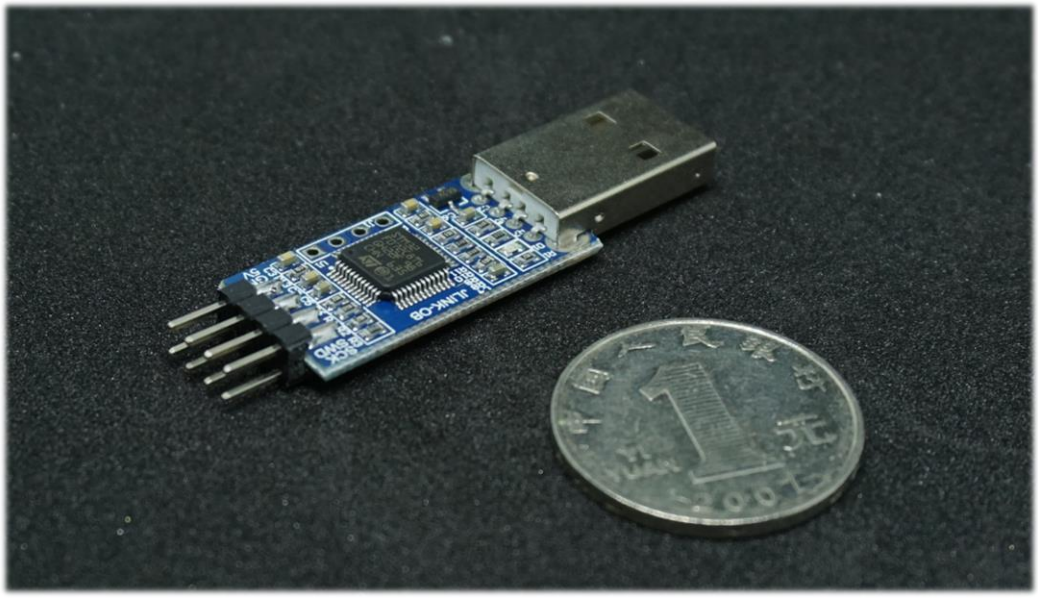
电子屌丝的的情怀：CMSIS DAP 仿真器

说到 ARM 开发，仿真器（准确的说应该是调试器）是必不可少的工具，开发人员可以用它进行程序的下载、调试，在很多时候在线调试可以让程序开发更高效。市面上有很多仿真器产品，最出名的莫过于大名鼎鼎的 Jlink。Jlink 除了可以在 MDK、IAR 等软件中得到很好支持外，官方还有很多与 Jlink 配套的工具以方便用户使用。但是说实话，现在市场上流通的 Jlink 基本都是盗版的，至于正版的价格大家可以到 SEGGER 官网看一下，作为一名电子屌丝，反正我是被吓到了。犹记得笔者当年初学 STM32 的时候，随便 30 来块钱就能买到一个，不过盗版毕竟是盗版，用起来还是不太稳定。

后来听说了 Jlink-OB，网上有很多爱好者 DIY 的各种版本。所谓 OB 就是 On Board，是 SEGGER 专门开发用于把 Jlink 集成到 PCB 板卡上方便调试下载的版本，可以理解为精简版的 Jlink。当时笔者也做了一个，基于 X893 大神的版本，带了串口。说到 X893，这是一个大神博主的名称，它的博客上有个 STM32 版块，分享了好多文章，网上流传的 JlinkOB 大多数都是用的 X893 的版本。这是原博文地址：http://akb77.com/g/stm32/jlink-ob/

这是当时笔者 DIY 的 Jlink-OB，精简了一下，把输出电源改成了 5V，还改了引脚排列。说实话，当时对这个仿真器是爱不释手，基本 Jlink 的功能都有，而且很稳定，还带了串口，下载、调试、串口、供电，一个小小的 JlinkOB 全解决了，再也不用拖着一个砖头还要另外加一个 USB 转串口模块。



再后来，了解到 CMSIS DAP，可能很多朋友第一次听说这个名词。CMSIS DAP 是 ARM 官方推出的开源仿真器，支持所有的 Cortex-A/R/M 器件，支持 JTAG/SWD 接口，在最新的固件版本中，还支持单线 SWO 接口，可以直接在程序里把相应的数据通过 SWO 接口输出到调试窗口，起到类似串口调试的目的。DAP 主要有以下特点：

1. 完全开源，没有版权限制，所以相应的价格会很便宜
2. 无须驱动，即插即用
3. 在新版本的 DAP 里集成了串口，除了下载调试外还能充当 USB 转串口模块，一机两用
4. 性能方面已经可以满足一般用户的需求

