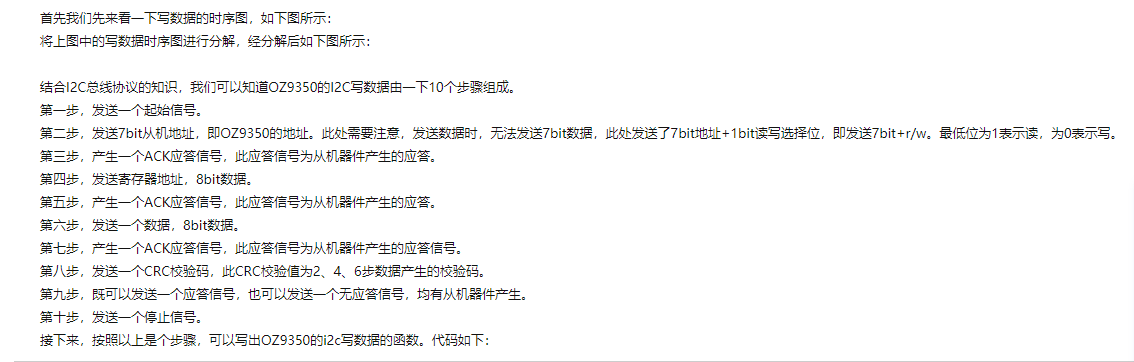
**20210412**

1. 学习I2C读写数据流程。

文本

描述已自动生成

2、github 创建项目，用GitHub客户端克隆，用vscode编译修改上传。协同合作。

**20210413**

1. 数据解析。一个地址存放一个字节，小端操作：指数据来了先存低四位再存高四位。小端操作需要用小端解析方法，51属于大端操作。

00000000: 3412 cdab badc 0100

00000008: 0e00 0000 0000 0000

00000010: 0000 0000 0073 5f7b

00000018: d812 aa33 e567 0000

1. 安装VS2019，net.frame版本太低。
2. 测试板子上的芯片传输数据。阻抗匹配。
3. 发展对象信息填写。
4. 书写报考部队文职的承诺书。

char ch;

FILE \*fp;

FILE \*fp = fopen("D:\\demo.txt", "r+");

ch = fgetc(fp);

while( (ch=fgetc(fp)) != EOF ){

putchar(ch);}

表示从D:\\demo.txt文件中读取一个字符，并保存到变量 ch 中。

在文件内部有一个位置指针，用来指向当前读写到的位置，也就是读写到第几个字节。在文件打开时，该指针总是指向文件的第一个字节。使用 fgetc() 函数后，该指针会向后移动一个字节，所以可以连续多次使用 fgetc() 读取多个字符。

6、fprintf(“ 正点原子”)；将正点原子 打印到串口，因为单片机自己重定义了fputc函数,将数据发送到串口了。

int fputc(int ch, FILE \*f)

{ while((USART1->SR&0X40)==0);//循环发送,直到发送完毕

USART1->DR = (u8) ch; return ch;

}

**20210414**

**1、**上午precede

Exclaim

proclaim

Suicide

pesticide

Decide

Concise

precise

Intercede

Exceed

excess

Unprecedentedly

Offer

Prefer

Differ

transfer

Recede

Introduce

Conduct

Seduce

Reduce

produce

Include

Exclude

Conclude

seclude

Declaim

Reclaim

acclaim

recession

Migrate

Emigrate

Immigrate

transmigrate

Subject

Object

Reject

Project

inject

Expensive

Pension

compensive

Depend

Independent

suspend

1. 安装Ecantools 调试单片机can总线上传感器发出的数据。
2. 飞哥设计的板子有7个温湿度传感器，一个i2c的多路多路开关TCA9548APWR。但是没有MCU需要外加主芯片。
3. I2C发送数据结构：从机地址 从机寄存器地址 长度 &数据 。其中从机寄存器地址如果没有需要用0x00。
4. 使用VSCODE修改代码，可以直接修改KEIL打开的代码。
5. STM32cubemax 可以自行搭配环境以后可能会用得到。
6. CAN中过滤器有两种模式，一种是屏蔽模式：规定了一组标识符的规格，一种是标识模式：规定了每个标识符必须与其相对应。CAN中显性电平是0，隐形电平是1.

**20210415**

1、

Respect

Suspect

spectator

Prospect

Inspect

expect

Bankrupt

Disrupt

Corrupt

erupt

Dispose

Propose

position

Expose

Compose

oppose

Expensive

Pension

Compensate

Reverse

Convert

Diverse

Introvert

extrovert

Attract

Contract

Extract

Distract

Constant

tractor

Visual

Evident

Evidence

Revive

Vivid

survive

Visible

Revise

Previse

supervise

2、命名空间的问题：

命名空间：实际上就是一个由程序设计者命名的内存区域，程序设计者可以根据需要指定一些有名字的空间域，把一些全局实体分别放在各个命名空间中，从而与其他全局实体分隔开来。现在命名空间成员包括变量a和b，注意a和b仍然是全局变量，仅仅是把它们隐藏在指定的命名空间中而已。如果在程序中要使用变量a和b，必须加上命名空间名和作用域分辨符“::”，如nsl::a，nsl::b。

