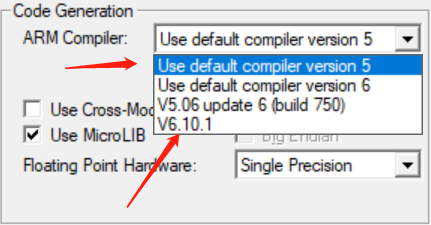
履带式ROVH7主控开发遇到的若干问题

**导言：**

这个文档主要是记录笔者在编写H7系列时候遇到的许多坑，包括CubeMX如何配置，驱动各个通信协议如何驱动，以及王田俊工程师电路板有点小BUG的地方来做个记录，踩过的坑避免后人再踩。参考文档还是正点原子的H7系列的HAL库开发指南，野火没对H7做一个专门的文档，正点原子有些地方说得又不够清楚，有些细节不能注意到。此外，本次记录也获得了许多经验，最重要的一条就是多看看ST官方的H7参考手册，很多问题都有说明，正点原子还是野火出的书本质也是参考ST官方的手册编写，如果出现了正点原子文档里面无法解决的问题，那么一定要多翻阅ST官方手册。

1. 关于编译器

编译器申龙号项目里面用的是ARM Compiler\_V5，建议安装MDK KEIL的版本为5.36之前的（笔者这里用的是5.27）。不太建议为了追求完美去arm官网下载最新的编译器，因为5.37以后的keil默认不自带v5版本交叉编译器。下图里面这个V6版本就是我之前下最新版编译器留下的。

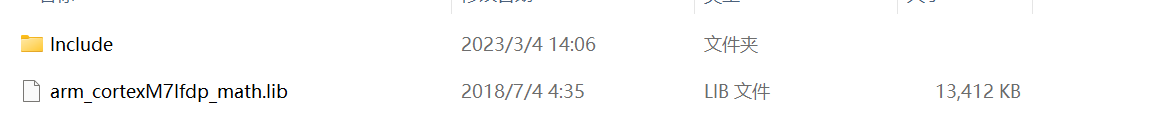


1. 关于DSP库的使用出现大量报错（1000+ error）
   1. 首先什么是DSP库

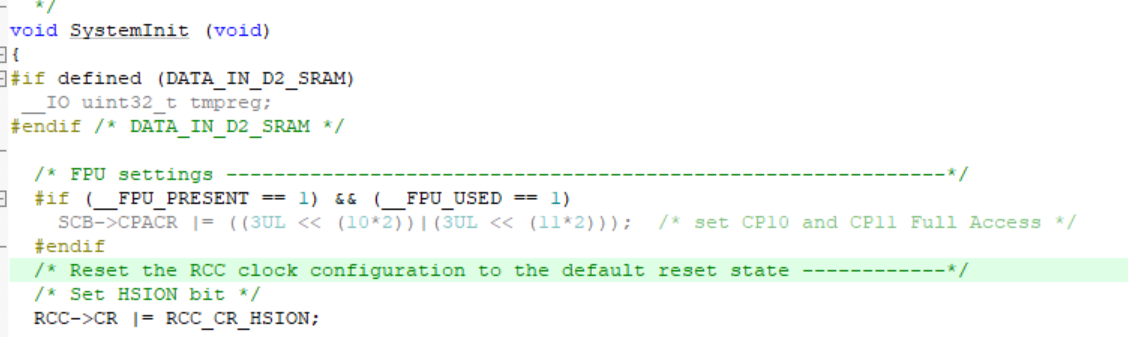
STMH32H7采用Cortex-M7内核，在数字信号处理方面增加了DSP指令集，支持诸如单周期乘加指令（MAC），优化的单指令多数据指令（SIMD），饱和算数等多种数字信号处理指令集。相比Cortex-M3，Cortex-M4在数字信号处理能力方面得到了大大的提升。还可以进行FFT，矩阵运算等，相比micro lib 库内的<math.h>函数快了不少。浮点运算有博客测试大概快6倍（厉不厉害）

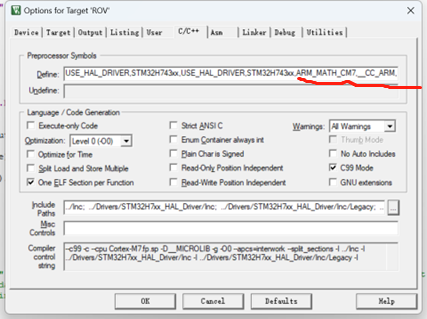
* 1. 如何使用DSP库

首先需要在工程里面添加lib外部库和头文件，本工程笔者放在了 ../components/algorithm 内



其次，需要在全局宏定内使能相关宏定义，要开启FPU功能，从启动文件如下图看出，FPU的开启需要\_\_FPU\_USED和\_\_FPU\_PRESENT 都为1.



