[这位大神](https://link.jianshu.com?t=http%3A%2F%2Fakb77.com%2Fg%2Fstm32%2Fjlink-ob%2F" \t "_blank)，从JlinkARM.dll 里面硬生生把 STM32F072 的Jlink OB 固件给拔出来了。

可惜他的帖子里面提供的文件，是从5.12版本的 JlinkARM.dll，里面拔出来的，现在最新的到了至少 6.20， 我的 keil 里面是 6.10j，每次链接都会提示固件升级，神烦。

而且在我的板子上，出现了点击取消升级后，板子掉线的情况。

**拿 JlinkARM.dll 开刀**

于是我打算自己尝试把固件拔出来。方法跟他的一样，从 JlinkARM.dll 下手。

他本人已经在帖子里面回复了最新的 6.10n 这个版本的 JlinkARM.dll 的固件位置

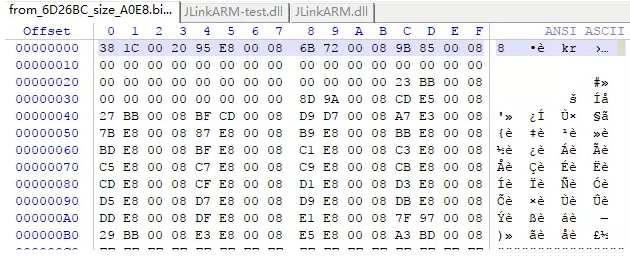
Can't say about 610i but for 610n start from 0x8C8054 and length 0x0B66C

不过我没找到这个版本的 JlinkARM.dll。我打算拿自己 keil 带的 6.10j 的dll来尝试定位 STM32 的固件位置。

**开始挖坑**

需要的工具是 WinHex

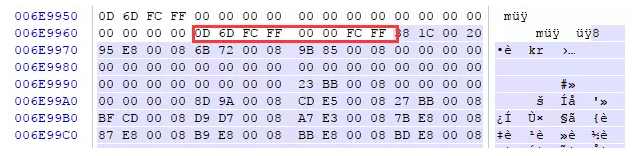
首先我尝试打开从他博客下载的固件，我的设想是，在我的 JlinkARM.dll 里面找看看，有没有一样的至少前面几个字节。比如这几个，0x38 1C 00 20

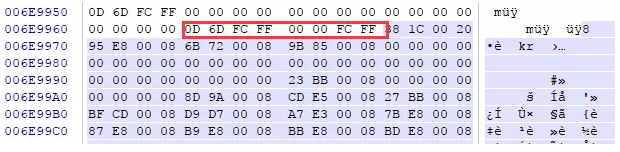


结果并没有。我又试试找了其他的字段，并没有什么线索。

我反思了一下，既然固件是有可能变化的，那我还是尝试一下在博客主用的 5.12j 的 JlinkARM.dll 里面先定位一下固件。

于是我把 5.12j 下了下来，打开后果然定位到了固件的位置。这个版本的固件，是 38 1C 开头的。

但是要怎么在其他版本的 JlinkARM.dll 里面定位呢？依旧用我手上的 6.10j 版本的 JlinkARM.dll 试一下。我注意到了固件之前，也就是 38 1C 之前的这几个字节。先不管他们有什么意义，先在 6.10j 里面搜搜试试。

哈哈哈，果然给我找到了，难道固件就是后面 4A 25 00 EA 开头的么？ 现在还不好确定。先假设这里就是开始位置，那么长度是多少？

于是我又回到 5.12j 里面，参考一下原博，找找是否有长度信息。

原博对 5.12j 版本的描述

1. Extract binary part from JLinkARM.dll (from JLink\_V512)  
   For example from offset 7153340 (0x6D26BC) and size 41192 (0x0A0E8).  
   Use any binary extraction tool as you prefer.