命名空间的作用：是建立一些互相分隔的作用域，把一些全局实体分隔开来。

在声明一个命名空间时，花括号内不仅可以包括变量，而且还可以包括以下类型：

·变量(可以带有初始化)；

·常量；

·数(可以是定义或声明)；

·结构体；

·类；

·模板；

·命名空间(在一个命名空间中又定义一个命名空间，即嵌套的命名空间)。

**使用命名空间解决名字冲突**

**标准命名空间std**

为了解决C++标准库中的标识符与程序中的全局标识符之间以及不同库中的标识符之间的同名冲突，应该将不同库的标识符在不同的命名空间中定义(或声明)。标准C++库的所有的标识符都是在一个名为std的命名空间中定义的，或者说标准头文件(如iostream)中函数、类、对象和类模板是在命名空间 std中定义的。std是standard(标准)的缩写，表示这是存放标准库的有关内容的命名空间，含义请楚，不必死记。

**20210416**

**-y** rainy wordy hairy 多····的

Diffuculty discovery mastery 抽象名词

-ism 主义 流派 特性

Individualism capitalism modernism humanism

-ist 人 ····家

Artist communist scientist

-ive 有···性质的/ 作用的

Attractive impressive creative

-ness 抽象名词

Weakness kindness darkness

-less 无···的

Homeless useless hopeless

-ize ····化

Realize centralize industrialize

-ment 行为的过程/结果

Punishment development attachment basement

-ward 向···的 朝···的

Downward backward northward

* able 能···的

useable moveable adaptable

-al 具有····性质的

Personal natural regional

抽象名词 refusal arrival 人 survival criminal

-ate 做 generate create

人 graduate candidate advocate

-ed 形容词后缀

Gifted skilled warm-harted

动+ed 已被··的 被···的

Extended educated married

-ten 使变成 shorten sharpen strengthen

-ence 抽象名词

Difference confidence dependence

-ful 富有····具有···性质的

Useful hopeful powerful

-fy 使变成

Simplify purify

-ic ····的

Historic basic realistic

-ion 抽象名词

Action correction discussion

**命名空间属于类库 DLL（链接库）中包含了类库**



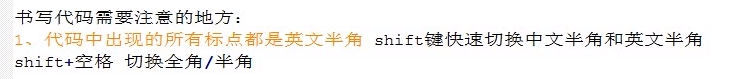
.net是开发平台 .net是厨房 .net　framwork 是锅碗瓢盆

Vs与winform 都是C/S交互模式 客户端与服务器，需要在电脑安装客户端。

C#项目包括：解决方案：很多项目：类文件（．ｃｓ文件）

图示

描述已自动生成



／／／三个斜线为文档注释　，给函数添加。

文本

描述已自动生成 文本

描述已自动生成

变量　　先声明再赋值最后使用；　highＳｃｈｏｏｌ；

**20210419**

strip() 方法用于移除字符串头尾指定的字符（默认为空格或换行符）或字符序列。

注意：该方法只能删除开头或是结尾的字符，不能删除中间部分的字符。

split() 通过指定分隔符对字符串进行切片，如果参数 num 有指定值，则分隔 num+1 个子字符串

str.split(str="", num=string.count(str)).

str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。

num -- 分割次数。默认为 -1, 即分隔所有。

返回分割后的字符串列表。

例

str = "Line1-abcdef \nLine2-abc \nLine4-abcd";

print str.split( ); # 以空格为分隔符，包含 \n

print str.split(' ', 1 ); # 以空格为分隔符，分隔成两个

['Line1-abcdef', 'Line2-abc', 'Line4-abcd']

['Line1-abcdef', '\nLine2-abc \nLine4-abcd']

range()函数内只有一个参数，则表示会产生从0开始计数的整数列表：

range(4)

[0, 1, 2, 3] #python 返回值

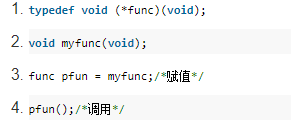
range(3, 10)

[3, 4, 5, 6, 7, 8, 9]

**20210420**

fvoid (\*func)(int)

**func是一个函数指针，**函数指针是指向函数的指针变量，他的本质是一个指针变量。**它的返回类型为空，它所指向的函数接收一个int型的参数。若是写成void \*func(int)则变成了:func是一个函数，它的返回类型是空指针，它接受一个int型参数。**

将myfunc函数的首地址赋给指针pfun

其实这里不能称为函数名，应该叫做指针的变量名。这个特殊的指针指向一个返回整型值的函数。指针的声明必须和它指向函数的声明保持一致。

其主要的区别是一个是指针变量，一个是函数。

函数定义

def 函数名(参数列表):

//实现特定功能的多行代码

[return [返回值]]

f \_\_name\_\_ == '\_\_main\_\_':的运行原理

每个python模块（python文件，也就是此处的 test.py 和 import\_test.py）都包含内置的变量 \_\_name\_\_，当该模块被直接执行的时候，\_\_name\_\_ 等于文件名（包含后缀 .py ）；如果该模块 import 到其他模块中，则该模块的 \_\_name\_\_ 等于模块名称（不包含后缀.py）。