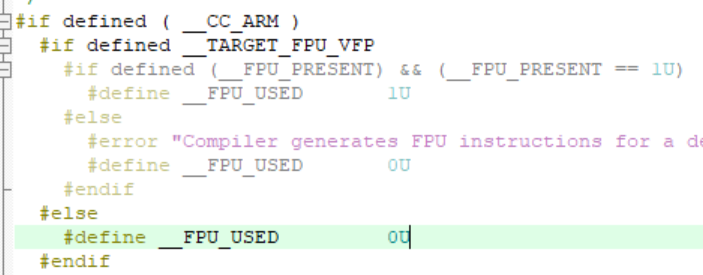
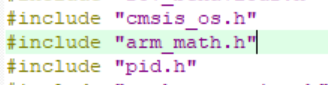


所以我们在工程里面添加

ARM\_MATH\_CM7,\_\_CC\_ARM,\_\_FPU\_USED=1U,\_\_FPU\_PRESENT=1,ARM\_MATH\_MATRIX\_CHECK,ARM\_MATH\_ROUNDING

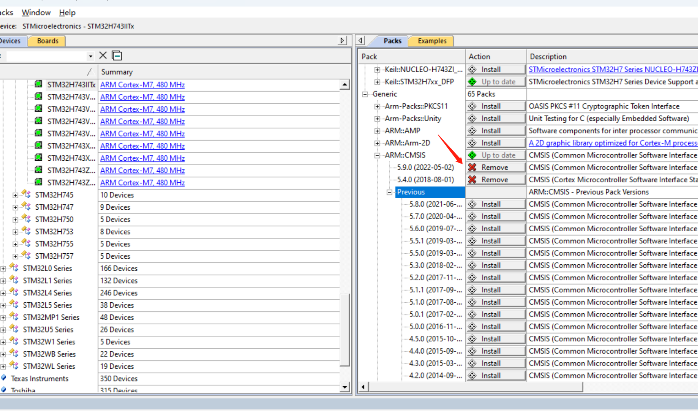
（为什么一个为1U，1个为1这个是根据自己内核包的版本决定，都为1会出现大量重复定义的问题，这个当时也弄了好久）下图内核文件定义的是1U

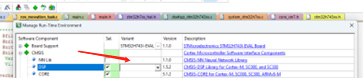
然后直接包含”arm\_math.h”文件即可，这个相比<math.h>做过优化

* 1. 问题的解决

这个问题根本原因是是由于CMSIS包太新的问题，DSP的source文件无法正常运行。参考博客：[在keil中加入DSP库并且使用arm\_math.h - 哆啦A梦的食用宇宙服 - 博客园 (cnblogs.com)](https://www.cnblogs.com/twinkle--star/p/16021904.html)