现在有两个思路，

1. 找到 5.12 里面的 0x0A0E8，看看这个长度信息的位置有什么线索。
2. 找到固件的结尾，看看是否是一个固定的特殊格式

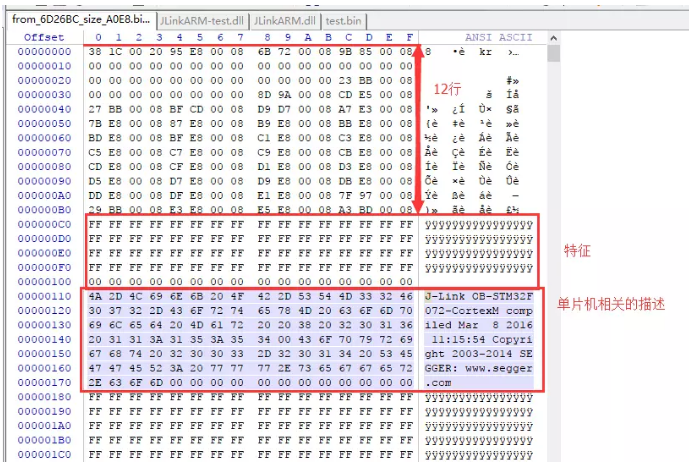
**打住！以上是错误的。**上面那个图，仔细看的话，就会发现，右边的描述里面，写着，这个固件是给 SAM7 用的，而不是STM32！！！ 而且，博主并不是用 5.12j 的 JlinkARM.dll 开刀的！ 偏移地址根本不对。 慢慢的坑。

**填坑**

虽然上面找到的代码，并不合适 STM32F072，但是！我找到了线索和规律，

**线索就是，直接搜索单片机的型号！**  
**规律就是，不同芯片的固件，是连续放置的！**

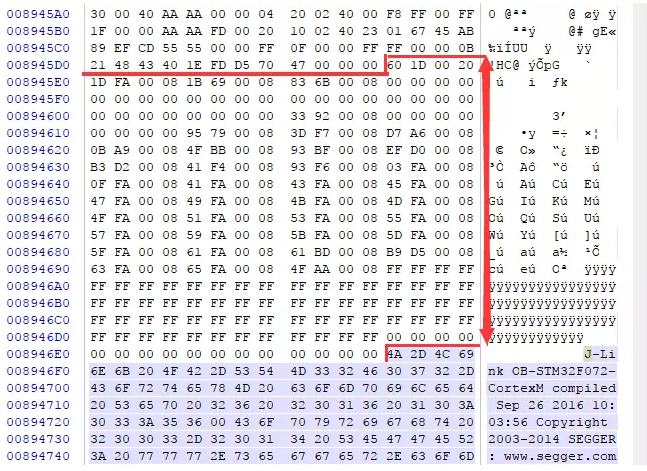
先看线索，在博主提供的 5.12 版本里面提取的固件中，有这样一个规律。

单片机相关的描述之前，有一些 64个字节的 FF 和 16个字节的 00，这些明显是保留的块，具体用来干嘛的我不懂，我也不想理。然后之前是 12 \* 16 个字节的数据，我印象中，这个地方应该是放置了中断跳转的函数指针，具体不确定。

但是，他们的长度是固定的，也就是说，找到单片机的描述，再往前推一定数量的字节，就是固件开始地方！

具体是多少个字节呢，17行 \* 16 = 0x1FA 个 （506个）。

好了，那我们就去其他版本的 JlinkARM.dll 里面搜，这里以我手上的 6.10j 版本为例。



很轻松就找到了关于 STM32F072的描述，然后根据 保留的 FF 和 00， 往前推 12 行， 看到一个“60 1D 00 20” 开头的字段。看到的第一眼，我就很肯定这就是固件的开头了！