而 “\_\_main\_\_” 始终指当前执行模块的名称（包含后缀.py）。进而当模块被直接执行时，\_\_name\_\_ == 'main' 结果为真。

用中括号是列表，小括号是元组，他们俩都是元素之间用逗号隔开

Bytes 主要是用于二进制的数据处理的数据类型

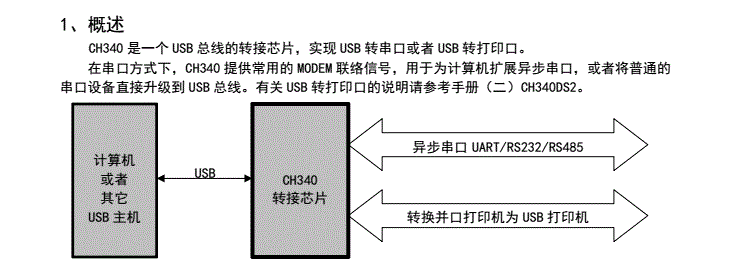
type(b'xxxxx') <class 'bytes'>

type('xxxxx') <class 'str'>

bytes是byte的序列，而str是unicode的序列。

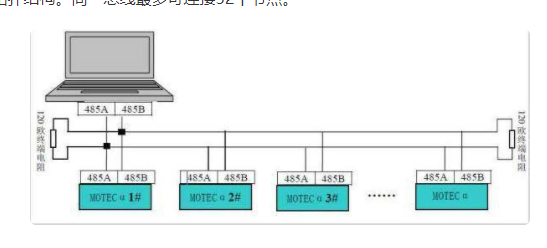
**20210421**

**1．**串口驱动 CH340模块 串口通讯(Serial Communication)是一种设备间非常常用的串行通讯方式，因为它简单便捷，因此大部分电子设备都支持该通讯方式。



**2.**

卸载讯读，需要到它安装路径下 点uinstall 卸载。控制面板中找不到。

**3.**

**485**串口调试，用485转串口模块，向传感器发送 闻询指令 传感器返回 当前数据。



**4.**

**Bootloader**

Boot loader(系统引导程序)，操作系统内核运行之前运行的一段程序。它首先完成系统硬件的初始化，包括时钟的设置、存储区的映射等，设置堆栈指针等；然后把操作系统内核从flash区拷贝带ram区，并跳转到内核的入口，将系统的控制权交给操作系统，从此系统的运行和Boot loader再无任何关系。

**内核**是一个操作系统的核心。它负责管理系统的进程、内存、设备驱动程序、文件和网络系统等等，决定着系统的性能和稳定性。是连接应用程序和硬件的桥梁。

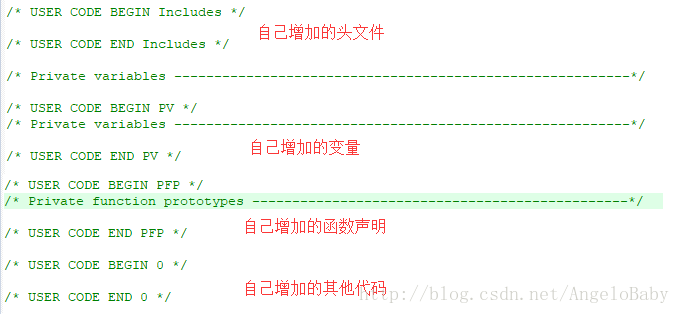
**5.**

看门狗是单片机里的一种技术（软硬件都可以），其目的是为了保护芯片避免其进入死循环（或者说程序跑飞）。看门狗有一个输入端和一个输出端。看门狗和单片机程序的主体部分连接，如果主体部分运行正常，那么每一段时间就会给看门狗的输入端一个信号，这时候看门狗就会知道程序运行正常，就不会做出反应。如果看门狗较长一段时间内没有收到信号，说明程序运行不正常，此时由看门狗输出一个信号，该信号会让单片机的主体程序复位进而起到保护的作用。

**6.**

JTAG（JointTestAcTIonGroup，联合测试行动小组）是一种国际标准测试协议，主要用于芯片内部测试及对系统进行仿真、调试，JTAG技术是一种嵌入式调试技术，它在芯片内部封装了专门的测试电路TAP（TestAccessPort，测试访问口），通过专用的JTAG测试工具对内部节点进行测试。

JTAG和SWD是ARM板一种调试模式，后面三种都是实现这种模式的一种工具：Jlink,Ulink,Stlink。

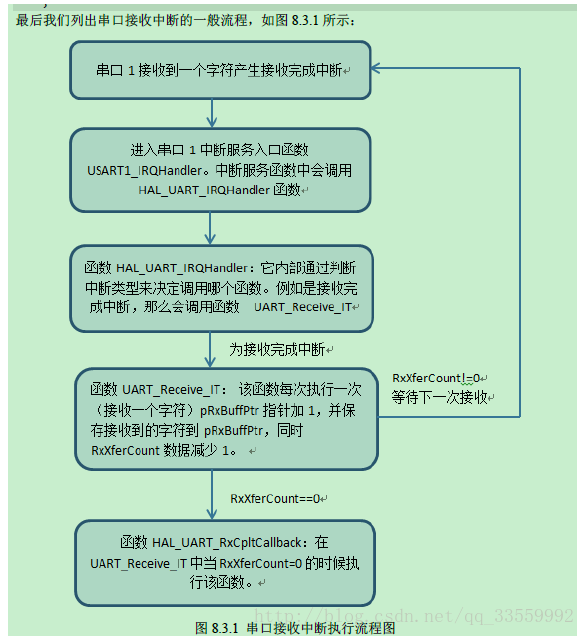


**20210422**

回调函数：回调函数就是一个通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这是回调函数。

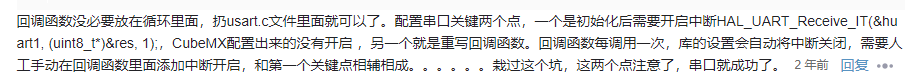
回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应