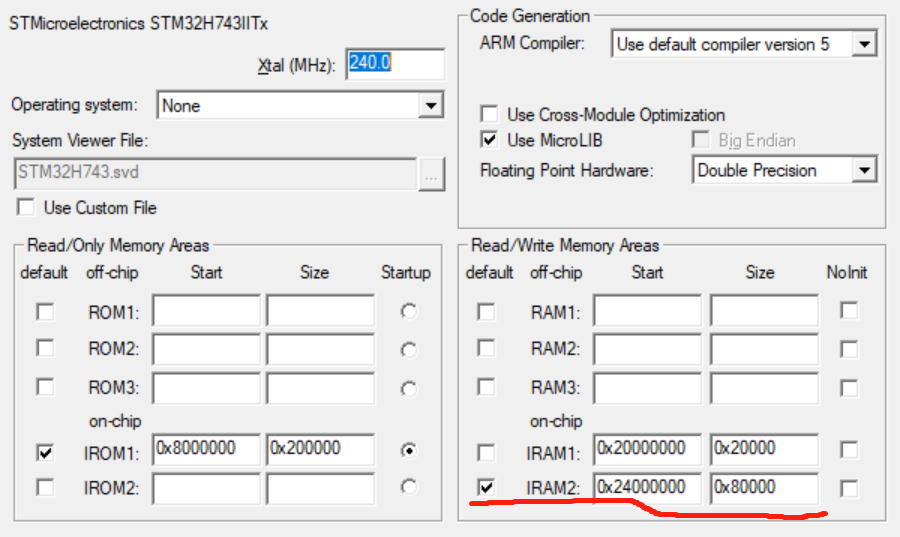


打开Pack包管理器，把新版本的CMSIS的Remove掉，安装老版本的，建议5.4.0，Previous里面可以选择历史版本。

1. 串口DMA的若干问题
   1. DMA搬运问题

在一开始，我直接使用CubeMX生成的代码，然后测试我自己直接操纵寄存器，发现上位机根本接收不到数据。心里想这是正常的，然后换了HAL库的接口，HAL\_UART\_TANSMITER\_DMA()这个，发现还是接不通，这就离谱了，你官方的API都拨不通嘛，而且这套代码我是拿之前的F4开发板上跑通过的。没办法，谁让H7系列资料少的可怜呢，自己找呗！网上说什么DMA配置应该在串口配置前面，我寻思mx生产的也是这样的啊，然后找各种初始化的问题找了半天。最后终于找到了一个帖子跟我有相同的问题了（[(21条消息) STM32H7上使用CubeMX,HAL配置UART串口DMA的坑\_stm32h7 串口dma\_秋阳 Cody的博客-CSDN博客](https://blog.csdn.net/cqy1028/article/details/117408683)）

TNND，原来H7的DMA总线访问不了DTCM区域（0x20000000，大小128KB）我的全局变量放在了BSS段，会自动从0x20000000这个地址加载，难怪尼玛的读不出来。解决办法也特别简单，搬运的变量把它定义在DTCM区域外即可，MDK通过（\_\_attribute\_\_关键字指定变量存放地址）。或者把RAM起始段改一下，如下图。正点原子的例程可以跑通就是他工程里面改了这个，但尼玛的开发指南又不说，面向新手的教程这么重要的细节不说我敲！

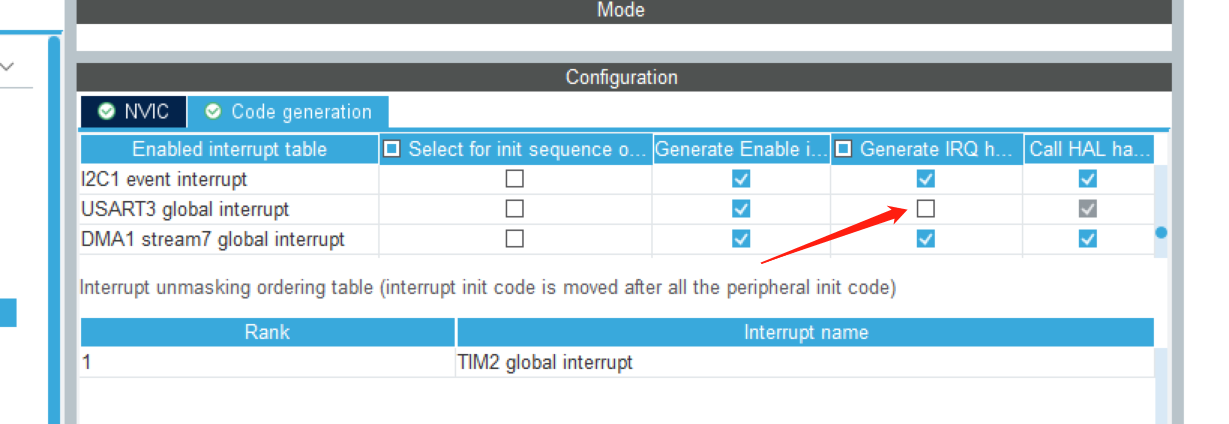


* 1. Cubemx生成配置中断注意点

带上FreeRTOS后，会涉及到临界保护段的操作，即会有一些中断被关掉。但是IMU的数据我们是需要实时更新，使用IMU任务是通过中断函数直接启动，CubeMX生成的时候，优先级最高只能是5。这里我们希望串口中断不受FreeRTOS的管理，即取消第三个(Used FreeRTOS Function)的对钩，就可以把优先级调到5以下，但是注意中断里面不能使用FreeRTOS的API