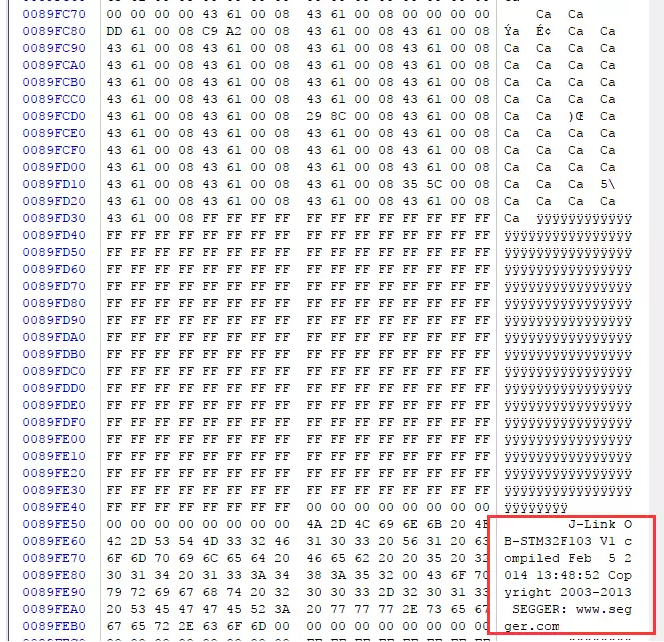
因为这个跟博主提供的 5.12 版本的固件太像了 “30 1C 00 20”。最后两个字节一模一样。“30” 和 “60” 在二进制上，只移位了1位，然后 1D 就很简单的是 1C+1.

**找到开头了，可喜可贺，那么固件的长度是多少呢？找不到固件的长度，裁剪不出来，有个毛用？**

然而，我不需要知道固件的长度。因为固件的存放，是连续的，这是我在浏览上下文发现的一个规律。我只需要知道下一个固件什么时候开始，然后大概裁剪一下就好了。原则是只能多裁剪，不能少裁剪。

有人会担心，如果多余的字节被裁剪进去了，会不会影响程序，答案是否定的。为啥自己去考虑考虑。

于是我开始寻找下一个固件开始的位置，怎么找？直接用手翻。也就 40+k 的固件大小，翻几十页就好了。鼠标点几下还是很快的。

嘿嘿，下一个居然是 STM32F103 的 Jlink OB 固件。

到这里其实可以结束了，想省事，直接裁剪到那堆 FFFF 里面随意位置就好，因为显然这个地方已经是另一个固件开始的地方了。符合“只多裁，不少裁”的原则。当然我是按 32bit 对齐裁剪的。新建一个文件，把固件开始位置，到结束位置复制过去，然后保存为 .bin 就可以了。

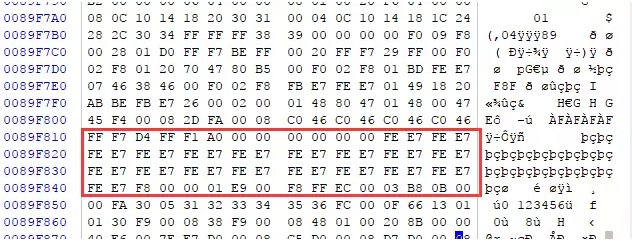
下载，测试，bingo！固件显示已经是 2017年5月的固件，比我的 jlink 驱动还新。这回不会再提示更新固件了吧

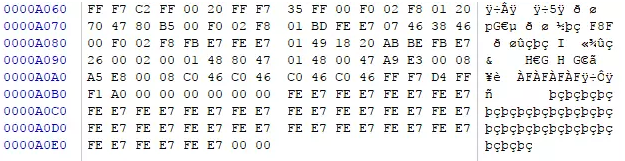
**但是如果我想知道固件准确的结束位置，怎么办**

**再挖一个坑**

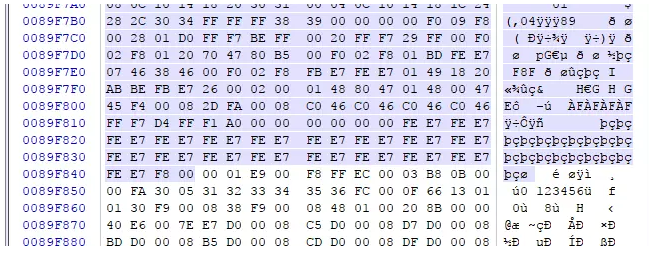
仔细看了一下 F103 的固件，保留的 FF 区域的大小，跟 F072 固件完全不同，那我就很难确定了。当然，F103 的 Jlink OB 固件满天飞，去网站上下一个，用相同的方法查看 F103 固件的偏移量，就大概能知道是什么地方开始的了。但是，知道开始的位置并没有什么用，虽然 JlinkARM.dll 里面固件看起来是连续存放的，但是固件之间可能会有一些其他的描述符。如果想去除这些，必须找到真正的固件结束位置。

