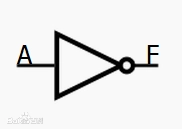
1. 或门

只要输入中有一个为高电平时（对应“1”），输出就为高电平（对应“1”）；只有当所有的输入全为低电平（对应“0”）时，输出才为低电平（对应“0”）。

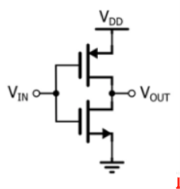


1. 非门

当输入端为高电平（对应“1”）时，输出端为低电平（对应“0”）；反之，当输入端为低电平（对应“0”）时，输出端则为高电平（对应“1”）

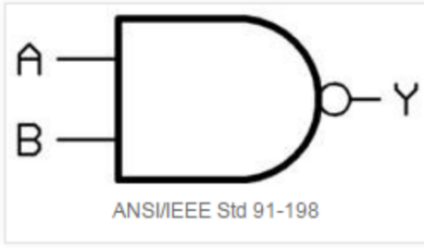


关于非门，目前最常见的是下图的电路。在该电路中，输入低电平（对应0）时，上面的PMOS导通而下端的NMOS截止，导致输出电压和VDD（即偏置电压）几乎相同，即输出了高电平（对应1）；输入高电平（对应1）时，上面的PMOS截止而下端的NMOS导通，导致输出电压和地相同，即输出了低电平（对应0），从而实现非门作用



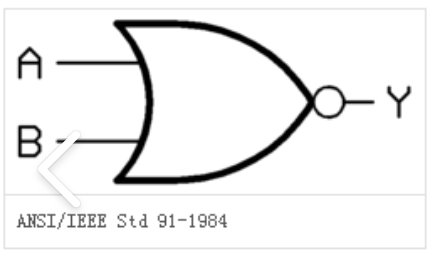
1. 与非门

就是先进行与运算（先经过与门）再经过非门（经过非门）。只有所有输入是高电平时，输出才是低电平（对应“0”）；否则输出高电平（对应“1”）



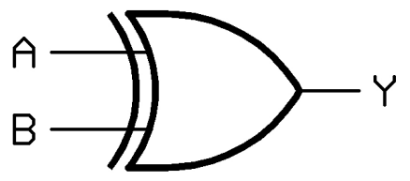
1. 或非门

就是先进行或运算（先经过或门）再经过非门（经过非门）。只有所有输入是低电平时，输出才是高电平（对应“1”）；否则输出低电平（对应“0”）



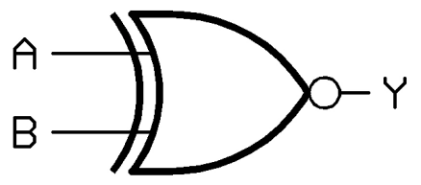
1. 异或门

若两个输入的电平相异，则输出为高电平1；若两个输入的电平相同，则输出为低电平0。



1. 同或门

当输入电平相同时，输出为高电平（对应1）；否则输出低电平（对应0）。



**中断**

中断是指导致单片机中止目前正在运行的主程序的特定现象或事件。在发生这些事件时，系统会暂停目前正在运行的程序，转而运行处理对应事件的程序。直到处理对应事件的程序运行完后再返回运行主程序。

以上为中断的广泛定义。若仔细区分，可以分为异常和中断。前者主要指系统内部的异常情况（如硬件错误Hardware Fault）；后者主要指一些特定的外部事件（如定时器装载值溢出，DMA传输完成），这些事件可以由程序指定是否中止主程序并进行处理。