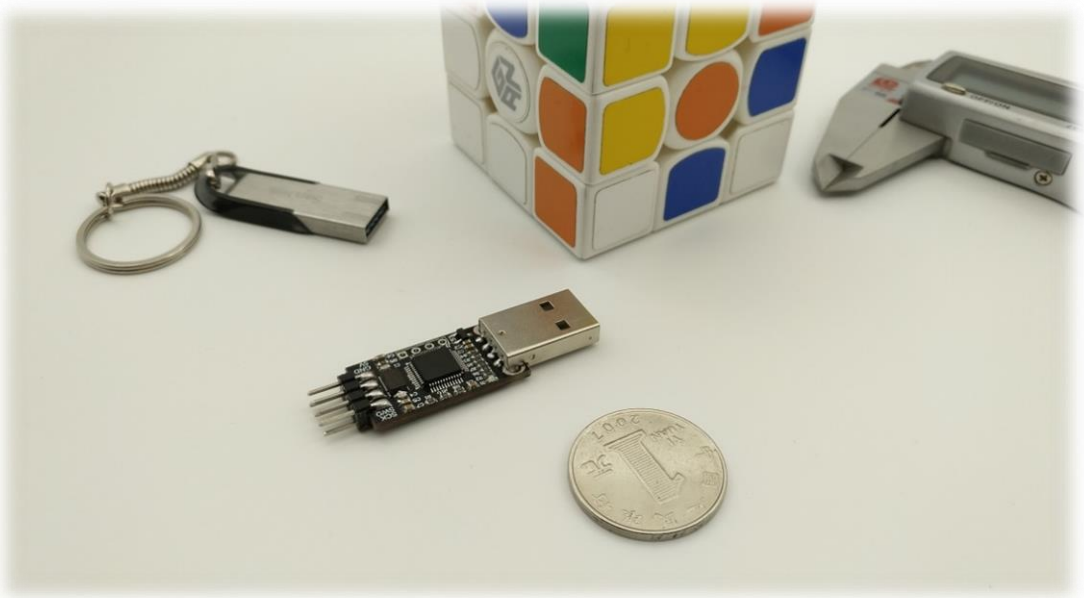
毕竟是开源的，经过这几年的发展，网上也有很多的版本，笔者这次用的同样也是 X893 大神的版本。不过在它的博客里没有相应的博文介绍，我是在 Git 上面找到的，连接如下：

https://github.com/x893/CMSIS-DAP。

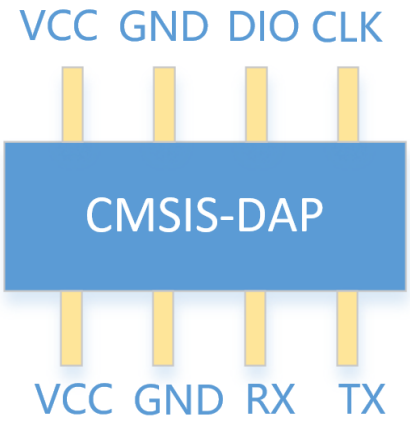
ARM 官网的 DAP 介绍：

http://www.keil.com/pack/doc/cmsis/DAP/html/index.html。

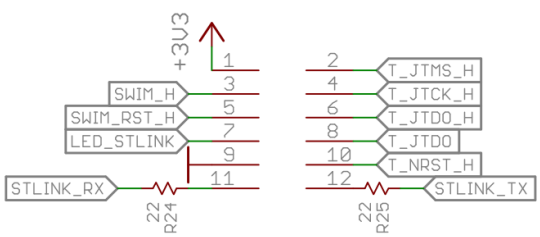
同样，笔者也做了相应的简化，原版本是保留有 JTAG 接口的，但对于大多数人来说 SWD 接口已经够用了，同时还去掉了硬件电路的复位线，不过这个复位线去掉的还真的是挺坎坷，后文会详细介绍整个制作经过，如果你只是想知道怎么制作下载器的话可以跳过这部分，直接到文章末尾下载原理图和固件即可。



采用 USB A 公头的设计，这样比起用 microUSB 线连接过去更简洁。引脚改成了 2*4P 的排针引出，非对称式电源设计，很多串口模块都把 VCC 和 GND 放在两端，这样一旦插反就会带来烧坏电路的风险。输出电源改成 5V，直接使用 4P 的杜邦线与目标板连接，实现供电、下载和调试的功能，需要串口时再增加两根杜邦线即可。



下图为原版的接口设计，保留了 JTAG 接口。



因为之前制作 JlinkOB 也是用的 X893 版本，那时候 X893 的 Jlink 也是保留了复位线，由于 Jlink 会使用软复位，硬件复位线是可以去掉的，所以设计成了上面的接口布局，一直用着，很稳定没出现问题。所以这次制作 CMSIS DAP 的时候去掉 JTAG 接口的同时也去掉了复位线，然后实际使用的时候发现能正常下载程序和调试，但是就是不能在下载程序的时候复位运行。起初以为是 MDK 里面设置的问题，于是把 MDK 里面关于下载调试的所有设置都试了一遍，还是没有复位。

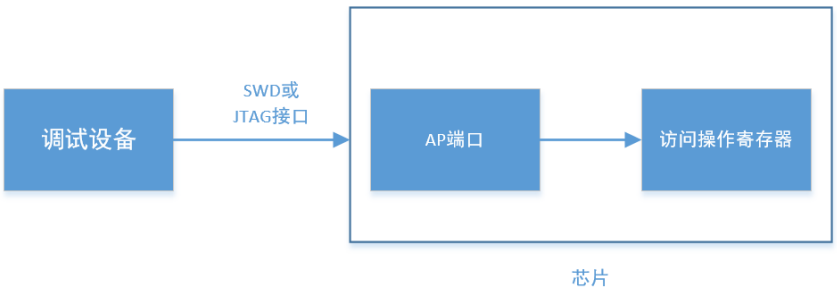
在 MDK 下载选项那里有一个下载后复位和运行的选项，在接上复位线的前提下，勾选了这个选项，下载程序发现可以正常复位，去掉勾选之后下载程序芯片没有自动复位，说明 DAP 还是接收了 MDK 相关复位指令的。接下来的测试中发现，去掉复位线下载程序之后芯片没有复位运行，但是一点击进入调试，在调试状态下点击运行芯片就马上执行运行了，而且退出调试之后也会复位。这说明了其实 DAP 是有软件复位的，只是在下载结束之后只进行了硬件复位。加上之前使用 Jlink 用的软复位一直没有出现过问题，这就更加证实了 DAP 是完全可以使用软复位的，继而省掉复位线，更加精简。

