# 最小系统有什么：

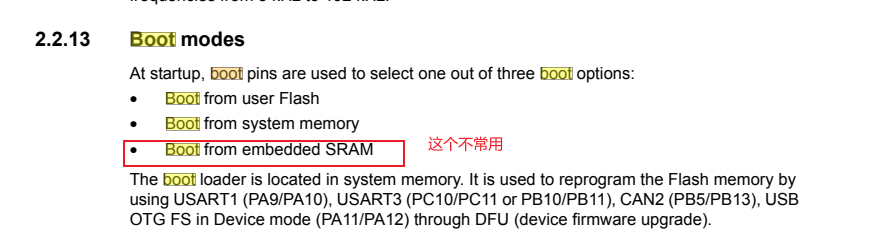
BOOT 电路

滤波电路

复位电路

外部晶振

电源电路



BOOT的3种启动模式

所谓启动，一般来说就是指我们下好程序后，重启芯片时，SYSCLK的第4个上升沿，BOOT引脚的值将被锁存。用户可以通过设置BOOT1和BOOT0引脚的状态，来选择在复位后的启动模式。

STM32上电或者复位后，代码区始终从0x00000000开始，三种启动模式其实就是将各自存储空间的地址映射到0x00000000中。

（1）从Flash启动，将主Flash地址0x08000000映射到0x00000000，这样代码启动之后就相当于从0x08000000开始。

（2）从RAM启动，将RAM地址0x20000000映射到0x00000000,这样代码启动之后就相当于从0x20000000开始。

（3）从系统存储器启动。首先控制BOOT0 BOOT1管脚，复位后，STM32与上述两种方式类似，从系统存储器地址0x1FFF F000开始执行代码。系统存储器存储的其实就是STM32自带的bootloader代码，在bootloader中提供了UART1的接口，通过此接口可以将用户所需的程序代码下载到主Flash中，下载完毕后，此时程序代码已经存储在主Flash当中，这时切换启动模式（从主Flash启动），复位后所执行的就是刚刚下载到Flash中的代码了。

STM32三种启动模式对应的存储介质均是芯片内置的，如下图：

为X的时候表示什么都不接



（1）用户闪存 ： 芯片内置的Flash。正常的工作模式。

（2）SRAM： 芯片内置的RAM区，就是内存。可以用于调试。

（3）系统存储器： 芯片内部一块特定的区域，芯片出厂时在这个区域预置了一段Bootloader，就是通常说的ISP程序。这个区域的内容在芯片出厂后没有人能够修改或擦除，即它是一个ROM区。启动的程序功能由厂家设置。

Main Flash memory:

是STM32内置的Flash，一般我们使用JTAG或者SWD模式下载程序时，就是下载到这个里面，重启后也直接从这启动程序。

System memory:

从系统存储器启动，这种模式启动的程序功能是由厂家设置的。一般来说，这种启动方式用的比较少。

系统存储器是芯片内部一块特定的区域，STM32在出厂时，由ST在这个区域内部预置了一段BootLoader，也就是我们常说的ISP程序，这是一块ROM，出厂后无法修改。

一般来说，我们选用这种启动模式时，是为了从串口下载程序，因为在厂家提供的BootLoader中，提供了串口下载程序的固件，可以通过这个BootLoader将程序下载到系统的Flash中。但是这个下载方式需要以下步骤：

Step1: 将BOOT0设置为1，BOOT1设置为0，然后按下复位键，这样才能从系统存储器启动BootLoader；

Step2: 最后在BootLoader的帮助下，通过串口下载程序到Flash中；

Step3: 程序下载完成后，又有需要将BOOT0设置为GND，手动复位，这样，STM32才可以从Flash中启动，可以看到，利用串口下载程序还是比较的麻烦，需要跳帽跳来跳去的，非常的不注重用户体验。

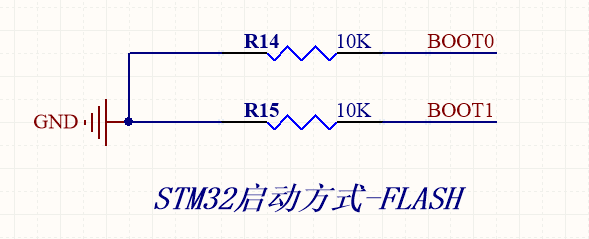
Embedded Memory:

内置SRAM，既然是SRAM，自然也就没有程序存储的能力了，这个模式一般用于程序调试。

假如我只修改了代码中一个小小的地方，然后就需要重新擦除整个Flash，比较的费时，可以考虑从这个模式启动代码（也就是STM32的内存中），用于快速的程序调试，等程序调试完成后，在将程序下载到SRAM中。

开发BOOT模式选择。

1、通常使用程序代码存储在主闪存存储器，配置方式：BOOT0=0，BOOT1=X;两个启动引脚通过10K电阻接地。



2、Flash锁死解决办法：

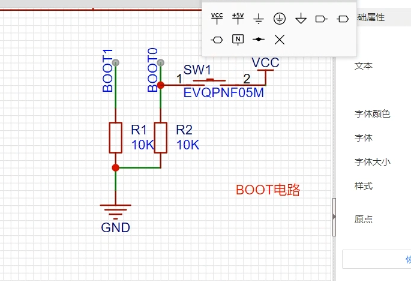
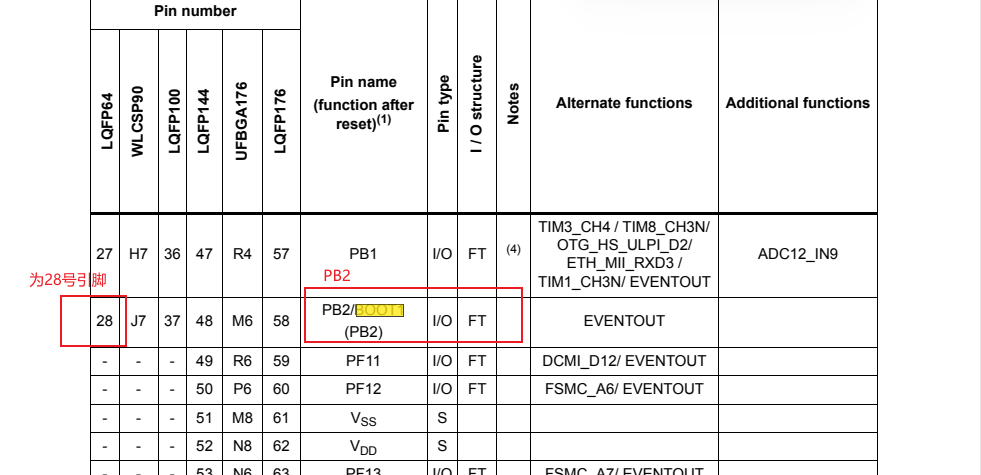
开发调试过程中，由于某种原因导致内部Flash锁死，无法连接SWD以及Jtag调试，无法读到设备，可以通过修改BOOT模式重新刷写代码。

