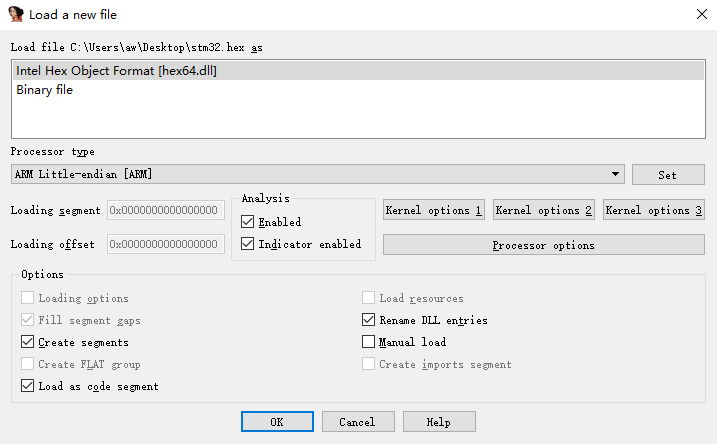
## 分析

程序地址：

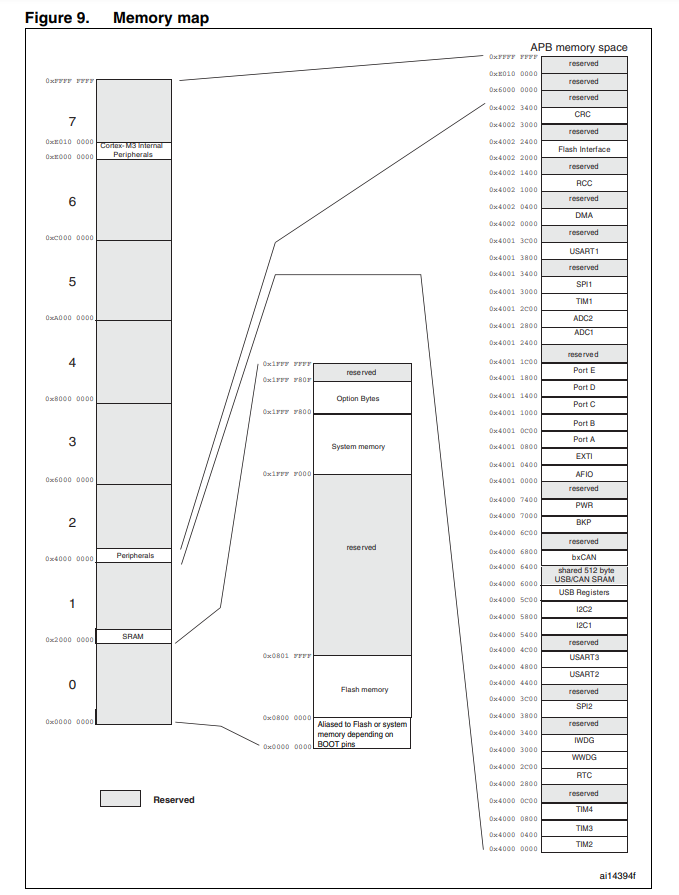
https://dn.jarvisoj.com/challengefiles/confusedARM.hex.f4e616545ff1a18526b9d1c90ea648ff