既然要实现软复位那肯定是操作某个寄存器，查阅《Cortex M3 与 M4 权威指南》，其中 Table 7.20 有着关于复位 SCB->AIRCR 寄存器的介绍。从表格中可以看出，要想操作这个寄存器，首先得往高 16 位写入校验字，如果是写操作则写入 0X05FA，读操作则写入 0XFA05。与复位操作有关的是 bit2 和 bit0，操作 bit0 的话只会复位核心处理单元，片内的其他外设都不会复位，是专门设计用于调试操作的，操作 bit2 则可以实现整个芯片的复位。那么基本思路就出来了，我们只要往地址为 0XE000ED0C 的寄存器 SCB->AIRCR 写入 0X05FA0004 即可实现我们要的复位功能。

Table 7.20 Application Interrupt and Reset Control Register (SCB->AIRCR, Address 0xE000ED0C)

Bits	Name	Type	Reset Value	Description
31:16	VECTKEY	R/W	–	Access key; 0x05FA must be written to this field to write to this register, otherwise the write will be ignored; the read-back value of the upper half-word is 0xFA05
15	ENDIANNESS	R	–	Indicates endianness for data: 1 for big endian (BE8) and 0 for little endian; this can only change after a reset
10:8	PRIGROUP	R/W	0	Priority group
2	SYSRESETREQ	W	–	Requests chip control logic to generate a reset
1	VECTCLRACTIVE	W	–	Clears all active state information for exceptions; typically used in debug or OS to allow system to recover from system error (reset is safer).
0	VECTRESET	W	–	Resets the Cortex®-M3/M4 processor (except debug logic), but this will not reset circuits outside the processor. This is intended for debug operations. Do not use this at the same time as SYSRESETREQ.

既然知道了要怎么操作寄存器，那么另一个问题就是要怎么通过 CMSIS DAP 去操作相关的寄存器？所谓 DAP 其实是 Debug Port 和 Access Port 的简称，外部的调试工具通过 SWD 或 JTAG 接口去访问芯片内部的 DP 和 AP 端口，通过 DP 和 AP 端口执行相应的操作。DP 端口主要是进行一些与调试相关的操作或者控制，与我们本次的目的相关不大，所以这里不做过多介绍。而 AP 端口的作用就是可以通过它对芯片所有的寄存器进行访问和操作。