然后我就继续往上翻了。 滚了几个滚轮之后，马上看到了熟悉的 pattern。 一大堆 "0xFE 0xE7"

对比一下 5.12 JlinkARM.dll 提取的固件结尾：

非常相似，所以呢，大概就是这个地方了。

然而并不是这样。。。反正这个地方失败了

保存，测试，失败。

**坑不填了**

就这么先用着吧。

因为固件不完美，最新的 Jlink OB STM32F072 的 bin 文件 （来自 6.20e） 就不放了，如果有需求清留言。

看完以上，需要的人应该已经能自己定位固件的位置自己提取最新版本的固件了。  
ST 的 DFU 软件也是各种各样的坑，祝好。

**关于固件自动升级的问题**

**答案是：不能**  
以上固件在单片机的 flash 中，是从 0x8004800 开始的，从 0x8000000 开始，是 bootloader。

问题在于，博客里面提供的 bootloader 并不是 jlink 写的 bootloader。博客里面能下载的bootloader，我查看了一下，非常短。我推断它只提供了很简单的跳转到 0x8004800 的功能。

这就导致了 Jlink 的驱动，无法升级固件。没授权下拿到 Jlink 提供的 bootloader 的可能性极小。

如果对你有帮助，请多支持我的博客： [Aircheese Studio](https://link.jianshu.com?t=http%3A%2F%2Faircheese.me) 谢谢~

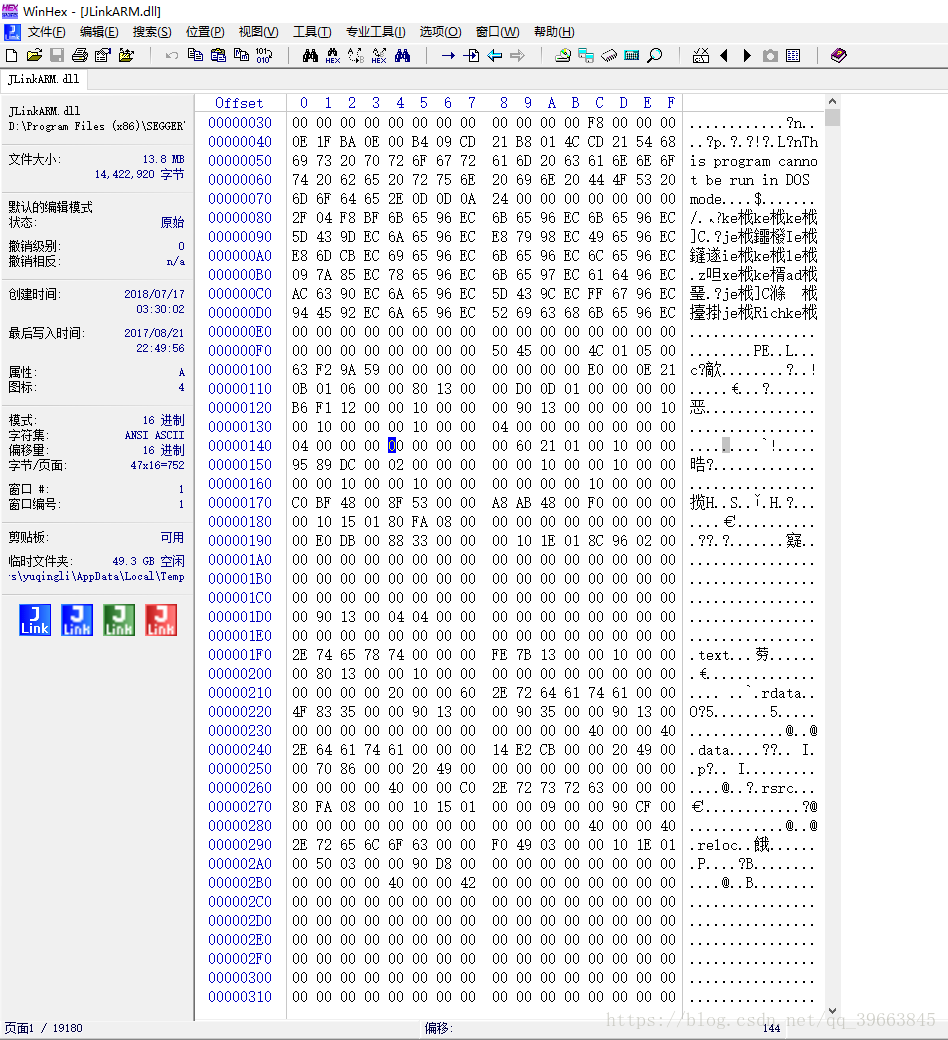
作者：毛小嘉  
链接：https://www.jianshu.com/p/0cd3a513f1b1  
来源：简书  
简书著作权归作者所有，任何形式的转载都请联系作者获得授权并注明出处。