2．HAL库串口中断



## HAL\_UART\_Receive\_IT

这个函数不是用来接收数据的，他是用来打开中断，配置串口中断的。main函数里面首先写下它，否则无法进入串口中断（亲测如此），其次还要在回调函数里面添加这个函数（因为之前就说过一旦进入回调函数，串口中断就会关闭），为了下一次接收数据考虑，需要这么做。



**3、硬件流控制，与软件流控制。**

文本

描述已自动生成

硬件流控制硬件流控制常用的有RTS/CTS流控制和DTR/DSR（数据终端就绪/数据设置就绪）流控制。硬件流控制必须将相应的电缆线连上，用RTS/CTS（请求发送/清除发送）流控制时，应将通讯两端的RTS、CTS线对应相连，数据终端设备（如计算机）使用RTS来起始调制解调器或其它数据通讯设备的数据流，而数据通讯设备（如调制解调器）则用CTS来起动和暂停来自计算机的数据流。

**比如单片机的主频只有20M或30M，ARM的处理能力可能是200M，PC机的处理能力是几个G，这种处理能力的不匹配造成了传输的时候数据容易丢失。硬件流控就是来解决这个速度匹配的问题。它的基本含义非常简单，当接收端接收到的数据处理不过来时，就向发送端发送不再接收的信号，发送端接收到这个信号之后就会停止发送，直到收到可以继续发送的信号再继续发送。因此流控本身是可以控制数据传输的进度，进而防止数据丢失。**

图示

描述已自动生成RTS 信号在数据没有被读取之前都是保持在高电平状态，我们可以看到在 Start 之前都是高电平，这也就是告诉发送端，数据还没有被拿走，请发送端等待，一旦数据被 DMA 或者 CPU 从 DR 寄存器读取之后，RTS 就释放高电平，变为低电平，这时候发送端如果想发送数据的话就可以直接发送了。

图示, 示意图

描述已自动生成

**如果使能了硬件流控的功能，就会增加一个实时检测的步骤。**在图3中，当没有收到CTS信号的时候，TX 发送线上数据是连续发送的，表现形式为：在 STOP 位后紧跟着就是下一个数据的 Start bit。

当 Data 2 还在 TX 线上进行发送的时候，如果此时在 CTS 信号上检测到了高电平，即使在 Data 2 的 STOP 位发送完之前写入了 Data 3，在当前的字节发送完之后是不会马上发送新写入的数据的，而是要等待，直到在 CTS 管脚上检测为低电平后，TX脚上才会开启 Data 3 的 Start 信号。

文本

描述已自动生成**20210423**

在STM32中 for(j;j<8;j++) j最好在for中赋初值。

**20210425**

1. **接收到**0X56 只是十六进制显示，本质上8位二进制数。可直接当成十进制操作与45 进行比较。

**2、485** 接收缓存器，每次接受完后 将index 复位，下次接受从0 开始。例如接收数据为 485[5] = {1, 2, 3 } 接收缓存器为 buff1[5], 则buff1[0] = 1；buff1[1] = 2；buff1[2] = 3，如果不将index 复位buff1[3] = 1，buff1[4] =2，等下一次中断接收buff1[0] = 3.

**意思是必须等一个buffer 填满之后才会 填下一个。**

**20210426**

1. 存储单元一般应具有存储数据和读写数据的功能，一般以8位二进制作为一个存储单元，也就是一个字节。每个单元有一个地址，是一个整数编码，可以表示为二进制整数。32K字节的 存储地址为 0X 8000
2. 图示, 示意图

   描述已自动生成

表格

描述已自动生成

为了提高转换的精确度，ADC使用一个独立的电源供电，过滤和屏蔽来自印刷电路板上的毛刺干扰，ADC的电源引脚为VDDA，独立的电源地VSSA，如果有VREF- 引脚(根据封装而定)，它必须连接到VSSA，确保共地。

3、ctrl + M 是测量 ，shift +c 是取消。Efc 设置中心点

Shift + E 定位中心 ctrl + w 原理图布线

4、稳压二极管：稳压二极管，是指利用pn结反向击穿状态，其电流可在很大范围内变化而电压基本不变的现象，制成的起稳压作用的二极管。图示, 示意图

描述已自动生成大电压（超过二极管击穿电压）过来 二极管击穿，与大地短路保护电路。

图片包含 蛋糕, 装饰, 灯光, 桌子

描述已自动生成

滤波电容的作用：c = Q/ U; c是定值，所以电荷量与电压按一定比例，U增大Q增大 超过电容的临界值，电容损坏，图中电容属于滤波电容，当有白噪声影响的时候，由于电容有充放电的特性，所以电容不能突变从而过滤掉杂波，例如当有12.1V的电压来了，电容充电，引脚4的电压不可能突变到12.1V。