再来看看 ARM 调试接口手册（见附件）的 Figure 7-1，这里介绍了如何通过 AP 端口访问寄存器的过程。

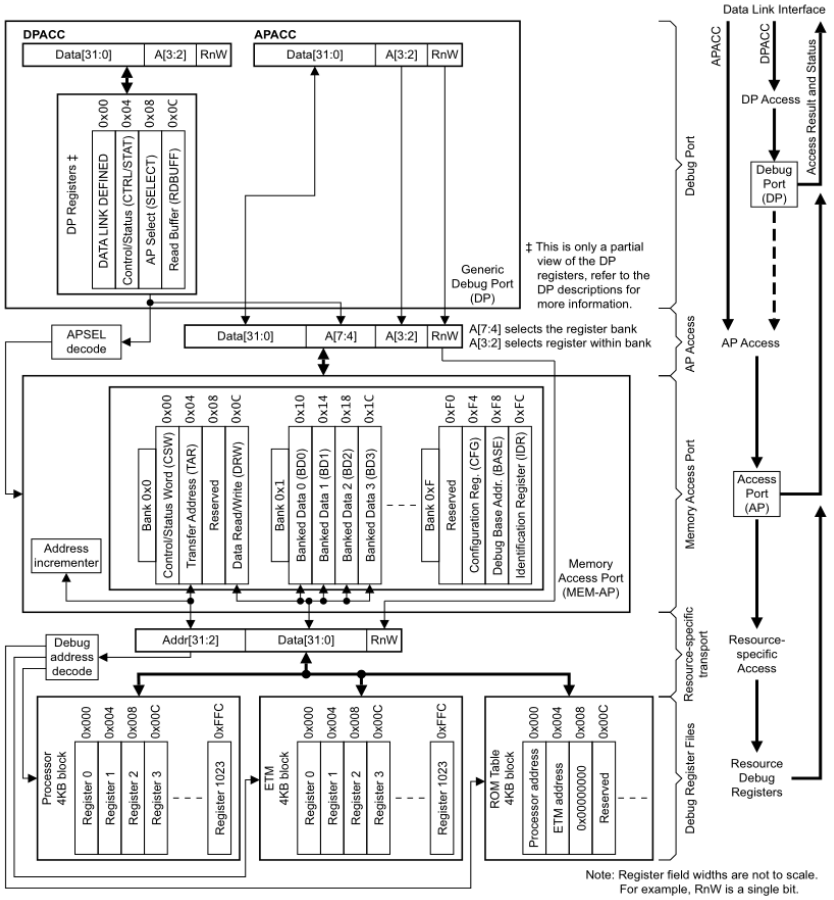


Figure 7-1 MEM-AP connecting the DP to debug components

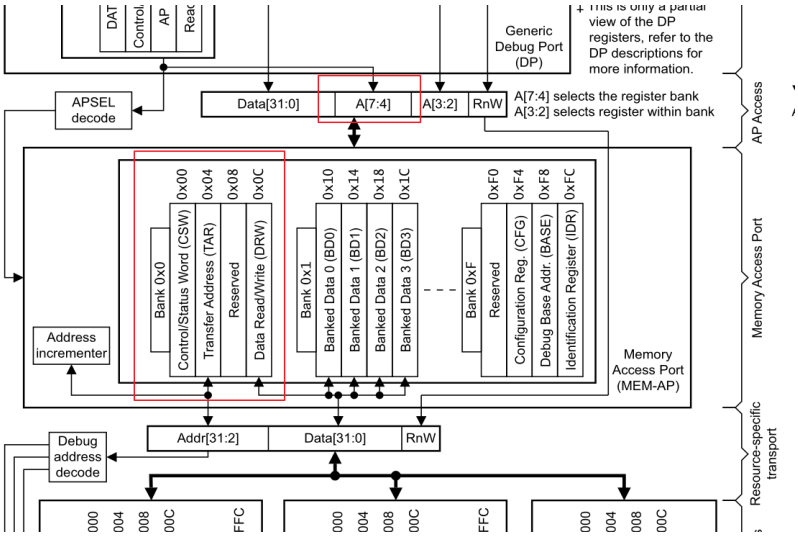
从图中可以看到，整个过程分为三部分。首先，是要确认访问的是 DP 还是 AP 端口、读还是写以及要访问的寄存器。在 ARM 官网对 DAP 的介绍中，有相关传输数据格式的说明。仿真器每次对芯片的访问都是由 Transfer Request 和 Transfer Data 组成，这里的 Transfer Request 的每一位的作用下图都有介绍。Bit0 用于确认访问的 DP 还是 AP，Bit1 选择读还是写，Bit2 和 Bit3 选择要访问的是 DP 或 AP 的哪个寄存器，至于高位默认为 0 就好。

DAP_Transfer Command:

BYTE	BYTE *****	BYTE **********	BYTE ************	WORD *********
> 0x05	DAP Index	Transfer Count	Transfer Request	Transfer Data

- **DAP Index**: Zero based device index of the selected JTAG device. For SWD mode the value is ignored.
- **Transfer Count**: Number of transfers: 1 .. 255. For each transfer a Transfer Request BYTE is sent. Depending on the request an additional Transfe sent.
- **Transfer Request**: Contains information about requested access from host debugger.
  - Bit 0: APnDP: 0 = Debug Port (DP), 1 = Access Port (AP).
  - Bit 1: RnW: 0 = Write Register, 1 = Read Register.
  - Bit 2: A2 Register Address bit 2.
  - Bit 3: A3 Register Address bit 3.
  - Bit 4: Value Match (only valid for Read Register): 0 = Normal Read Register, 1 = Read Register with Value Match.
  - Bit 5: Match Mask (only valid for Write Register): 0 = Normal Write Register, 1 = Write Match Mask (instead of Register).
- **Transfer Data**: register value or match value
  - for Write Register transfer request: the register value for the CoreSight register.
  - for Match Mask transfer request: the match mask for the CoreSight register.
  - for Value Match transfer request: the match value of the CoreSight register.
  - no data is sent for other operations.

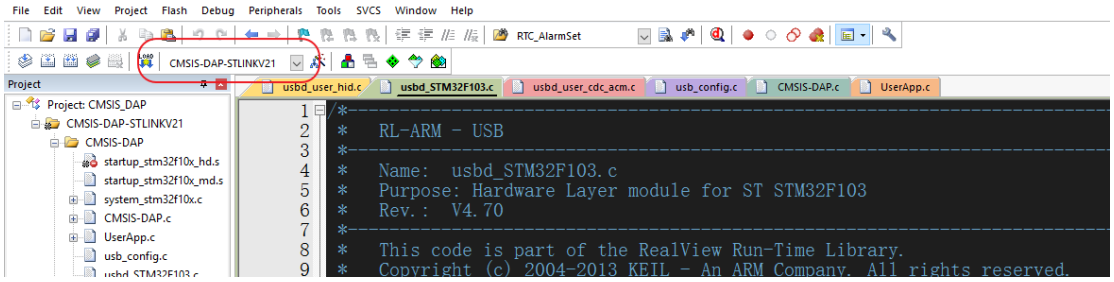
确认选择 AP 端口之后，再来确认选择 AP 的哪个寄存器。图中圈出的 A[7:4]是用来确定是要访问 AP 寄存器的哪个区，通 DP 端口来选择，但是由于默认的是选 Bank 0，也就是我们想要访问的区域，所以这里并没有用到 DP 端口。然后就是根据上面说的 Bit2 和 Bit3 也来确认选择哪个寄存器。在 Bank 0，我们这里用到两个寄存器。首先是 TAR 寄存器，地址为 0x04，这个寄存器的作用是当你把想要访问的芯片内核寄存器地址写入 TAR，AP 端口就会去访问这个地址对应的内核寄存器。然后 DRW 寄存器的作用就是 AP 端口会把 DRW 的值写入 TAR 对应的寄存器中或读取 TAR 对应寄存器的值到 DRW。



那么整个思路到这里就清晰了，我们先是发送控制字 0X05（选择 AP 端口，写操作，选择 TAR 寄存器），再发送地址 0XE000EDOC，然后再发控制字 0X0D（AP，写操作，选 DRW 寄存），再发数据 0X05FA0004，这样就可以实现复位的功能。接下来的工作就是找到源代码里相应的接口函数即可，这个不难，仔细阅读源代码，一层层找下去就能发现底层的接口函数。DAP 每次操作 IO 口都会调用 DAP_SWJ_Pins()函数，由于舍弃 JTAG 接口，每次进入该函数就说明要进行复位操作，所以在该函数添加相应的复位代码即可。