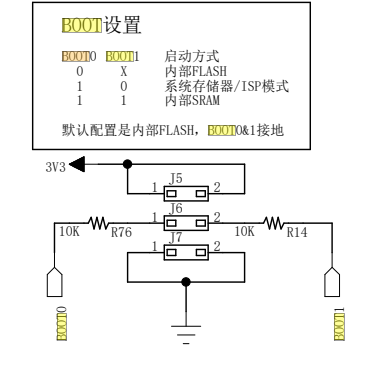
修改为BOOT0=1，BOOT1=0即可从系统存储器启动，ST出厂时自带Bootloader程序，SWD以及JTAG调试接口都是专用的。重新烧写程序后，可将BOOT模式重新更换到BOOT0=0，BOOT1=X即可正常使用。

嵌入式系统的启动还需要一段启动代码（bootloader)，类似于启动PC时的BIOS，一般用于完成微控制器的初始化工作和自检。

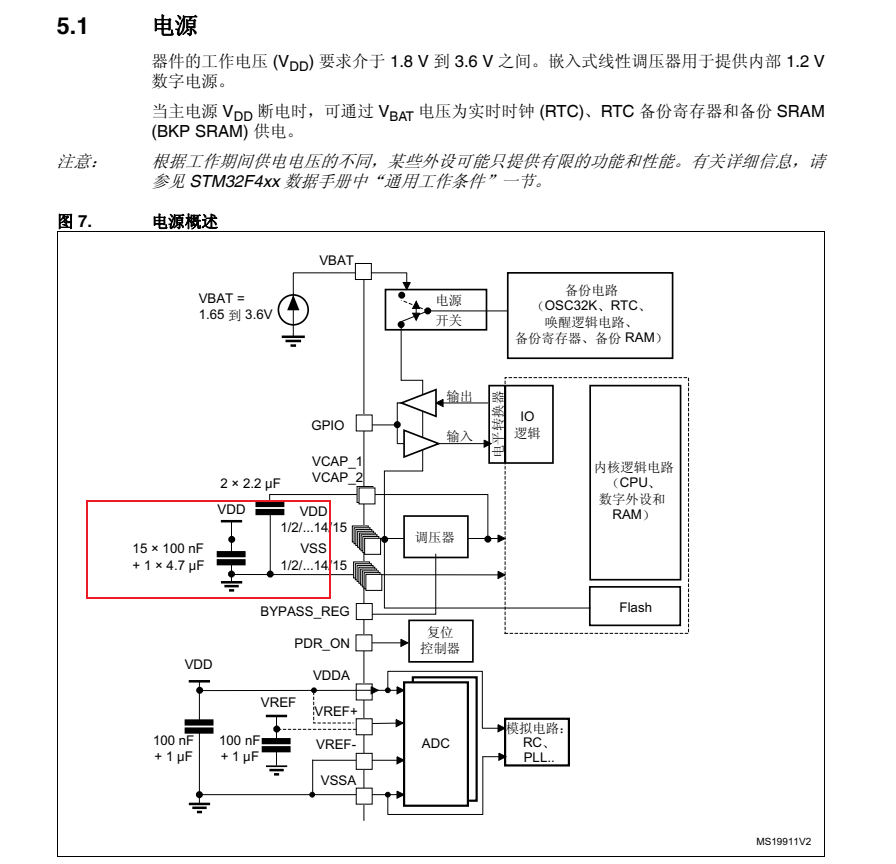
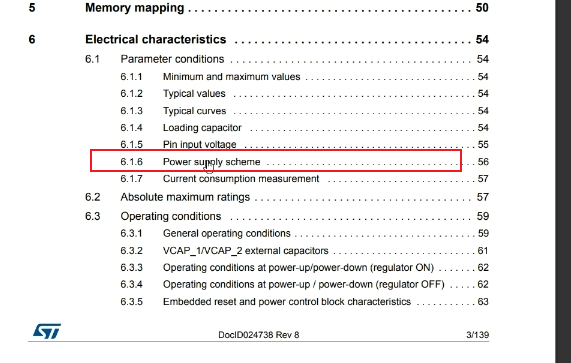
STM32的启动代码在startup\_stm32f10x\_xx.s（xx根据微控制器所带的大、中、小容量存储器分别为hd、md、ld）中，其中的程序功能主要包括****初始化堆栈、定义程序启动地址、中断向量表和中断服务程序入口地址，以及系统复位启动时，从启动代码跳转到用户main函数的入口地址。****

# 我们找到了BOOT0，没找到BOOT1，所以我们去手册上搜BOOT1,



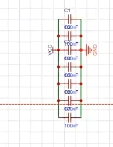
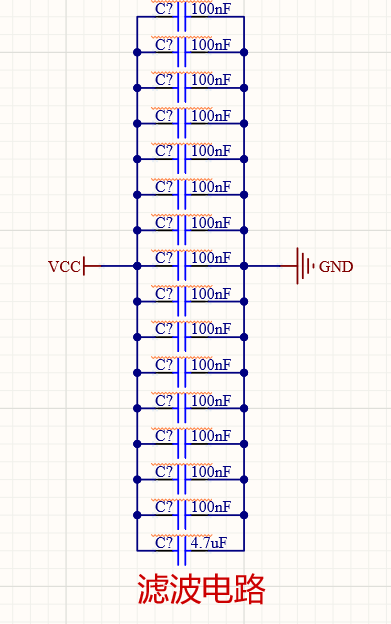
野火是使用排针和跳线的方式来实现的