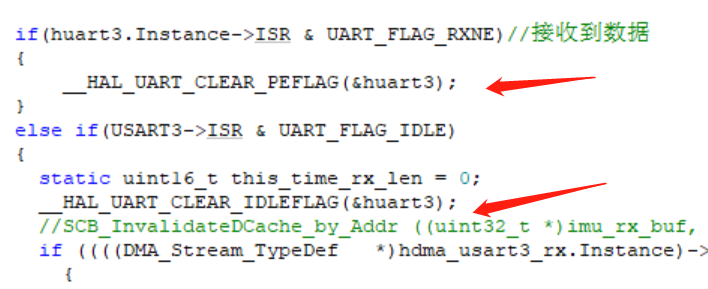


此外，由于我们有多个串口中断，所以直接重写中断函数，而不是重写回调函数。所以CubeMX取消生成IRQ函数自己写。



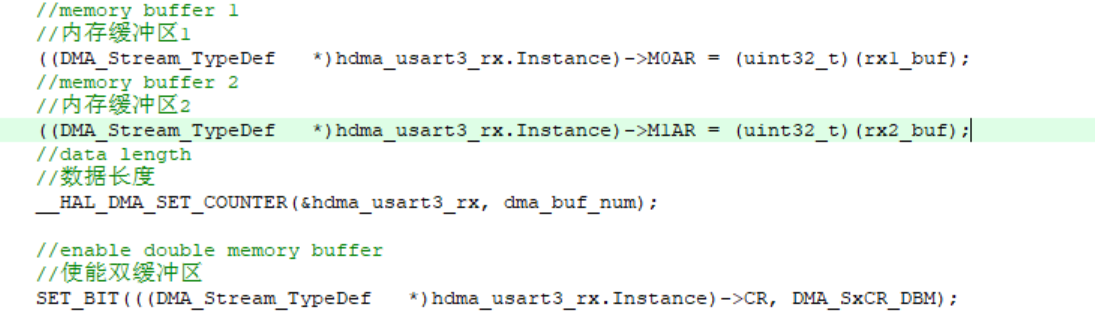
* 1. 串口空闲中断（IDLE）

清除中断应该采用对应清除函数，比如下面那个，上面那个是清除串口3所有中断标志，实际测试发现根本无法清除，导致串口空闲中断进不去的问题。



* 1. DMA双缓冲配置

这里我为什么不使用HAL库的那个DMA接口，一个当然是因为这是在中断里面，讲究快进快出，直接操作寄存器更合适。第二个，也是最最最最重要的原因，当时没有找到HAL库用双缓冲收发的API（虽然后来找到了）。初始化就是让DMA目标地址指向接收Buffer即可



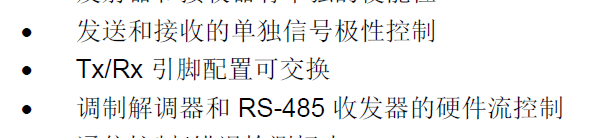
中断里面的判断逻辑也很简单，判断哪个缓冲器是空的就放那边。

* 1. 高度计的引脚问题

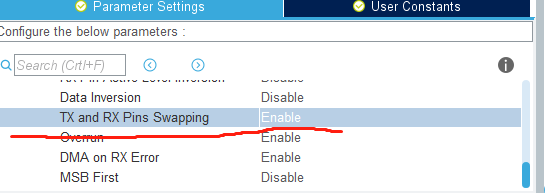
由于画最小系统的原理图时候，笔者没有仔细看，导致UART5的两个引脚反了。如果是CPU引脚直连的话，把TX、RX反接即可，但是后面加了5V转3.3V的逻辑芯片，所以改变线序也没有用。硬件上可以解决，在逻辑芯片那边短接一下即可，但是这样就绕过了逻辑芯片，而且不够优雅。