工程路径为：

打开之后，选择相对应的工程



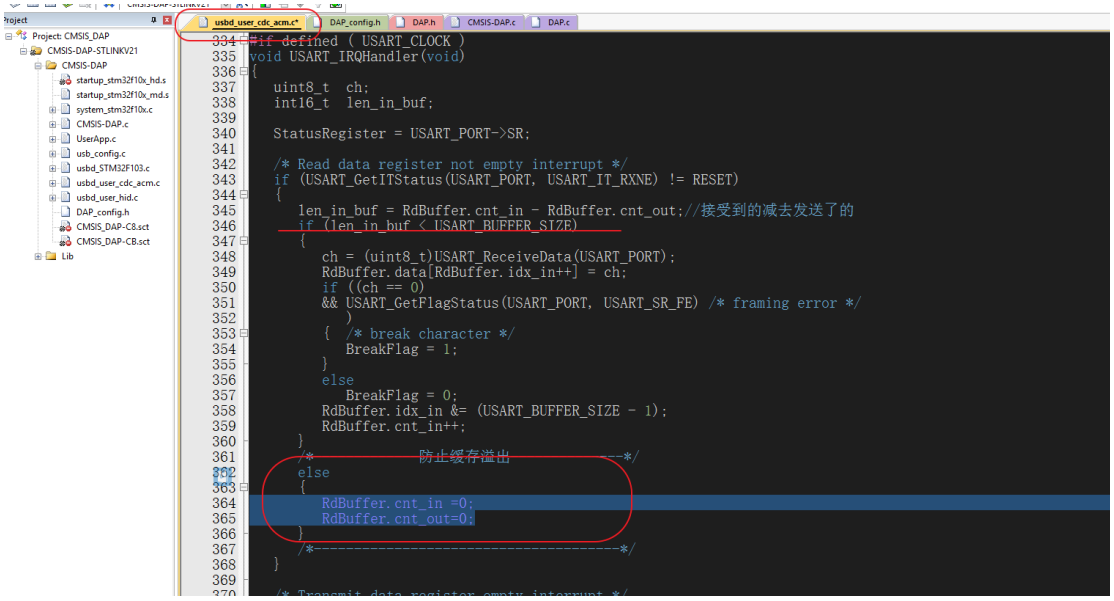
在 DAP.c 里面找到 DAP_SWJ_Pins()函数进行修改：

```
//----------------进行软件复位 -------------------------//
Reset_Count ++ ;
if(   Reset_Count>0  ) //第二次进入才复位
{
   Reset_Count = 0;
   write_data = 0xe000ed0c;SWD_Transfer(0x05,&write_data);Delayms(1);   // 往TAR写入 SCB->AIRCR 地址
   write_data = 0x05fa0004;SWD_Transfer(0x0d,&write_data);Delayms(1);   // 往DRW写入 目标设计
}
```

之所以要进行第二次复位的判断是发现当下载大程序的时候（笔者测试的 hex 大小为 150K）会莫名其妙下载到一半就失败，用逻辑分析仪发现进行了两次复位，不知道出于什么原因要进行两次复位，但可以肯定第二次是下载程序结束后的复位，所以我们只要跳过第一次复位即可。

本以为可以安心使用了，但是后来又发现 CMSIS DAP 的串口有点小 bug，当你的单片机不停的通过串口快速（笔者测试为 10ms）发数据时，这时候会发现在 MDK 里面会提示找不到 CMSIS DAP，需要重新上电才能检测到 DAP，然后当你关闭串口助手再打开时会发现此时串口助手没再接受数据，给人的感觉就是 DAP 卡死了。

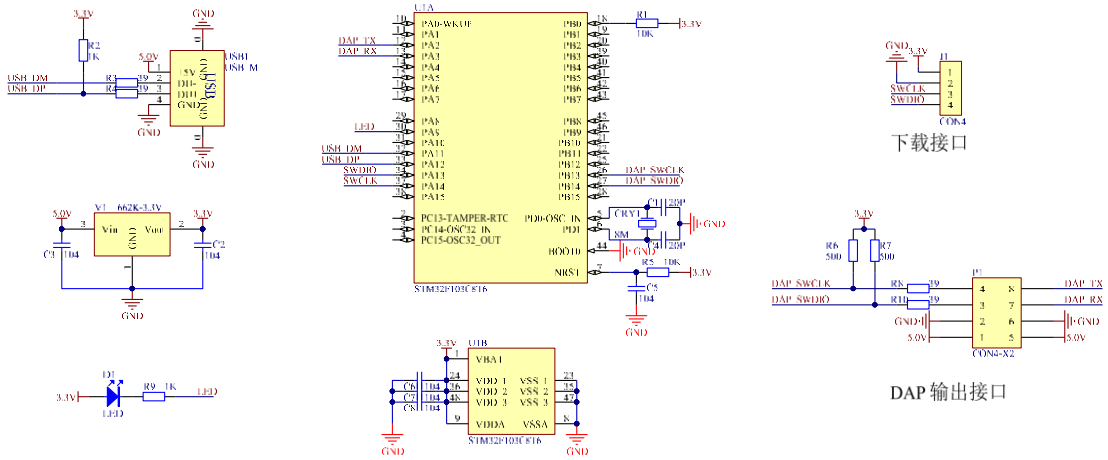
再次查看源代码，发现在串口接收中断里，DAP 会对接收到的数据的多少进行相应的判断处理，但是却没有当接收数据大于接收缓存时的处理。所以当串口数据量很大时或者 DAP 正在处理其它任务没有及时处理接收的数据就会造成缓存溢出，如果此时不做处理就会出错。笔者的解决方法是增加一句判断，当缓存溢出的时候，清零输入输出的计数变量，这样就会重新计算接收数据的长度。