10uf电容对于滤除低频干扰有较好作用，但对于高频干扰，电容呈现感性，阻抗很大，无法有效滤除，因此再并一个0.1uf的电容滤除高频分量．

电容并联放在IC电源脚到地，一般是为了滤除那些对系统不利的交流信号。

由于实际供电电源，夹杂着高频以及低频干扰杂波，10uf电容对于滤除低频杂波有较好作用，但对于高频杂波，电容呈现感性，阻抗很大，无法有效滤除，因此再并一个０.１uf的电容滤除高频杂波。

从理论上（即假设电容为纯电容）说，电容越大，阻抗越小，通过的频率也越高。但实际上超过1μF 的电容大多为电解电容，有很大的电感成份，所以频率高后反而[阻抗](https://baike.baidu.com/item/%E9%98%BB%E6%8A%97)会增大。有时会看到有一个电容量较大电解电容并联了一个小电容，这时大电容滤低频，小电容滤高频。电容的作用就是通交流隔直流，通高频阻低频。电容越大高频越容易通过。具体用在滤波中，大电容（1000μF）滤低频，小电容（20pF）滤高频。曾有网友形象地将滤波电容比作“水塘”。由于电容的两端电压不会突变，由此可知，信号频率越高则衰减越大，可很形象的说电容像个水塘，不会因几滴水的加入或蒸发而引起水量的变化。它把电压的变动转化为电流的变化，频率越高，峰值电流就越大，从而缓冲了电压。滤波就是充电，放电的过程。

**电感**：电感器(Inductor)是能够把电能转化为磁能而存储起来的元件。电感器的结构类似于变压器，但只有一个绕组。电感器具有一定的电感，它只阻碍电流的变化。如果电感器在没有电流通过的状态下，电路接通时它将试图阻碍电流流过它;如果电感器在有电流通过的状态下，电路断开时它将试图维持电流不变。电感器又称扼流器、电抗器、动态电抗器。

**LC 调谐电路 ，LC滤波电路**

墙上的涂鸦

中度可信度描述已自动生成**低频扰动信号，通过大电容滤除。**

20210427

1、电容电阻外形尺寸与封装的对应关系是: 0402=1.0x0.5 0603=1.6x0.8 0805=2.0x1.2 1206=3.2x1.6 1210=3.2x2.5 1812=4.5x3.2 2225=5.6x6.5