最后笔者在翻阅ST官方手册时候，关于串口通信这里看到了H7里面，发送偏移寄存器（TX Shift Reg）和接收偏移寄存器（RX Shift Reg）与TX引脚，RX引脚之间弄了个交叉连接，这里的意思是支持了引脚互换功能，这样大家在设计PCB的时候就可以比较随性了，接反了也没有关系。（不得不说H7是真尼玛高级）



我们要使用这个功能也很简单，即在CubeMX使能这项功能即可，此时RX和TX的引脚是反过来的。

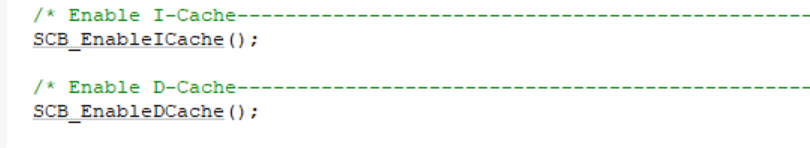


1. Cache的一些小问题
   1. 清除缓存

正所谓H7的Cache真的是把双刃剑，它提高了程序的执行效率，同样也给开发增加了许多麻烦。凡是涉及有FIFO的，DMA搬运的，建议转移数据之前都清除下缓存，强迫CPU读取内存，以此保证接收到的数据是最新的



* 1. 开启Cache



建议开启Dcache和Icache，不然某个驱动又可能出现莫名其妙的bug。唉，H7系列真是服气。

1. 关于ADC模块（水位计）的问题
   1. 时钟频率

这里我给ADC时钟为75MHZ，然后对其进行6分频，也就是说AD得频率为12.5MHZ左右。正点原子文档推荐的ADC频率需要在36MHZ以下，否则就会不准

* 1. 采样时间

项目的采样时间我选择的是64.5个循环，时间还是比较长，不过无所谓，水位计检测任务优先级不高。这里尽量越长越好，最开始1.5个循环时候，检测CPU内部温度的那个AD，采集出来的电压根据ST官方的H7参考手册的换算公式居然尼玛出现了负值。在我把时间变到64.5时候，就恢复了正常，大约40℃。

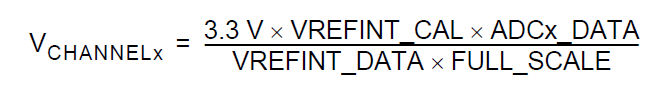
* 1. H7的注意点

在我测试AD时候，一开始用的之前F4上面跑的程序，但是同一个程序怎么H7又不对了，尼玛的！接5v读出来的数值居然比0v小，又是一些诡异的BUG，为此我仔细的对比的正点原子的例程，一个个排除问题。原来H7的AD采集是支持差分输入的，如果默认的话，那么一端是AD，另一端就是内部的参考电压。难怪3.3v读出来比0v小，所以AD转换的时候需要多加配置成单边模式。



* 1. 内部参考电压的坑

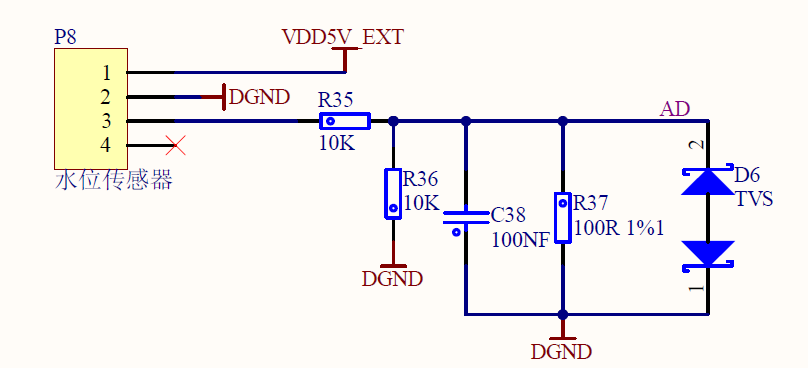
由于外部电源是存在波动的，所以VDDA引脚接的不准的话，那么最后换算的值也并不是准的。但是单片机内部都是有个参考电压的，ADC3的ADC\_CHANNEL\_VREFINT引脚就是采集内部电压。官方的手册有个公式，是可以根据内部参考电压的值来推出VDDA的实际电压值，从而得到准确的电压值。但是出现了个问题，官方数据手册说内部参考电压约为1.2v，但是我读取出来的不知道为什么只有0.4v，这个坑我暂时没有找到问题。