这个程序是STM32F103X8，因此可以在这个网站上找到对应的dataset：https://www.alldatasheet.com/。

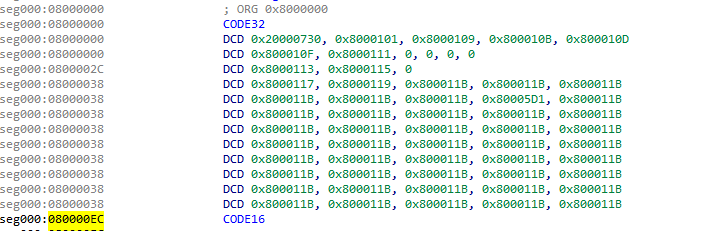
用ida打开该程序，把程序设置成arm小端序：



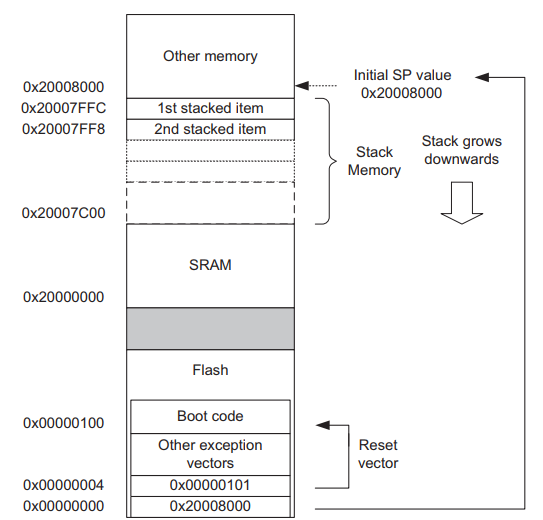
Dataset中其内存映射如下图：



打开后，首先是如下图的数据：

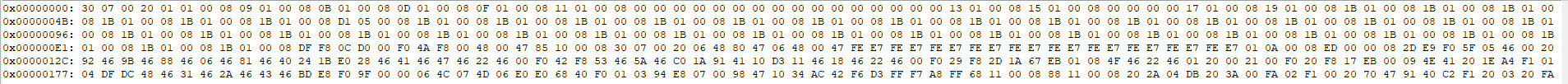


可以看到，这些数据并没有被识别位代码，第一个0x20000730，根据上面的内存映射可以知道这里是SRAM区域，紧跟着第二为0x8000101位于SRAM区域下面的那一块儿内存区域中的Flash memory中。内存映射图的最下面的详细的内存映射如下图：

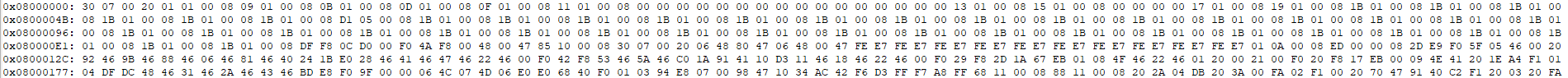


可以看到Reset是在0x00000004处，指向了0x00000101，并且在stm32中支持重映射，其会将0x08000000开始的内容重映射到首地址0x00000000中。查看这两处内存的内容：

0x00000000：

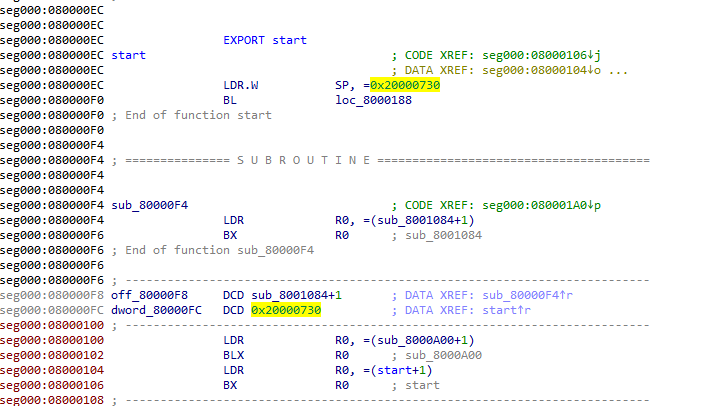


0x08000000：



可以看到这两处内存中的内容是一样的。

也就是说在ida中看到的第二个数据0x8000101就是reset所执行的地址，也就是reset。到0x8000101处：



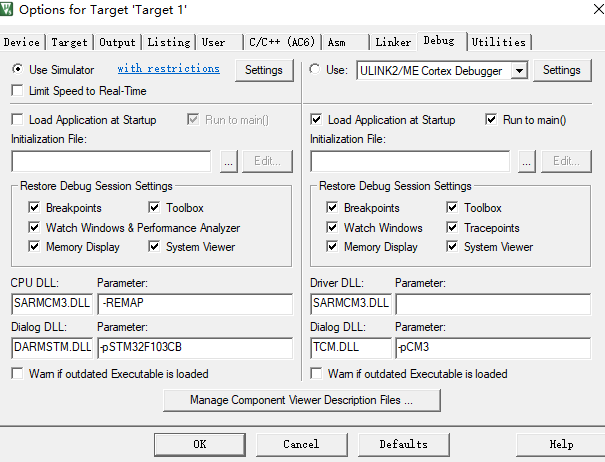
可以在0x8000104这条指令的地方有一个start，这个start是sp，也就是说0x20000730是堆栈地址，那么0x08000100基本上就可以确定是程序的入口地址了。在0x080000f6处有一个比较大的跳转，调用了函数sub\_8001084。看一下sub\_8001084：