# 滤波电路

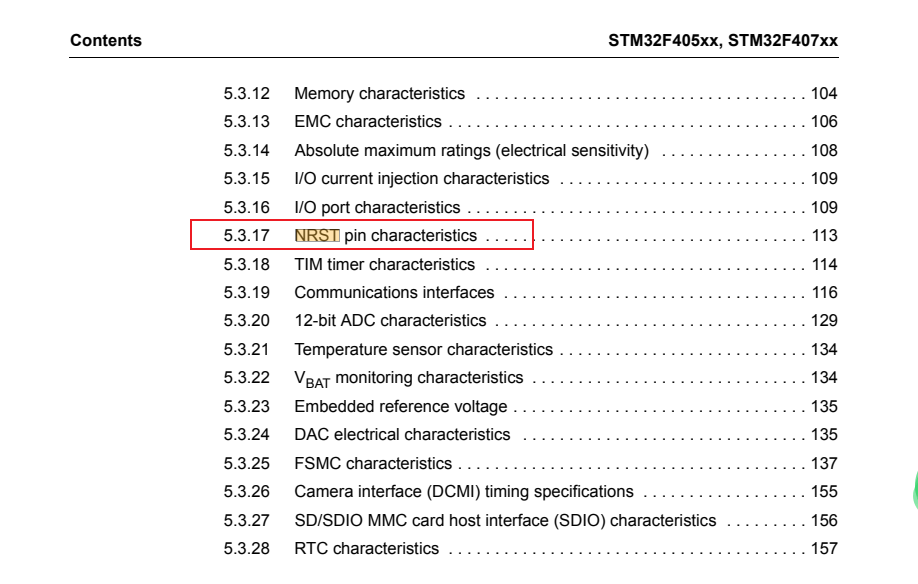


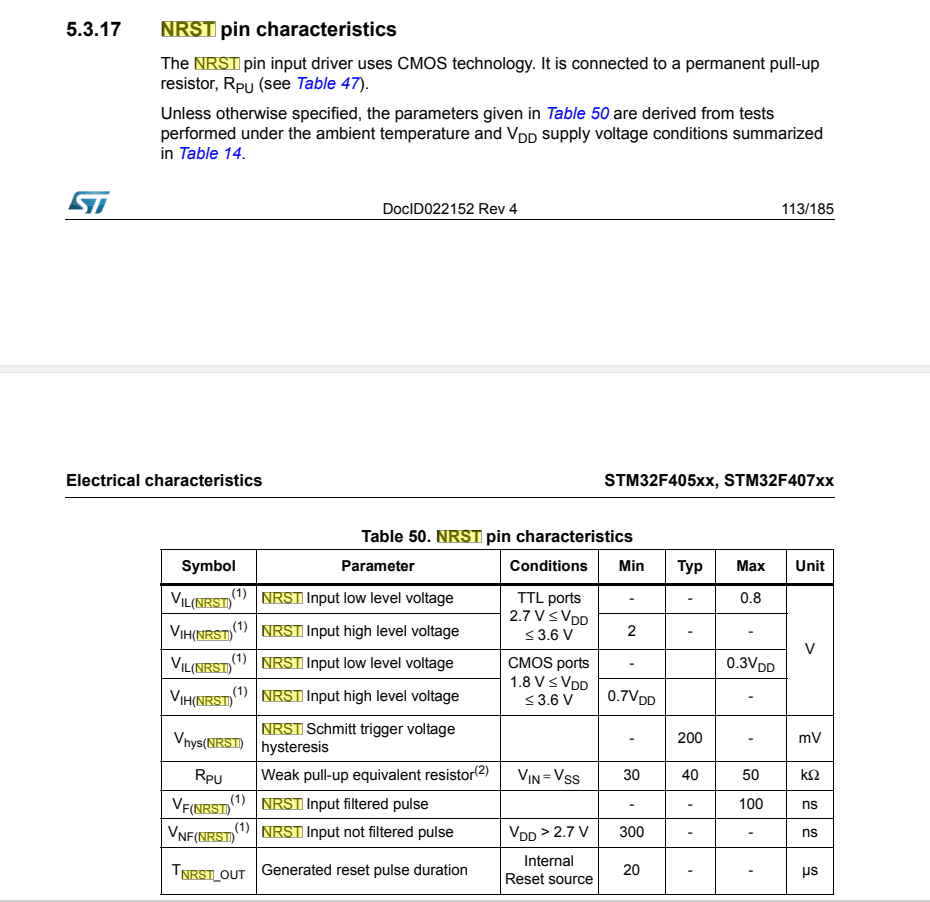
这个就是它的滤波电路

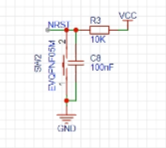
那这个就是15个100nF 加上 1个4.7uF的电容

并起来，

# 复位电路







# 外部晶振电路

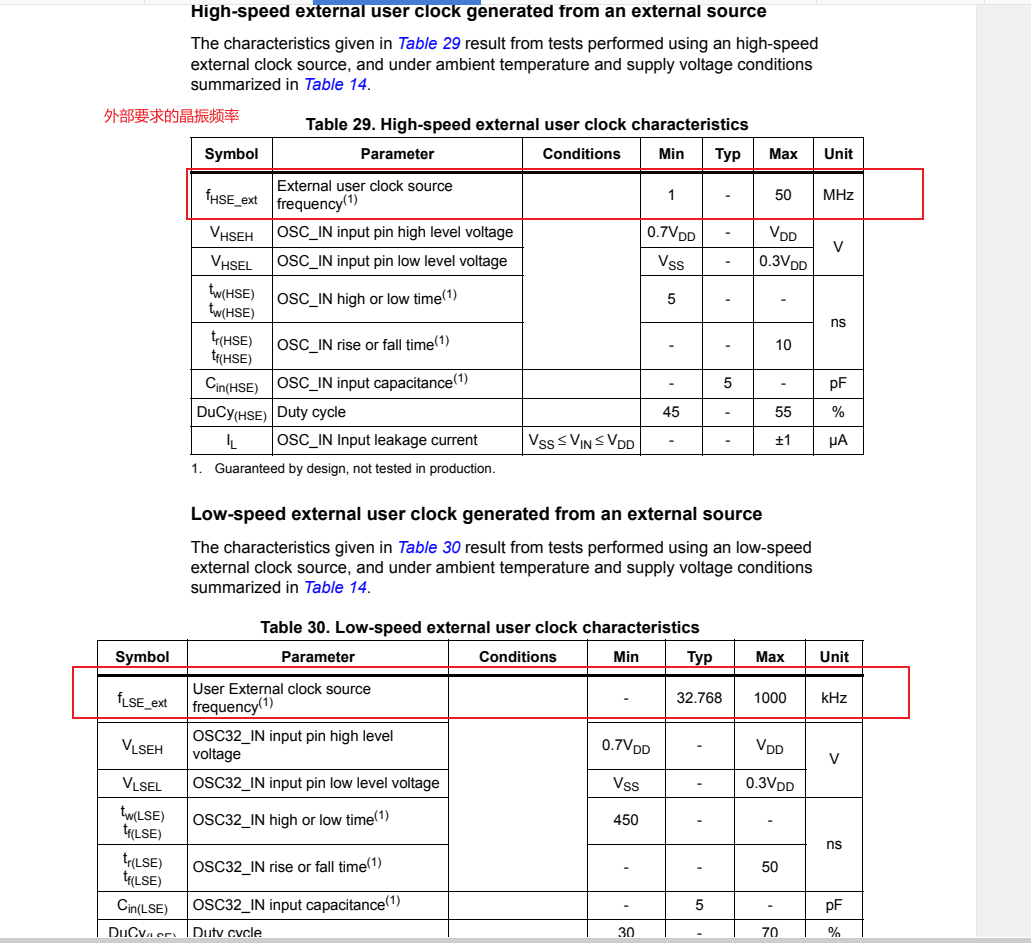
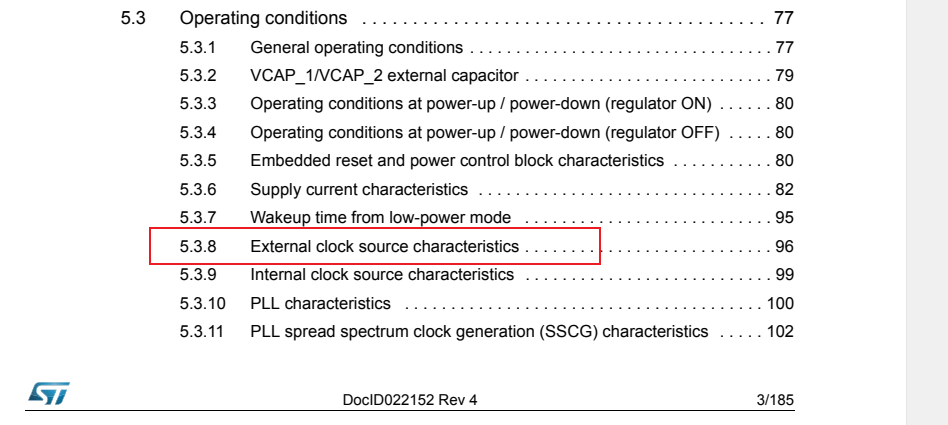
外部高速晶振 和 外部低速晶振