至此，CMSIS DAP 的所有问题全部解决，使用至今一直很稳定工作。不过值得一提的是，修改后的软复位代码只对 Cotex M3 和 M4 内核的单片机有效，至于其他的系列请大家参考同样的方法去修改。

下面给出笔者精简之后的原理图，去掉了 JTAG 接口，同时在 SWD 输出接口处增加两个 500 欧上拉电阻，增强 32 的 IO 口输出驱动能力,实际体验就是允许使用更长的杜邦线连接到目标板。笔者测试过可以使用 50cm 的杜邦绞线，不过这个因杜板线的质量而定，当你发现下载时提示错误时，这时你就得考虑更换更换更短

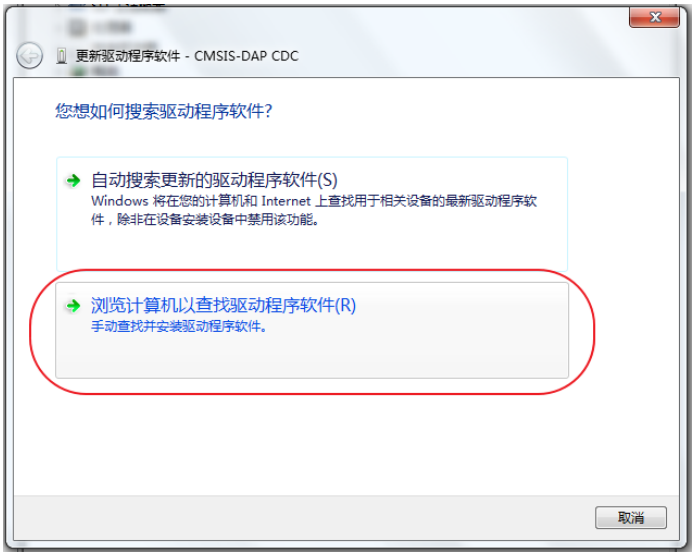
以及质量更好的杜邦线。



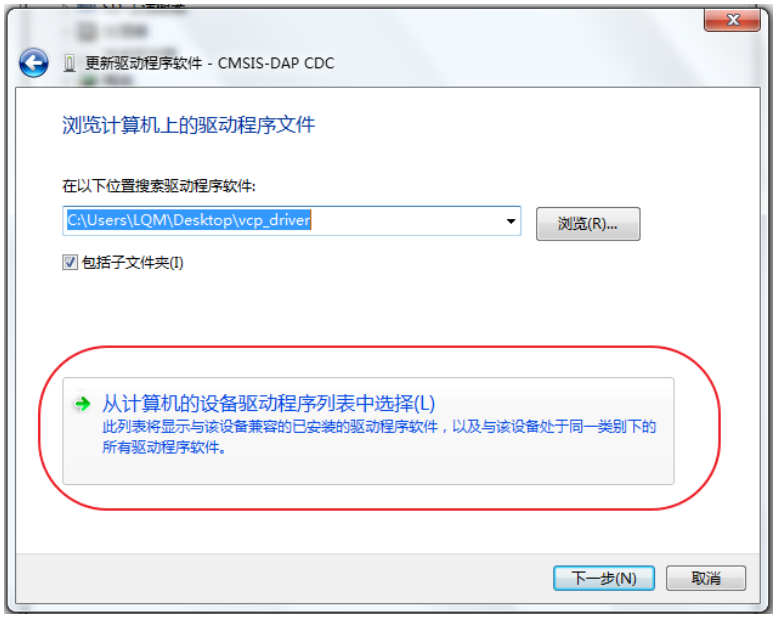
虽说 DAP 是免驱动的，但是在部分 WIN7 或 WIN8 系统，会存在 DAP 的虚拟串口不可用的情况，这时需要手动添加驱动。若你是 WIN10 系统或者设备管理器 DAP 的串口设备没有问号就可以跳过这部分。