电容本身的大小与封装形式无关，封装与标称功率有关。它的长和宽一般是用毫米表示的。但是型号是采用的英寸的表示方法。

R=U/I=U方/P P = u方/R

文本

描述已自动生成

日历

中度可信度描述已自动生成

表格

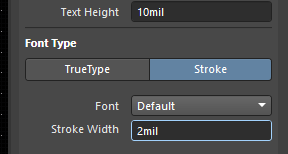
描述已自动生成

**电容10uf 用 0804 封装**

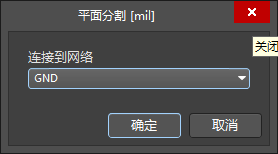
2、在PCB中快捷键D 找出类对象

3、改变元器件标识大小，首先选中文本右键查找相似对象，



保存

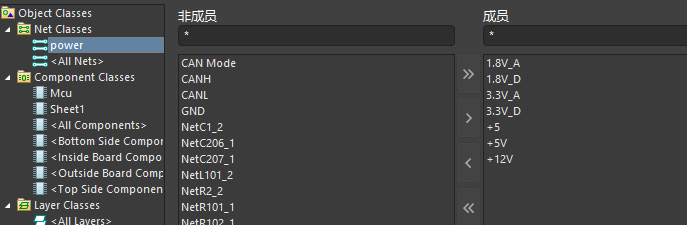
Ctrl A 全选，按A 定位器件文本 放在中间。

1. eos 设置原点。Q 切换 mil与mm
2. ms 移动
3. 新建层可以通过给层建立网络。

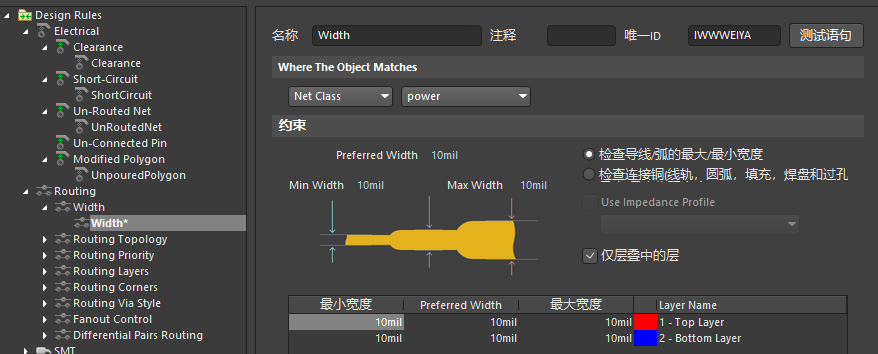


我设置的 2 布线 3 打孔。

1. 在画pcb过程中 n 是显示 和隐藏连接线。
2. 在设计中 查看类 设计电源类；



* 中打开PCB 可以设置power类高亮；及所有电源线高亮。



选择net class 里面 power 类 修改其中电源走线的线宽。

可能需要使能

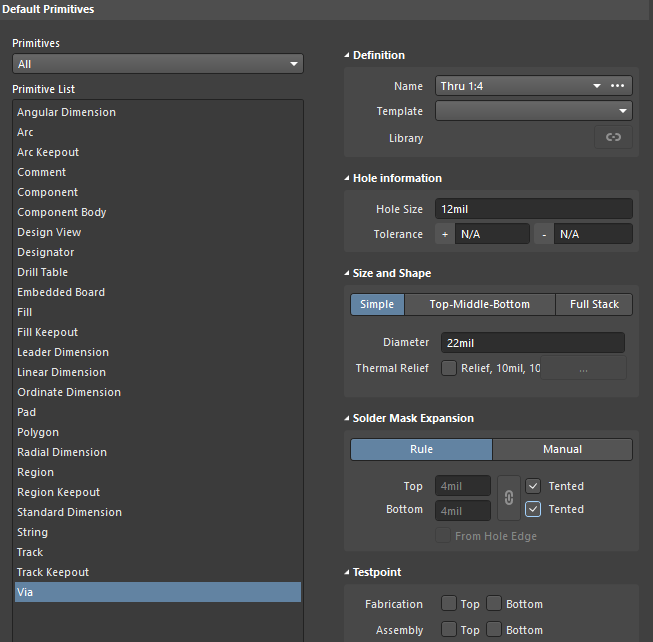
1. 1mm等于39.37个mil

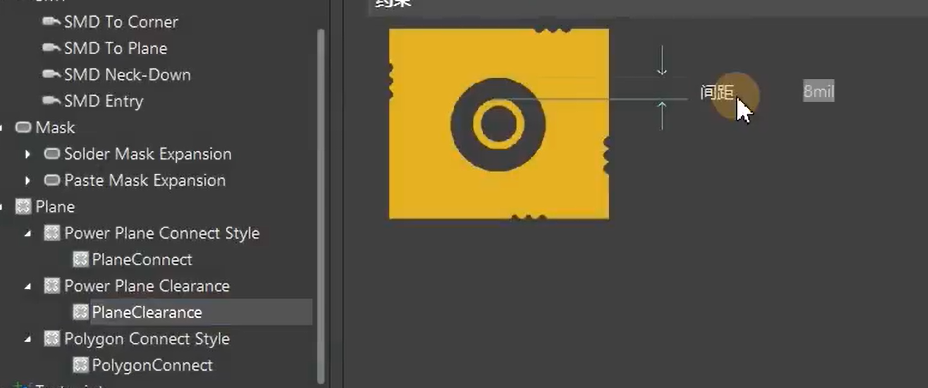


设置过孔，空大小为10则直径为20/18/22

1. 焊盘 采用十字连接，过孔全连接。



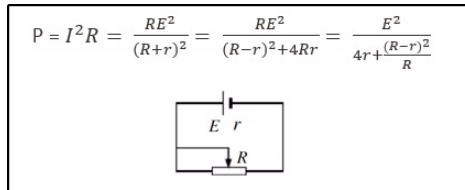


1. 负片层 过孔不相连的情况下设置绝缘间距

**20210428**

1. **滤波电容尽量贴近芯片。就近保护芯片。**
2. **命名尽量统一例如芯片为u101 那么这个芯片连接的元器件 电容电感的 都为 R10？开头。**
3. **高频低频 功率 电源 ，布局。**
4. **电源线根据电流大小设置线宽。特性阻抗配。**
5. EMC全称Electromagnetic Compatibility，即电磁兼容，要求电源模块等电子设备内部没有严重的干扰源及设备，或电源系统有较好的抗干扰能力。
6. **信号完整性：**信号完整形式指，在高速产品设计中由互连线引起的所有的问题。主要受这三个方面的影响。
7. 时序  
   2.噪声  
   3.电磁干扰（EMI）
8. **电压驱动的如**：场效应管，因为它的内阻很大，加电压控制时电流很小，近似为零，所以可以理解成：电压驱动；
9. **电流驱动的如：**普通的NPN、PNP型三极管，因为它的内阻较小，加电压控制时电流相对较大（一般小功率的都有100uA以上，大功率的可达20mA以上），所以可以理解成：电流驱动；
10. 电压驱动型器件内阻大，加电压后电流小，只需要较小的驱动功率，场效应晶体管(FET)是电压控制器件、MOS电路是电压控制器件，又比如气体型霓虹灯，电压低了连亮都不亮。亮了以后电流很小，那么就把它确定为电压型器件。

   电流驱动型器件内阻较小，加电压后电流较大，需要较大的驱动功率，双极型晶体管(BJT)是电流控制器件、TTL电路是电流控制器件，又比方发光二极管，其输入电压基本是一个确定值，提高电流可以改变亮度，减小电流降低亮度，它确定为电流型器件。