外部高速晶振通常是 8M

外部低速晶振通常是 32.768k

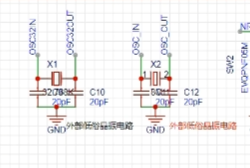
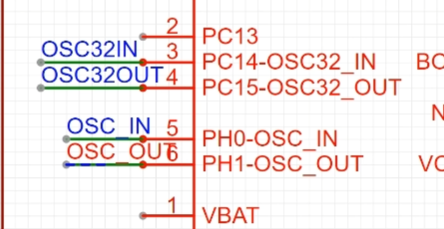
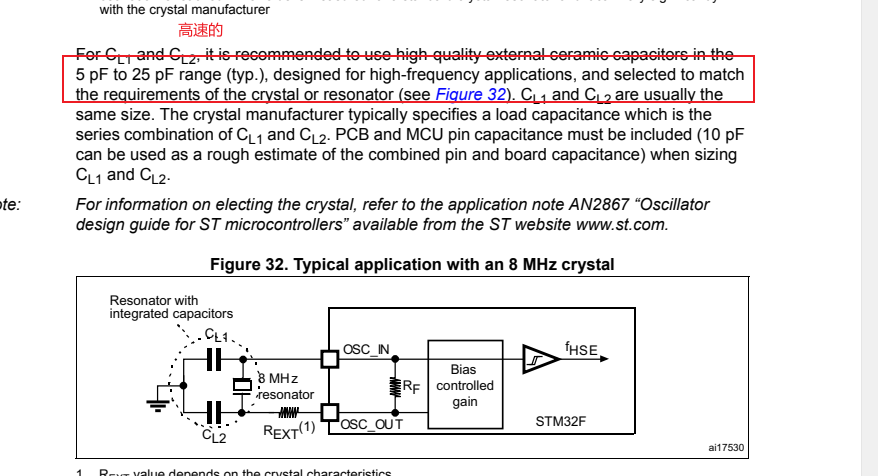
它们2边的电容都推荐是 20pF

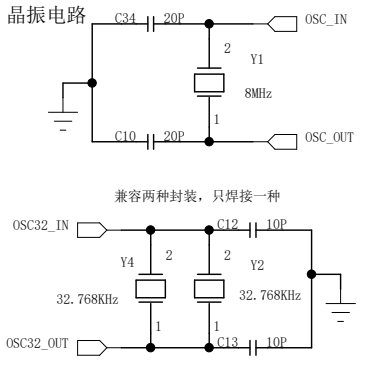
通过裂变器裂变成很多的

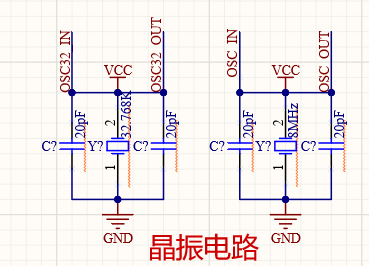


为什么是32.768因为 /1024 = 32

对于这个晶振要接多大的电容 他有提供公式：



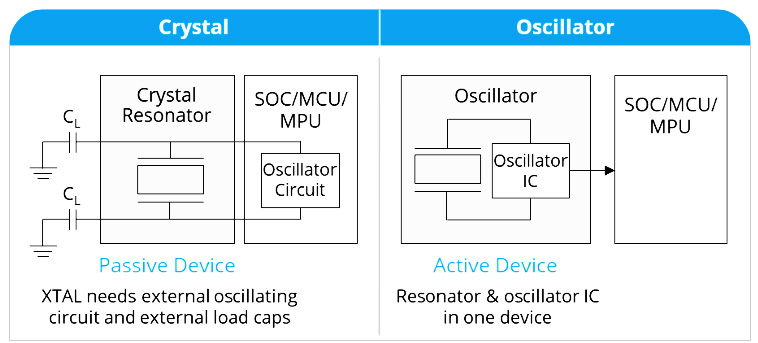




# 选择晶振：IMG_256

有源晶振自身即可起振,而无源晶振则需要外加专门的时钟电路才能起振

总的来说，有源晶振的精度、稳定性都要好于无源晶振



几乎每款智能数码产品都有其独特的单片机系统，而一个线路板则由诸多单元组成，如：单片机最小系统主要由电源、复位、振荡电路以及其它扩展部分构成。若要设计一个完整的振荡电路，晶振就是首先要考虑的不可或缺的频率元件。

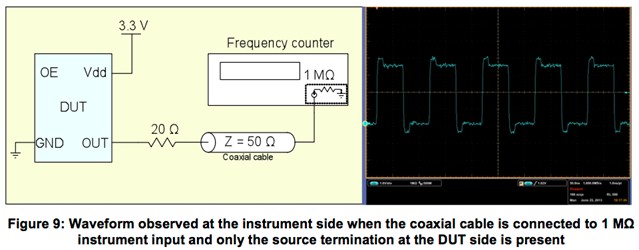
晶振的作用是结合单片机内部电路，产生单片机所需的时钟频率。单片机晶振提供的时钟频率越高，那么单片机运行速度就会越快。系统所有指令的执行都是以晶振提供的时钟频率为基础。

晶振通常分为无源晶振和有源晶振两种类型：无源晶振，英文为Crystal，中文全称为石英晶体谐振器，而有源晶振英文则叫做Oscillator，中文全称为石英晶体振荡器。

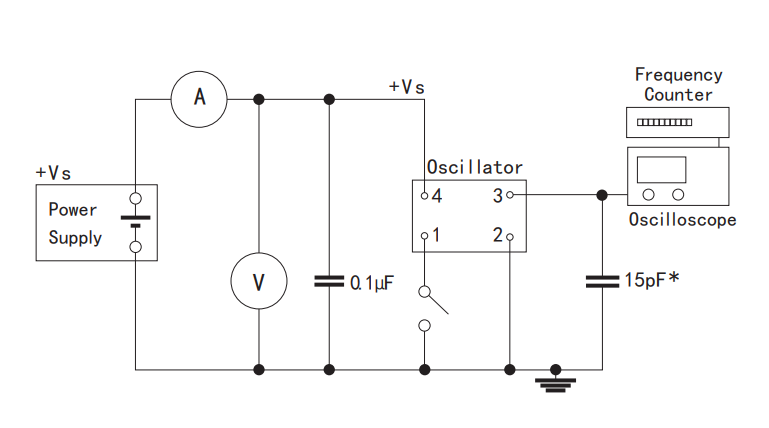
有源晶振是一个完整的谐振振荡器，利用石英晶体的压电效应来起振，所以有源晶振需要供电。当我们把有源晶振电路做好后，不需要外接电路，它就可以主动产生振荡频率，并且可以提供高精度的频率基准，信号质量比无源信号好。