右键，选择更新驱动：

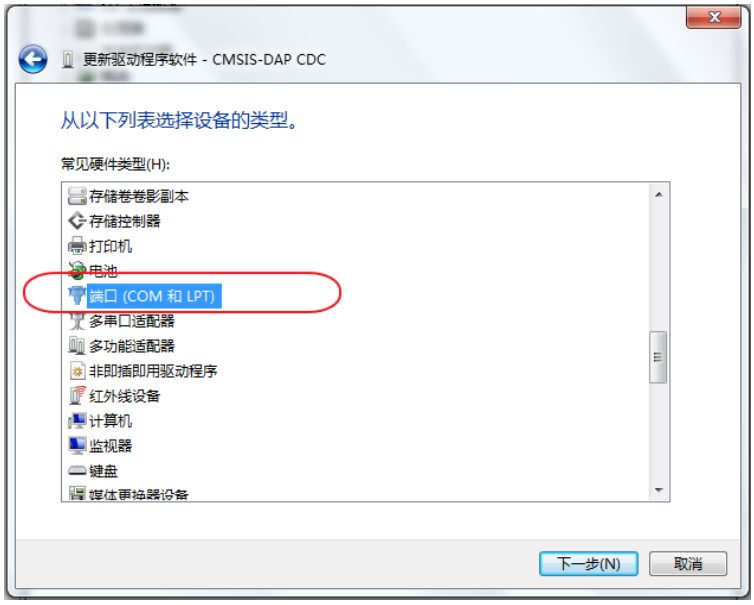
选择浏览：



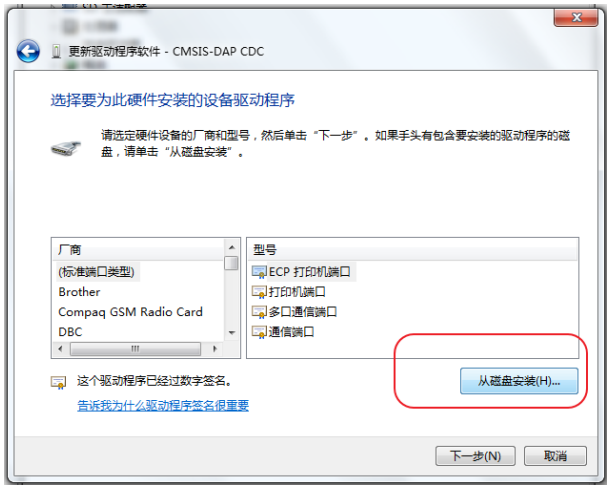
再选择：



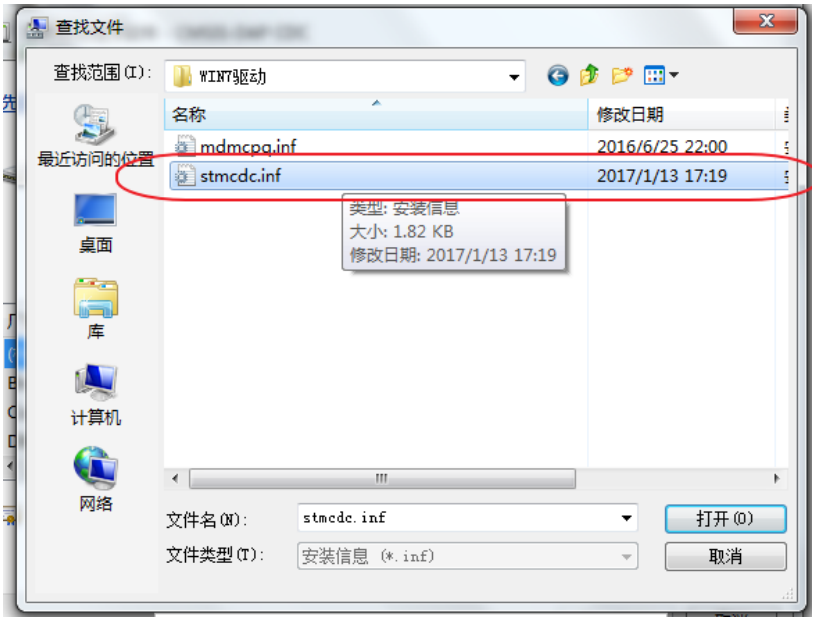
选择端口，点击下一步：



点击从磁盘安装：



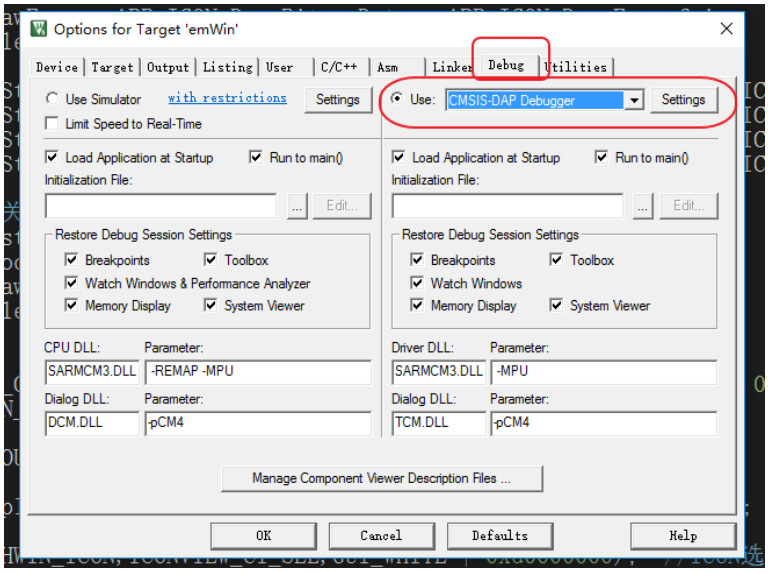
选择附件里 WIN7 驱动文件夹下的 stmcdc.inf



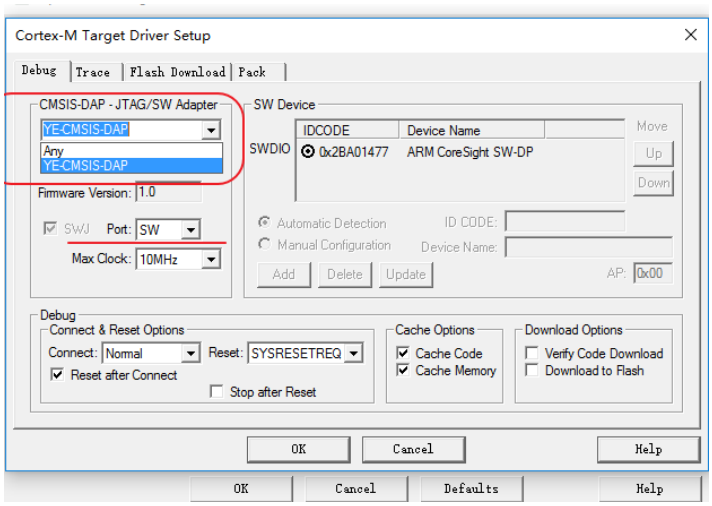
选择之后，点下一步，这时候会跳出警告，无视，直接安装即可，出现以下界面即安转成功。