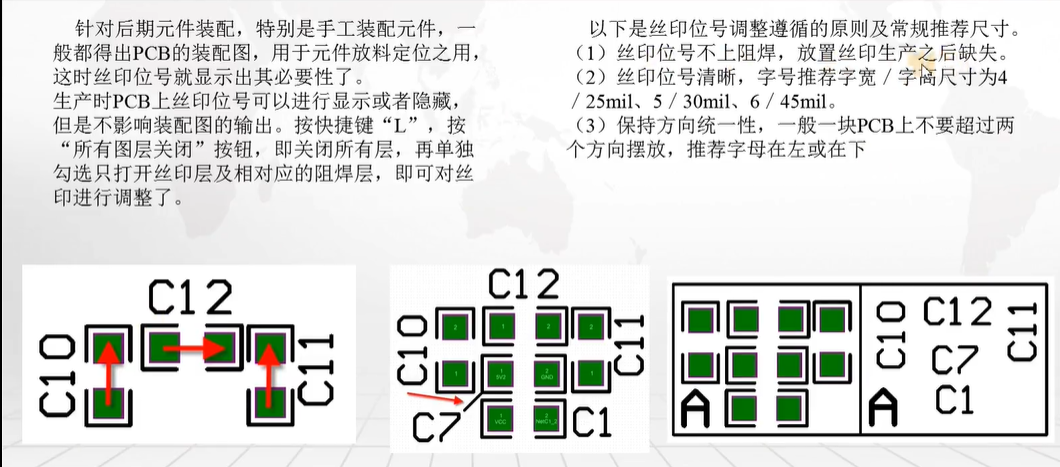
1. 

纯电阻电路

在中学物理电学中曾讲述这样一个问题：把一个电阻为R的用电器，接在一个电动势为E、内阻为r的电池组上，在什么条件下电源输出的功率最大呢？当外电阻等于内电阻时，电源对外电路输出的功率最大，这就是纯电阻电路的功率匹配。假如换成交流电路，同样也必须满足R=r这个条件电路才能匹配

11、5V 12V 布线看电流大小选择合适宽度，

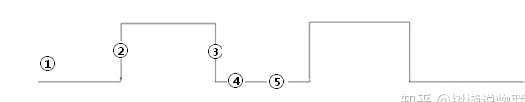




12、过孔连接net 哪层有 连哪层 。

**20210429**

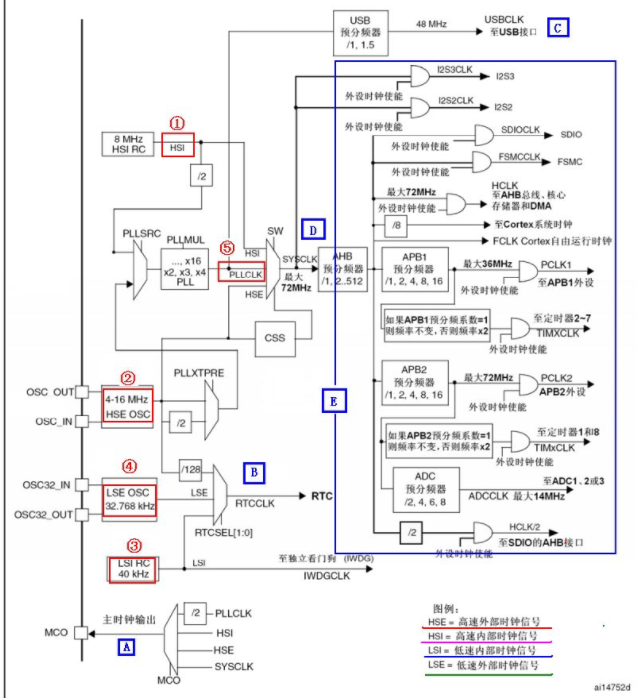
1. 输入捕获：输入捕获模式可以用来测量脉冲宽度或者测量频率



先设置输入捕获为上升沿检测，记录发生上升沿时TIMx\_CNT的值。然后配置捕获信号为下降沿捕获，当下降沿到来的时候发生捕获，并记录此时的TIMx\_CNT的值。这样，前后两次TIMx\_CNT的值之差就是高电平的脉宽。同时根据TIM的计数频率，我们就能知道高电平脉宽的准确时间。

2、LSI：40kHz低速内部RC振荡器，驱动独立看门狗，或用程序选择驱动RTC(实时时钟)

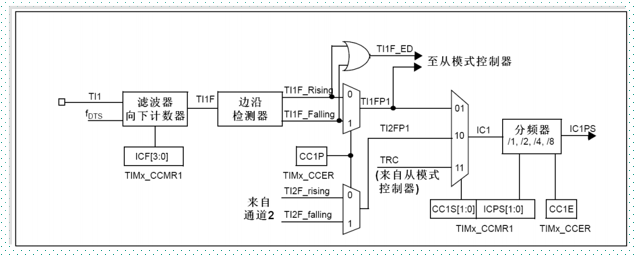
HSI：8MHz高速内部RC振荡器,可用于系统时钟；系统启动时默认使用该时钟为系统时钟，频率就是8MHz。