****无源晶振和有源晶振的焊盘脚位区别：****

****普通有源晶振通常有4个引脚,VCC,GND,晶振输出引脚和一个悬空引脚。****

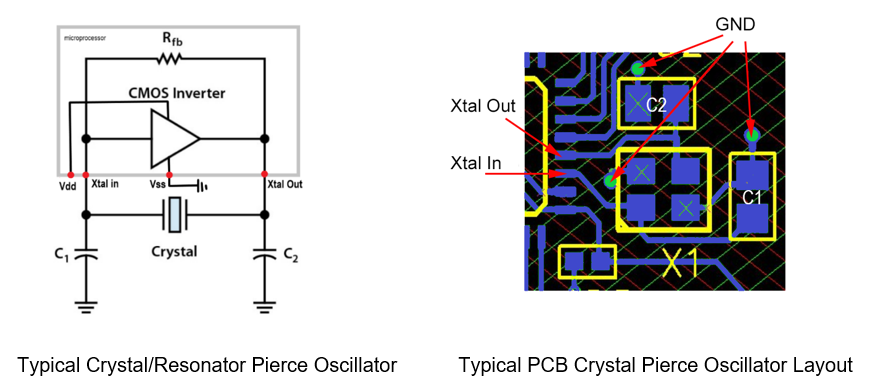


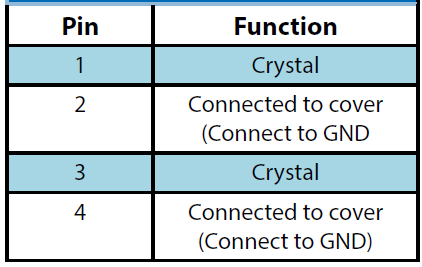
****有源晶振测试电路如下：****



****关于无源晶振的常识****

若为2引脚，分别为频率输入脚和频率输出脚，不具备方向性，不用担心接反；若为3引脚，中间引脚已与晶振外壳连接，接地（GND）即可，两侧引脚同2引脚晶振一样，不具备方向性。若为4引脚，脚1与脚3为频率输入脚和频率输出脚，不具备方向性，不用担心接反。脚2与脚4一般与晶振外壳连接，可接地（GND），也可悬空处理。



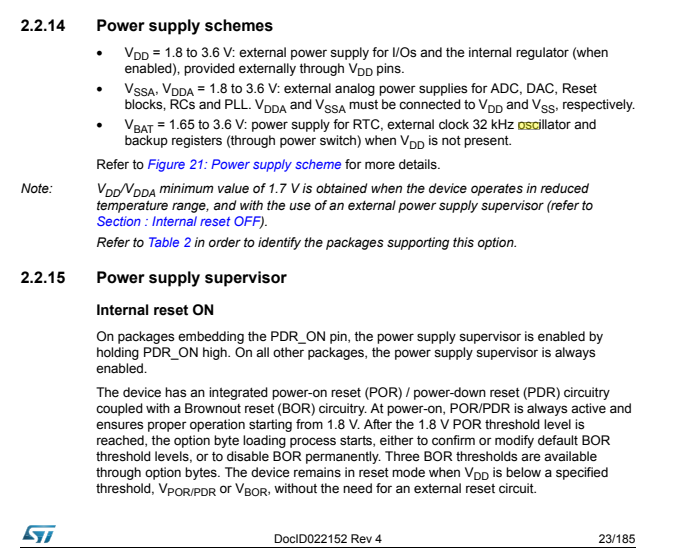


****无源晶振和有源晶振接法基本原理****：

无源晶振：用单片机上的两个晶振引脚接上去即可（频率输入脚和频率输出脚）

有源晶振：只接到单片机的晶振的输入引脚上,输出引脚无需连接

# 电源电路



这样通常是3.3V为更好的

通常来说我们都是用LDO线性稳压器

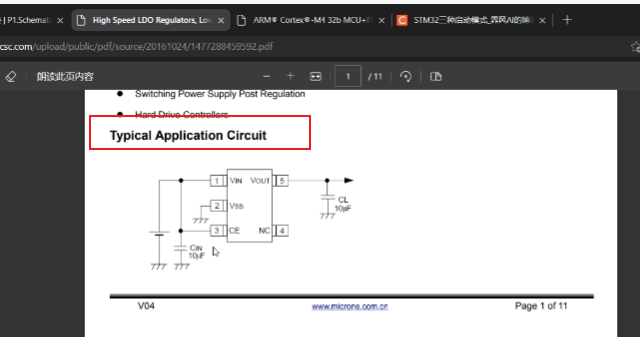


通常我们使用1117，

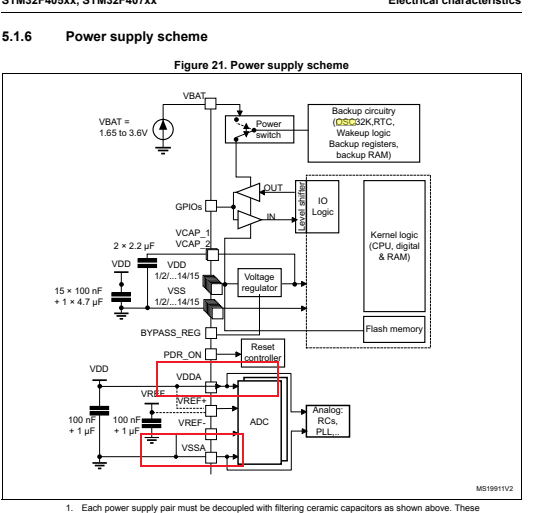
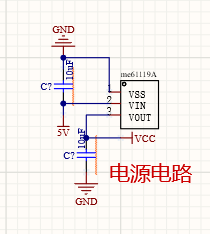
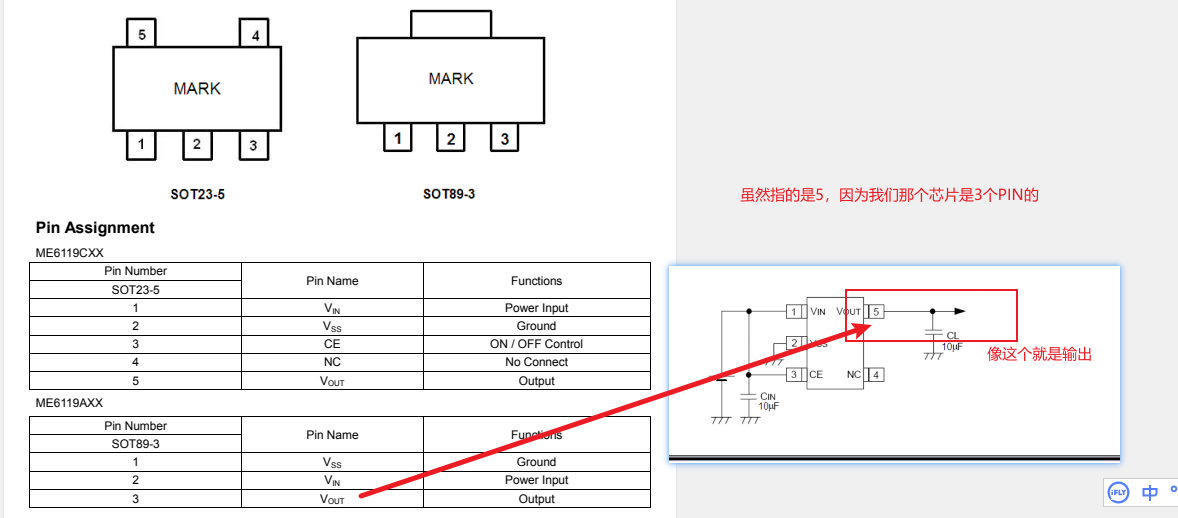
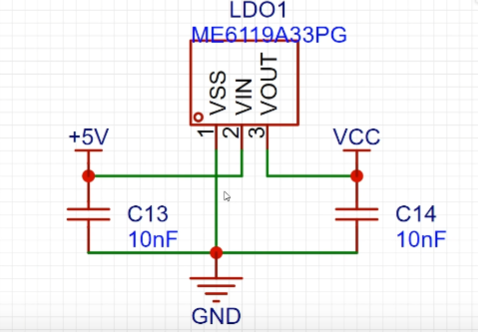