那么要在 MDK 里面使用 CMSIS DAP 需要简单的设置一下，点击“魔术棒”按钮：

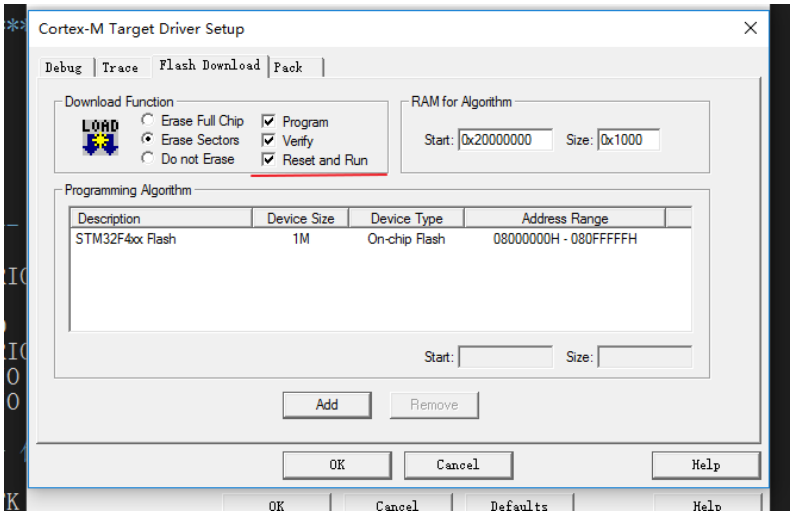
选择仿真器，并点击 setting：



在弹出的选项框里，把 any 换成仿真器，并选择 SW 模式，其他默认就好：



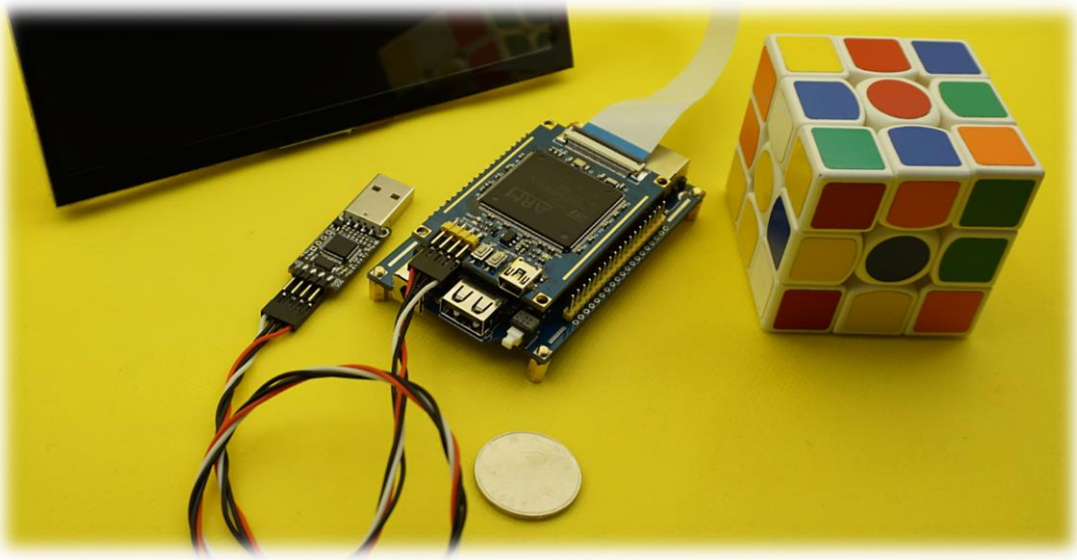
最后别忘了勾选下载结束够复位运行：



至于性能方面，笔者拿 165K 大小的程序文件在 MDK 里进行下载测试，DAP 需要 11 秒，而 JlinkV8 同样也需要 11 秒。DAP 的 SWD 接口速度可以达到 10M，所以调试和下载速度都不输给 Jlink，显然 DAP 对于我们很多人来说已经够用了。至于串口部分的测试，大家打开一个串口助手，把 DAP 的 TX 和 RX 短接测试即可，

笔者测过在波特率 1382400（此为笔者串口助手的最高波特率）下仍可正常工作。而且 CMSIS DAP 不需要安装驱动，电脑会识别为 HID 设备，插进电脑就可以使用，同时还利用 32 的 USB 虚拟串口在实现仿真器的同时还实现了 USB 转串口的功能，很巧妙。毕竟是开源的东西，没有版权的限制，这对于我们这些电子“屌丝”而言无疑是福音。对于这个 DAP 仿真器笔者一直很喜欢，之前的 Jlink 已经冷落在箱底堆灰尘了。

可能有些朋友会问：怎么不加一个自恢复保险丝？因为平常开发 32 的应用经常要外接一个大的液晶屏，加上其他的一些外设，整个系统供电经常达到 800ma，由于自恢复保险丝随着电流越大压降越大，很多时候会让一些电路因为低压而无法工作，加之现在的电脑 USB 口都有很完善的电源保护，不必担心会因意外烧毁 USB 口。当然，你也可以按照自己的喜好重新进行设计。



至此，整个 CMSIS DAP 制作介绍完成，详细的源代码以及原理图见附件。

另外，笔者在聚丰众筹发起了众筹，如果大家喜欢这个仿真器的话还请支持一下我们。

http://z.elecfans.com/103.html

反派小智

2017-3-18