开发产品一定离不开的是调试器，现在国内市场上大多用的是盗版的j\_link ob和st\_link。这两个调试器diy的可能性很大，我见过二合一的调试器。但是世面上的固件一般都是老版本的，在用新驱动的时候容易识别盗版导致不能用，虽然可以将驱动版本调低来解决。本文就是介绍如何提取最新的固件。

首先来看一下jlink ob。用到的文件是驱动安装目录下的JlinkARM.dll这个文件（这里以比较新的驱动版本6.18C为例）。这个文件里面包含了arm内核的所有固件，不止是jlink ob，不过其他的我没实验。还用到的软件是winhex。准备好下面开始提取固件。

第一步，用winhex打开JlinkARM.dll。

如图：

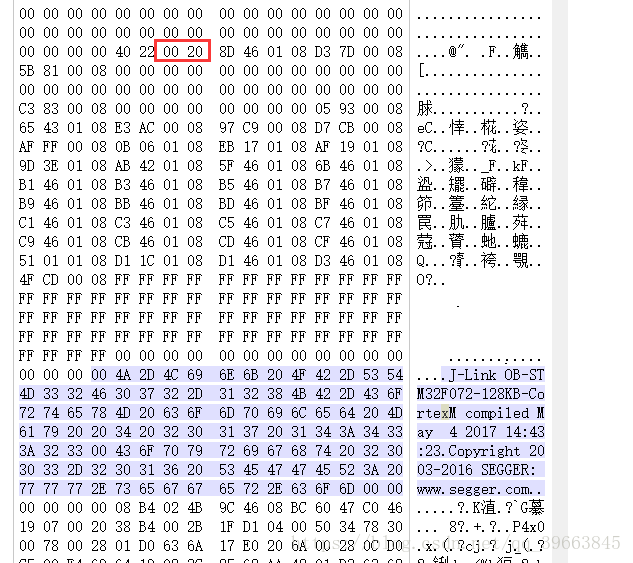


第二步，寻找固件位置。这里我就不卖关子了，直接用搜索功能 向下搜索stm32（ob的ic是stm32的）。由于stm32关键词比较多，固件位置一般靠下，可以鼠标向下适当拉一下在向下搜索。