和ME61

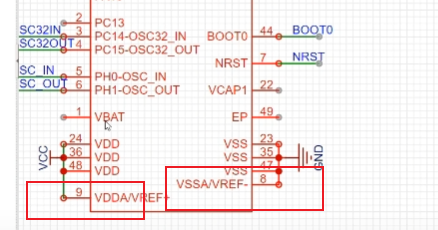


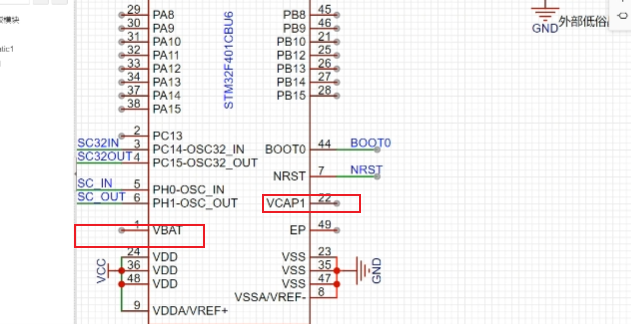


不能瞎画，我们 根据这个芯片的手册来画

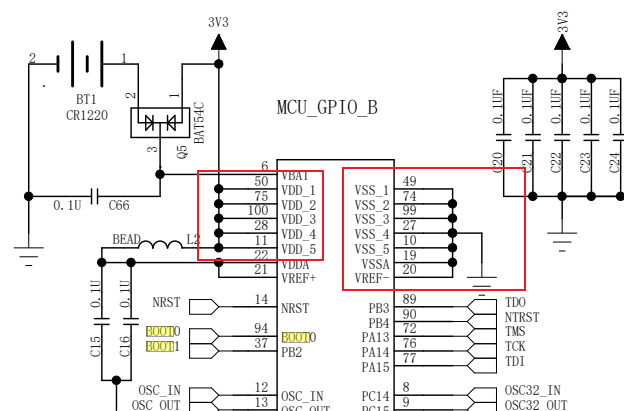
电容我们选择的是0603规格的这个尺寸小一点

vdda就是涉及到ADC相关的东西

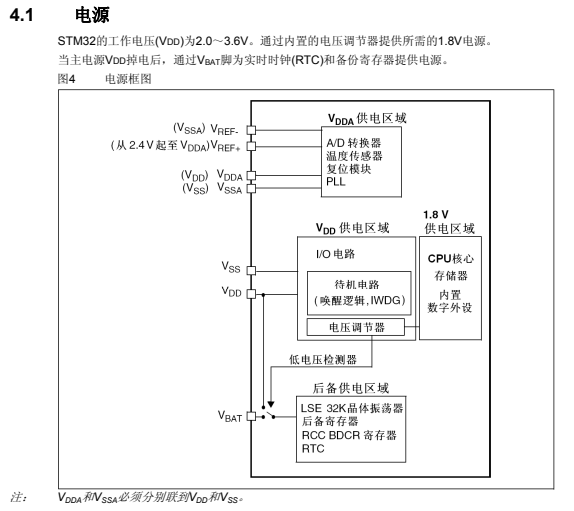




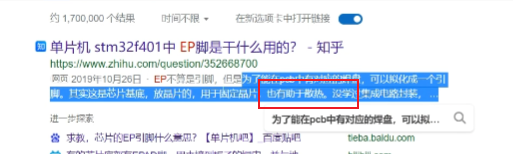
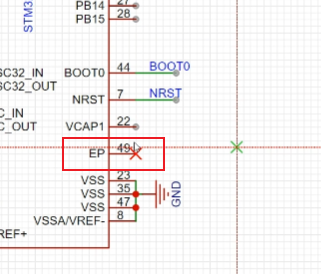
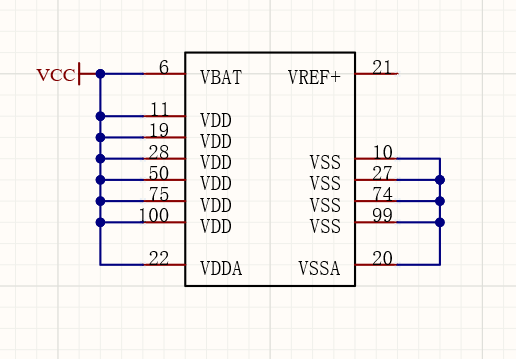
像这些我们就再去手册中找,上去根据



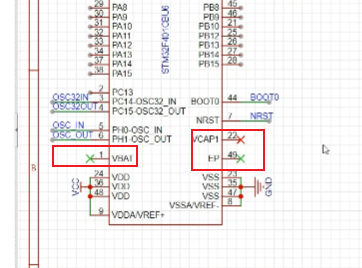
野火是这么接的，



由于这个vdda和vdd要求的都是1.8-3.6之间我们就可以给它一起接到我们的vcc 3.3v那里



这个EP脚是用来散热的



在最小系统里这vBat 和 vcap 都是不用的

# ----------------按上面的画，就完成了，----但是别的文章又完善了它---上面的部分就已经是最小系统了，

<https://blog.csdn.net/weixin_52531699/article/details/137430301>

STM32核心版PCB设计及总结

RouDragon

已于 2024-04-23 12:06:58 修改

阅读量1.1k

收藏 35

点赞数 29

文章标签： stm32 嵌入式硬件 单片机

版权

本人写这篇文章主要是对核心版PCB设计的总结，是基于STM32F407VET6进行开发，供开发者们借鉴与学习，本人水平有限，如有不足之处请指正。

首先本人ChatGPT大致了解了一下STM32需要的电路，STM32核心板设计电路主要包括以下几个部分：

1.电源转换电路：这部分电路主要是将5V输入电压转换为3.3V输出电压。

2.JTAG/SWD调试接口电路：这部分电路采用了标准的JTAG接法，这种接法兼容SWD接口，因为SWD只需要四根线（SWCLK、SWDIO、VCC和GND）。JTAG/SWD调试接口电路可以对STM32微控制器进行在线调试。对BOOT电路讲解也在此部分。