OSC\_OUT和OSC\_IN开始，这两个引脚分别接到外部晶振8MHz,第一个分频器

1. **输入捕获步骤：**

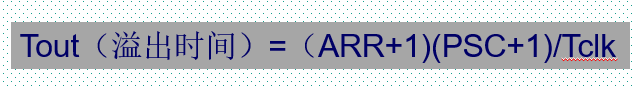




**这里分频器/1/2/4/8的意思是1个或2个或4个上升沿捕获一次。**



**3、**

**通过定时器中断配置每1000ms中断一次。**

**Tout(溢出时间)=(ARR+1)\*(PSC+1)/Tclk**

**其中Tout=1000ms；**

**Tclk来自系统时钟，APB1时钟为42MHZ，当不分频的时候为42MHZ，一般都分频为84MHZ。**

**令PSC=8399；**

**Tclk=84MHZ;**

**溢出时间的意思就是一次分频的时间乘以重装载值；**

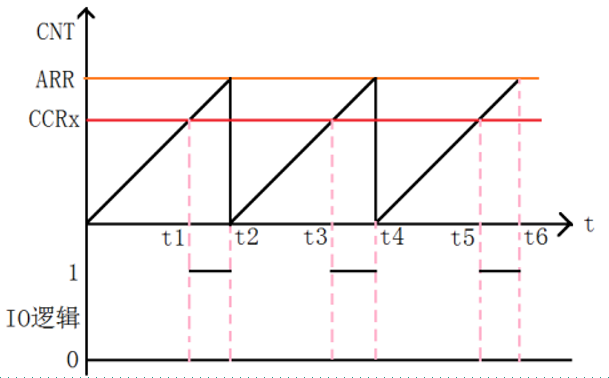
**(8399+1)/84MHZ=0.1ms;**

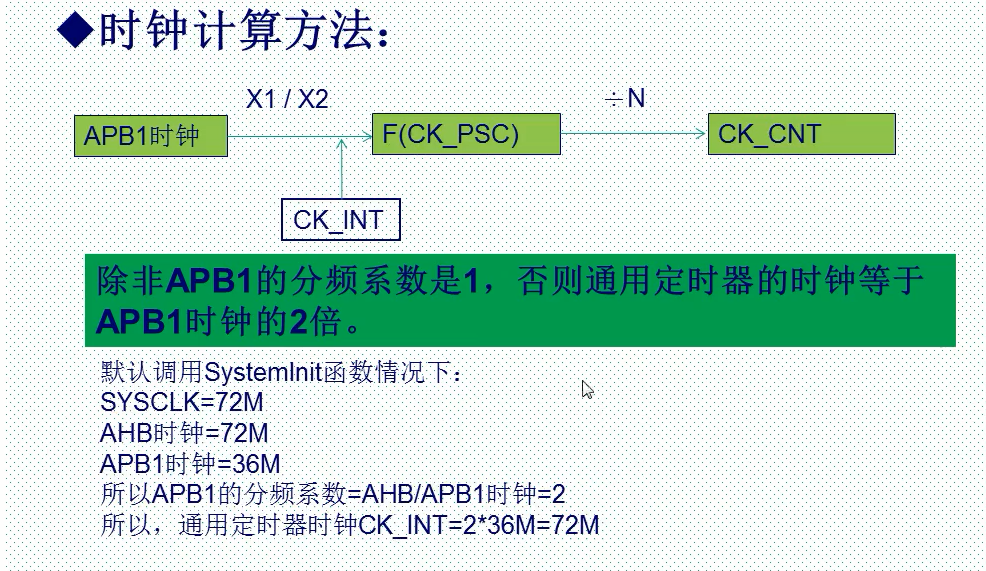
**所以Tout=（ARR+1）\*0.1ms；**

**ARR+1=1000/0.1=10000；**

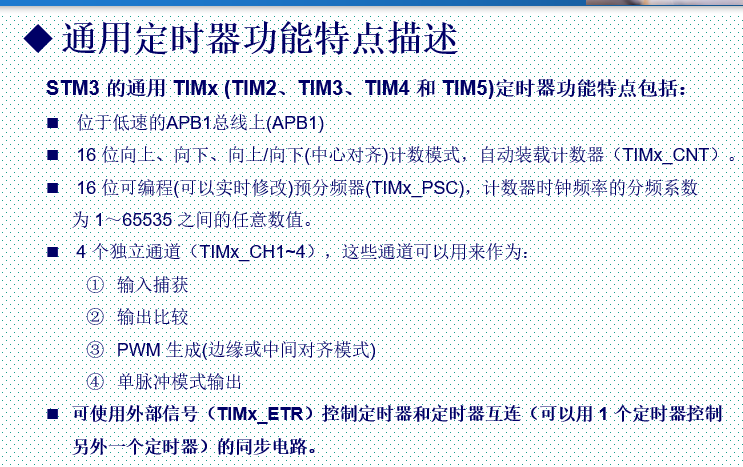
**所以ARR=9999**

1. **PWM**



**4、**

**5、1Mhz 频率 时钟周期为1us**

**6、**

**7、** **推挽输出模式一般应用在输出电平为0和3.3伏而且需要高速切换开关状态的场合。**在STM32的应用中，除了必须用开漏模式的场合，我们都习惯使用推挽输出模式。

**20210430**

1. **hal库写输入捕获，首先**

HAL\_TIM\_IC\_Start\_IT(&htim5, TIM\_CHANNEL\_1); *//启动输入捕获*

**HAL\_TIM\_IC\_Start\_IT**()                  函数用于使能定时器某一通道的输入捕获功能，并使能相应的中断

**HAL\_TIM\_IC\_Stop\_IT**()                 函数和开启功能相反，是关闭定时器某一通道的输入捕获功能和相应中断

**2、全局变量 不要用或指令，用赋值指令。**