伪代码如下图：

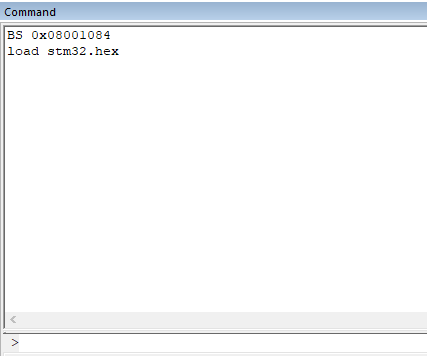


可以看到满足某些条件就可以输出出来flag。但是这些红色的内存看起来不太好看，可以在Edit->Segments->Creat segment来创建一个段，把这些内存区域包括进去就好了。那么看来函数sub\_8001084就是主函数。

接下来使用MDK动态调试一下程序,在开始之前先去下载对应的包(https://www.keil.arm.com/packs/),创建好工程后,修改配置:



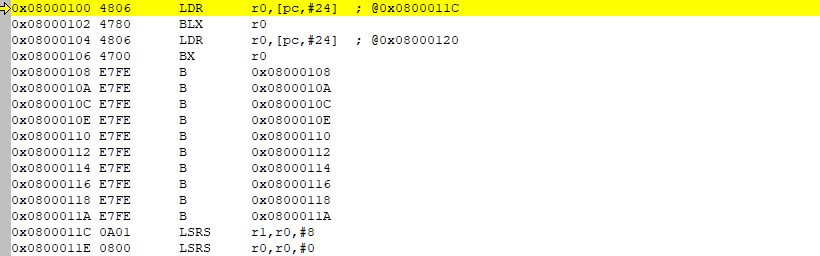
按CTRL+f5开始调试,然后在下方command窗口将源文件load进去:



然后点左上方的RST按钮:



然后程序就会自动跳到0x08000000处并且断下:

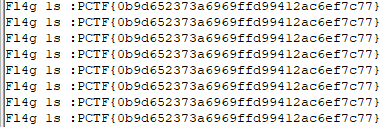


从上面知道函数入口点在0x08000000,主函数在0x08001084,直接运行到0x08001084:



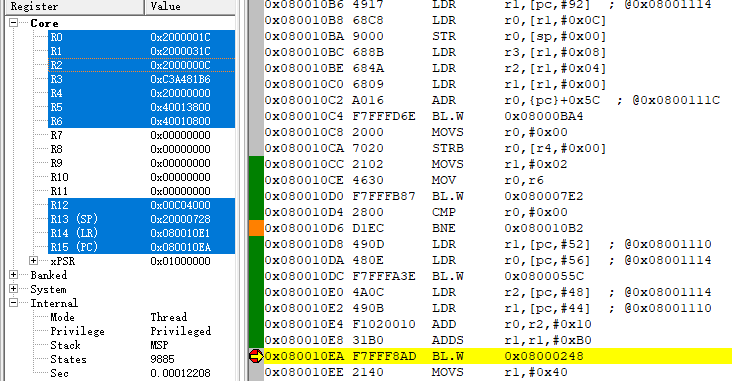
根据主函数中的伪代码，可得知加密函数的key的地址为：0x2000000C，解密后的flag的地址为：0x2000031C。在主函数中sub\_8000560和sub\_800055C都对key进行了操作，那么这俩肯定不是解密函数，只剩下了sub\_8000248函数。

那么直接运行程序貌似就可以直接输出flag了：

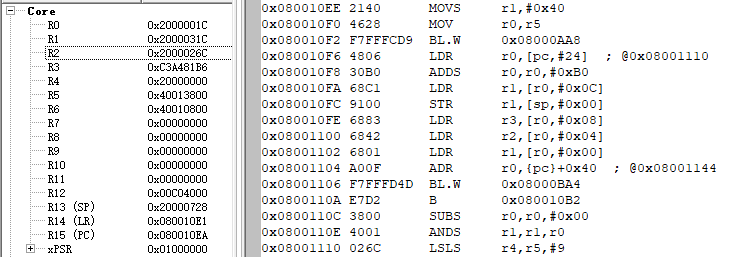


但是提交flag是错误的。

想一想好像也只能说解密函数出问题了，运行到地址0x080010EA调用sub\_8000248函数的地方：



根据结合上面的函数的参数分析，sub\_8000248函数的第三个参数应该是解密的key，从上面调试来看，传入sub\_8000248函数的key的地址是0x2000000C，但是调用sub\_8000248函数之前sub\_800055C函数对key进行了处理，也就是说key传错了，传入的是处理之前的key，因此这个地方要传入的应该是0x2000026c，所以把传入的地址改为0x2000026c就可以了。因此直接在MDK中修改R2寄存器的值为0x2000026c：



然后运行，因为是临时修改所以只有第一个flag是正确的输出：



当然也可以直接给程序打patch，在程序中把代码改掉。

## 参考文章

https://www.armbbs.cn/forum.php?mod=viewthread&tid=109321