3.独立按键电路：STM32核心板上有三个独立按键，分别是KEY1、KEY2和KEY3，每个按键都与一个电容并联，且通过一个10K电阻连接到3.3V电源网络。因此，按键未按下时，输入到STM32微控制的电压为高电平，按键按下时，输入到STM32微控制的电压为低电平。

4.高速外部晶振电路和低速外部晶振电路：STM32微控制器具有非常强大时钟系统，除了内置高精度和低精度的时钟系统外，大家还可以通过外接晶振，为STM32微控制器提供高精度和低精度的时钟系统。

5.LED电路：除了标识为PWR的电源指示LED外，STM32核心板上还有两个LED，LD1为蓝色，LD2为绿色，每个LED分别与一个330Ω电阻串联后连接STM32F407VET6芯片的引脚上，在LED电路中，电阻起着分压限流的作用。

6.STM32微控制器电路：这是STM32核心板的核心部分，由STM32滤波电路、STM32微控制器、复位电路、启动模式选择电路组成。STM32核心板上的STM32F407VET6连接外部的32.768KHz晶振，8MHz晶振等。

以上就是STM32核心版设计电路的主要组成部分。

一.电源转换电路：

本人电源电路设计的是3个方面

F407电源设计及功能引脚将以上英文翻译得

•VSA, VDDA = 1.8至3.6 V:用于ADC, DAC，复位块，RC和PLL的外部模拟电源。VDDA和VSA必须分别与VDD和VSS相连。

•VDD = 1.8 ~ 3.6 V: I/O和内部稳压器(使能时)的外部电源，通过VDD引脚对外供电。

•VBAT = 1.65至3.6 V:当VDD不存在时，为RTC，外部时钟32 kHz振荡器和备份寄存器(通过电源开关)供电。

注:VDD/VDDA最小值为1.7 V，当器件工作在降低的温度范围内，并使用外部电源管理器

1.根据以上可知模拟电源供电引脚为VDDA，负端为VSA。VSA, VDDA支持宽范围电压输入的 。

2.VDD是F407内部供电引脚，VBAT的作用类似于备用电源，可以不接3.3V但是原则上接好电源最优。

参考官方提供的电源设计图，一点点设计。1.VBAT输入的电压为1.65-3.6V,一般通入3.3V。2.每个电源VDD对必须用滤波陶瓷电容器去耦。这些电容器必须尽可能靠近或低于PCB底部的适当引脚，以确保器件的良好功能。3. 4.7µF陶瓷电容必须连接到其中一个VDD引脚上。4.VDDA=VDD 、 VSSA=VSS引脚 5.在VCAP上分别接入2.2uf的电容 6.VREF+接VDDA，VREF-接VSSA 7.VREF+处接100nf+1uf并联，VDDA处接100nf+1uf并联切记小电容（100nf）离芯片引脚更近。

根据以上要求设计原理图

图中VBAT、VDDA至+3.3V之间为短接符。

模拟3.3V电源

5V转3.3V电压设计

本人这部分用的是线性稳压器(LDO)，线性稳压器（LDO）是一种低压差线性稳压器，它使用在其线性区域内运行的晶体管或FET，从应用的输入电压中减去超额的电压，产生经过调节的输出电压。LDO的英文说法是:Linear regulator，中文一般称为LDO。LDO中，实现调压功能的一个关键的组件是一个调整管，它就像是一个可变电阻串在输入和输出之间，当输入电压或是负载电流发生变化时，受误差控制电路驱动的调整管可以及时的调整其呈现出来的电阻量，使得负载端的电压总是处于恒定的状态。在资料链接中提供芯片手册。

如图选用XXX-3.3型号也就是线性稳压器输出在3.3V左右。

图中-ADJ结尾的是可以通过电阻的配置更改电压输出的线性稳压器，本人使用的是CJA1117B-3.3。

参考电路主要是第二个，配置相应的电容即可

最终电源电路如图。注意你的TVS（TVS管作为浪涌静电），焊接时注意正负，这个家伙标志很小！！！

参考这位大佬的文章你就会明白TVS了，不用谢我只是大佬的搬运工

链接: https://blog.csdn.net/u010614434/article/details/105338417

二.JTAG/SWD调试接口电路

Arm SWJ-DP接口是嵌入式的，是一个组合的JTAG和串行线调试端口，可以将串行线调试或JTAG探针连接到目标。仅使用2个引脚而不是JTAG所需的5个引脚执行调试（JTAG引脚可以重复用作具有替代功能的GPIO）：JTAG TMS和TCK引脚分别与SWDIO和SWCLK共享，TMS引脚上的特定序列用于在JTAG-DP和SW-DP之间切换。

对烧录引脚没有详细的介绍直接排针引出即可。

boot电路设计

BOOT0和BOOT1引脚用于控制启动模式和引导加载程序。这两个引脚的电平状态决定了芯片复位后从哪个存储器区域开始执行程序。

翻译译文：在启动时，引导引脚用于从三个启动选项中选择一个:

•从用户Flash启动

•从系统内存引导

•从嵌入式SRAM启动

BOOT0=1,BOOT1=0代表从系统存储器启动，而BOOT0=0,BOOT1=0则是最常见的用户闪存启动模式，这是设备正常工作的模式。主要对从嵌入式SRAM启动这个模式讲解，当BOOT0和BOOT1都被设置为1时，STM32F4将从内嵌的SRAM启动。这种启动模式通常用于调试目的，因为它允许程序直接下载到RAM中，这对于一些调试操作来说是非常有利的。然而，在正常生产环境中，我们通常会选择默认的用户闪存启动模式，这是因为STM32F4的FLASH可以擦出10万次，这是一个非常大的数值，足以满足大部分应用的需求。

嵌入式SRAM启动优势

调试便利：由于SRAM的速度非常快，对于一些小型的调试任务来说，直接将程序下载到RAM中可以大大加快调试速度。

安全启动：在某些应用场景中，我们需要确保设备只执行我们想要的代码。通过将BOOT0和BOOT1都设置为1，我们可以确保设备从RAM中加载固件，而不是从用户闪存或系统存储器中加载。

然而，值得注意的是，将BOOT0和BOOT1都设置为1并不是在所有情况下都是最佳选择。在大多数生产环境中，我们仍然倾向于使用默认的用户闪存启动模式BOOT0=0,BOOT1=0。只有在确实需要进行快速调试或需要实现安全启动的情况下，才会考虑使用SRAM启动模式。

综上所述，STM32F4 BOOT电路的一般设置是BOOT0=1,BOOT1=1，主要是为了适应特定的调试和安全启动需求。BOOT电路原理图如图所示，上拉电阻BOOT引脚一般工作状态为高电平，图中用了两个排针引出通过跳帽改变BOOT引脚电平状态。

三.独立按键电路

按键电路本人设计的是两大部分，复位以及自定义功能按键。

首先需要对复位有所了解。

结合图片1，2可知复位NRST是通过按键外部低电平复位。芯片引脚内部是有上拉电阻RPU阻值在40K左右，外部接一个0.1uf电容滤波即可。复位网络保护器件免受寄生复位的影响。用户必须确保NRST引脚上的电平低于表51中规定的VIL（NRST）最大电平0.3VDD=0.3\*3.3=1V。否则，设备不考虑重置。