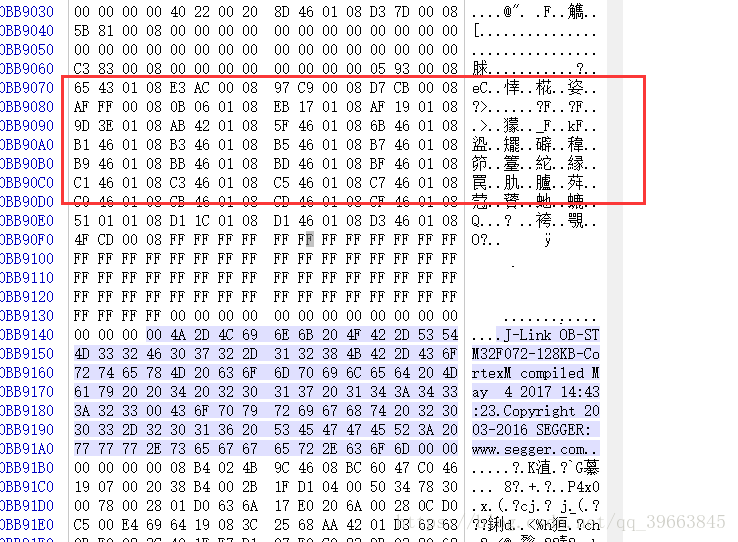
我搜索到的第一个内容，可以看到这个固件的编译日期是2017年 用的ic是stm32f072。

找到固件的大题位置了怎么找到开头那？我就值说了，为什么就不说了，在这里向上搜索16进制字符0020（注意是16进制）。



找到了这个，首先是0020的位置一般是偶地址开始的（比对其他固件没发现不是）。

还有就是后面不愿处跟着一大堆有规律的数据，什么规律我也不知道，但是能看出来特殊，像这样



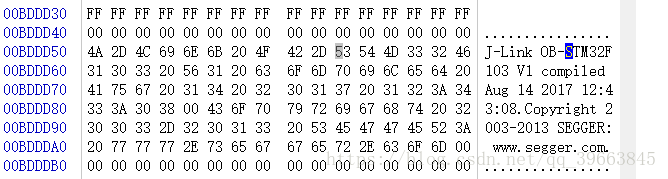
看着差不多就行了。然后从0020在往前推2个字节，比如这个也就是40 22 00 20；40就是固件的开始，开始找到了，那么怎么找到结尾那。这个比较简单，因为你多取没事，但是不能少取；最极端的方法就是看看你jlink ob用的ic的flash多大，就取多大（注意，因为固件前面还有一段bootloater，用来更新固件用的，所以要减去这个长度，这个bootloater后面会说）。

第二种方法就是找到下一个固件的开始，然后截取，不一定这么精确，多取一点没事。

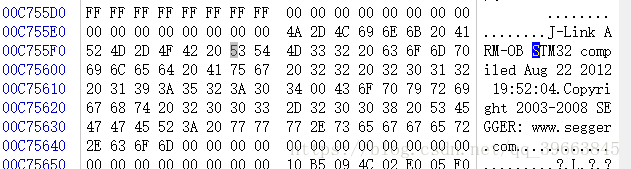
把着一段内容复制出来，用winhex新建一个文件，粘贴过去。这个文件就是你提取的固件了，比如这个是2017年编译的已经非常新了，再也不怕报错了。

市面上的jlink ob大多是stm32F103的，这个固件不适合用，读者们自己接着往下搜索，就可以找到stm32F103的固件，提取方法时一样的。

F103的固件有两个：



这个编译时间比较新2017年，不知到时什么版本的，我烧写到我的jlink ob（stm32f103c8）里面时可以用的，但是好像指示灯和我手上的不一样了，不影响使用。但是最高支持2250K的频率。



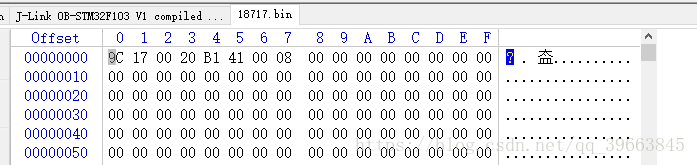
这个版本在我的jilin ob（stm32f103c8）指示灯正常，显示最高频率4000K。（但是我经过测试两个固件的烧写速度一样，设置1M 2M 3M 5M烧写速度全部一样）。