我借鉴了许多人的核心板设计电路，在此过程中本人遇到一个问题为什么F407复位引脚一般会接一个外部的上拉电阻，内部不是有40K上拉电阻吗？

答案：在STM32F407芯片中，复位引脚（NRST）具有内部的40K上拉电阻。但是，有些设计中仍然会外接一个上拉电阻，这是为了增强复位电路的可靠性和稳定性。外接的上拉电阻能够确保复位引脚在上电时确保有效地被拉高至逻辑高电平，从而避免一些可能出现的电路漂移或噪声干扰对复位信号的稳定性产生影响。因此，即使具有内部上拉电阻，有些设计仍会选择外接上拉电阻来提高复位引脚的可靠性。

问题二为什么是10K不是40K呢？

答案：在STM32F4微控制器的NRST（复位）引脚上使用10K电阻而不是40K电阻可能会对系统的复位性能产生影响。NRST引脚是用于控制系统复位的引脚，当引脚处于高电平时系统处于正常工作状态，当引脚被拉低时系统会执行复位操作。

使用10K电阻的主要优势是它提供了更快的上升时间，因为在NRST引脚被释放时，与VDD连接的电容会更快地充电。这意味着当复位信号被取消时，系统可以更快地从复位状态恢复到正常工作状态。

另一方面，40K电阻会导致NRST引脚的上升时间更长，这可能会延迟系统从复位状态恢复到正常工作状态。在某些应用中，这种延迟可能是可以接受的，但在其他情况下，特别是对于要求快速响应的应用，较长的上升时间可能会导致问题。

因此，选择10K电阻或40K电阻取决于系统的复位要求以及对系统响应时间的需求。

所以理论上1K也行，一般外部会接入一个10K上拉电阻提高响应速率。

如图所示为复位电路图

用户按键电路

也就是通过引脚高低电平控制并执行相应的特殊任务。原理图如图所示。

四.高速外部晶振电路和低速外部晶振电路

高速外部晶振设计

关于晶振电路设计规则，晶体/陶瓷谐振器产生的高速外部时钟高速外部（HSE）时钟可由4至26 MHz晶体/陶瓷谐振器振荡器提供。本段中给出的所有信息均基于表32中规定的典型外部元件获得的表征结果。\*\*在应用中，谐振器和负载电容必须尽可能靠近振荡器引脚放置，以最大限度地减少输出失真和启动稳定时间。\*\*有关谐振器特性（频率、封装、精度）的更多详细信息，请咨询晶体谐振器制造商。

Rext值取决于晶体的特性。(官方手册没有详细介绍这个电阻的设计所以其取值问题是我不明白的问题有大佬的话请留言指导一下哈)

参考设计中文译文：对于CL1和CL2，建议使用5 pF至25 pF（典型值）范围内的高质量外部陶瓷电容，设计用于高频应用，并根据晶体或谐振器的要求进行选择（见图32）。CL1和CL2通常大小相同。晶体制造商通常指定负载电容为CL1和CL2的串联组合。在确定CL1和CL2的大小时，必须包括PCB和MCU引脚电容（10 pF可用作引脚和板电容组合的粗略估计值）。

根据以上设计规则本人电容选用22pf，8MHZ晶振，高频晶振电路如图所示。

低速外部晶振设计

低速外部(LSE)时钟可以提供一个32.768 kHz晶体/陶瓷谐振振荡器。

电容设计要求请遵守高频晶振设计要求。根据以上设计规则本人电容选用12pf，32.768kHZ晶振，高频晶振电路如图所示。

五.LED电路

LED电路主要是两部分电源指示灯以及引脚控制灯，具体不介绍了直接上原理图。

六、其它电路

其它电路1.排针IO引脚引出电路

其它电路二TypeC接口电路

我之所以想提及一下这个电路是因为，第一次打板误以为TypeC只要引入电源接口即可。后来才发现不对不能正常供电。供电需要了解一下CC协议。

推荐一下两位大佬的文章:

链接1: https://blog.csdn.net/AirCity123/article/details/103554177

链接2: https://blog.csdn.net/one\_\_\_\_rookie/article/details/132859403

认真学习过这两篇文章后，你就会明白，充电从端我需要配置成UFP——Upstream Facing Port，也就是Device模式，具体我就不细讲了请参考大佬文章。

TypeC电路如图所示。

以上介绍就是核心版的重要电路讲解，接下来测试核心版。

8兆晶振测试通过

32KHZ晶振测试有问题我怀疑是焊接问题，只能用内部晶振了

串口测试成功

按键控制红色LED灯成功

注意烧代码的时候BOOT0,BOOT1得接GND接好跳帽。

个人小结

本人焊了3块板子，第一块VBAT没接3.3V不能烧代码，第二块晶振电路焊接有问题，没有接BOOT引脚第三次终于成功了。

嘉立创PCB设计链接: https://oshwhub.com/roudragon/stm32f407vet6-he-xin-ban-she-ji

百度网盘存放的是数据手册以及测试时的代码

百度网盘链接: https://pan.baidu.com/s/1Os8tppub4f810\_kKUejz5Q?pwd=0h3n

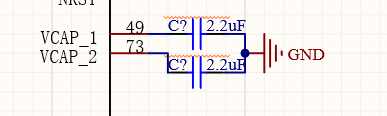
提取码：0h3n

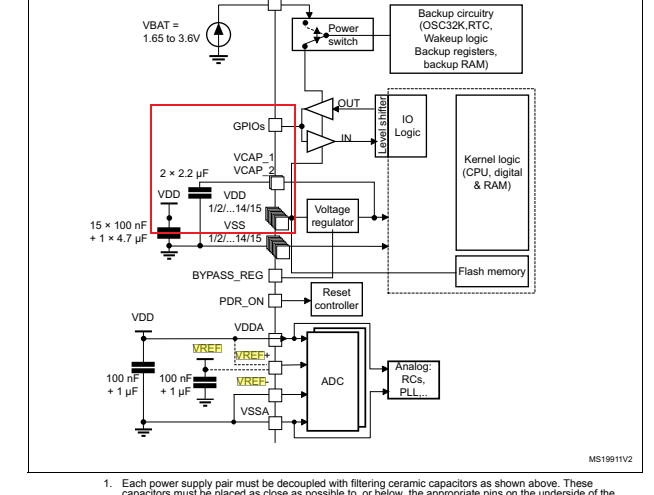
希望这篇文章对你有所帮助，有不足之处本人水平有限请谅解

————————————————

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

原文链接：https://blog.csdn.net/weixin\_52531699/article/details/137430301





5、短接符

立创EDA专业版不支持在同一条导线上同时设置或者放置两个不同的网络名。

可以使用短接符把两个不同的网络连接在一起。

1.顶部菜单 - 放置 - 短接符；

2.顶部快捷栏 - 短接符图标；

短接符没有快捷键，可以通过以下方式设置：

在顶部菜单栏找到设置，点击下拉菜单中的快捷键，在此可以搜索设置所有的快捷键，可以根据需要设置短接符的快捷键