你以为完了吗？当然没有，上面提到过前面还有一段用来更新固件的bootloater，这个bootloater的大小有0x4000（f103c8的大小，其他不知道）的大小并且在jlinkarm.dll里面找不到。怎么办？

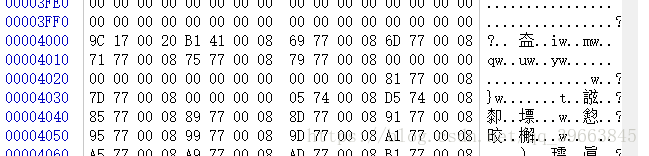
有两种办法解决：

不用这个bootloater直接跳过，但是也要经过一些处理。在你提取出来的固件前面用winhex添加0x4000大小的空字节，充填数据全是0x00就行了，然后改充填完0x00的固件前面8个字节和提取出来的固件前8个字节一样

比如我的



改了两个，没影响。



这个是真正固件的开头（无改动，注意偏移地址在stm32f103c8上是0x4000）。

这样把改完的固件烧写到stm32F103c8里面就行了，至于为什么这样改这里就不多讲了，内核决定。但是这样处理是不能用官方软件更新升级固件的，因为根本没有bootloater

第二种处理方法是，找一段官方的bootloater加在固件前面，这个bootloater我在下面提供下载（但是我认为没必要，因为本来就是盗版）

下载资源含有两个完整的jlink ob固件；适用ic：stm32f101c8/cb stm32f103c8/cb。一个bootloater

下载链接：https://download.csdn.net/download/qq\_39663845/10547285

注：世面上用的ic有的是stm32f101c8/cb stm32f103c8/cb都是通用的。

大家可能会问101不是没有usb功能吗？其实是硬件是有的，但是出厂没测试，或者是残次品。

也就是说你可以用但是st不售后，出了问题别找我。

还有就是f103c8的flash其实有128K，和上面一样也是没测试。stm32f101c8/cb stm32f103c8/cb这4个感觉就是一个东西不同名而已

---------------------

作者：不起名

来源：CSDN

原文：https://blog.csdn.net/qq\_39663845/article/details/81086499

版权声明：本文为博主原创文章，转载请附上博文链接！

上次写了个从JLinkARM.dll提取 f103版本的ob固件的帖子https://blog.csdn.net/qq\_39663845/article/details/81086499。

最近有空做了个stm32f072版本，我吧GPIOB 的io全部引出，而且画上了外部晶振（jlink用不到外部晶振），之所以加这些，是为了这个板子同时可以当做开发板使用。板子面积大约是3.2\*5的样子。

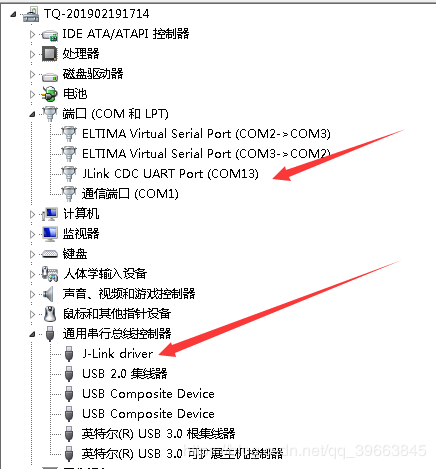
072版本比f103的优势就是成本更低，而且带串口，支持swo。缺点就是swd只有2M。

固件的提取和上面的103方式一样，我没找到 loaderboot ，处理方式见上面的帖子，很简单。

值得注意的是072版本的固件和103版本的loaderboot大小不一样。所以真正固件的偏移地址也不一样，072的偏移地址是0x4800。

还有一点要注意，前面的bootloader 没有一定要充填 0xFF，不然jlink的sn号可能会报错。

下面上几张图，最后提供ad版本的原理图、pcb图和6.22驱动提取的固件。



https://download.csdn.net/download/qq\_39663845/11109032

---------------------

作者：不起名

来源：CSDN

原文：https://blog.csdn.net/qq\_39663845/article/details/89225694

版权声明：本文为博主原创文章，转载请附上博文链接！