* 1. ~~王师兄电路上的小BUG~~

~~在我程序终于跑通，把电阻焊上去的时候，诡异的事情又发生了，尼玛的接3.3v读数又特别小，但是0v的更小。由于确定了程序没问题，那么问题只可能在电路了，我电表测量电压和电阻阻值，发现都没问题，然后我又仔细看了看电路图，嘶~~~

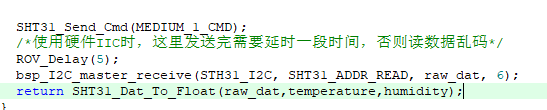
~~首先王师兄的想法是没问题的，AD进来接个RC低通滤波器，但是这个R37会和R36并联，导致这个分压电路出现问题，电压都分在了R35的两端，难怪3.3v读数特别小，一定要把R37取下来，不要焊接！王师兄扣鸡腿，耽误了我半天找bug的时间，用一个电容做滤波就够了。~~



1. IIC（温湿度模块）的问题

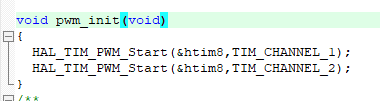
网上找到的STH31的例程多半是用普通IO口，采用软件模拟IIC通讯做的。至于很多教程采用软件模拟的原因，说是ST为了规避飞利浦的专利问题，把硬件IIC做的比较复杂，不稳定。但F4以上的系列貌似就不存在这些问题了，不过U1S1，以前学F1时候BUG确实很多，唉。

易验盯真，都什么年代了还用传统IIC。H7系列既然主打的是高端，那么肯定是硬件IIC木得问题啊。但是我一开始实际测试的时候发现，IIC总是容易接到乱码的数据，但是我单步调试有几率没有问题，emmm，一开始我以为这种情况是见鬼了。查询资料发现，单步调试会主动阻塞语句，根据情况应该在IIC读取后添加延时。



1. PWM波的问题

关于PWM遇到的问题还好，舵机模块和水下灯都可以正常工作，但是注意调一下舵机限位。



记得要开启一下

1. CAN总线遇到的问题

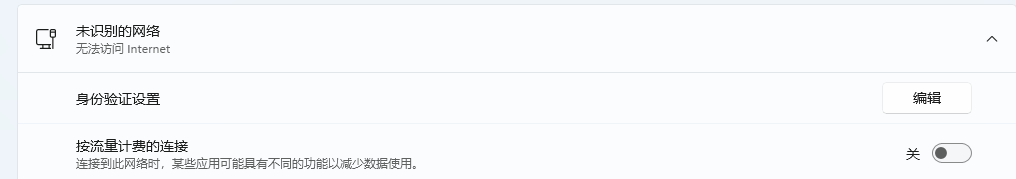
CAN的硬件方面接线简单，软件层面也是不能直接用CubeMX生成的代码，需要注意开启一下



1. 网络问题

网络这部分BUG比较多，一开始是信号灯不亮，说明网卡并没有正常工作信号灯正常应该是一个常亮，另一个发信息时闪烁

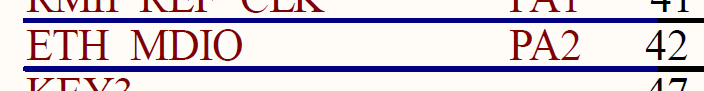
一开始发现灯都没有亮，地线接触不良，重新焊下总算是亮了，但是网卡一直掉，一会识别一会不识别



于是只能继续找硬件问题，首先网络芯片跑着跑着掉，一摸还特别烫，可能是电源部分有问题了，王师兄测了电源一块，发现有个电阻阻值不对，发生局部短路（两点间阻值只有40Ω），重新焊阻值变为600Ω

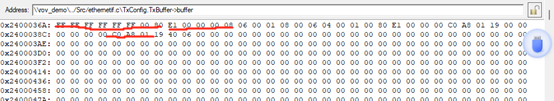
重新焊接，板子识别倒是稳定了一些，但还是没有数据，于是继续找硬件的问题

第一版的硬件图因为名字不一样，所以有根线没有连（离谱）





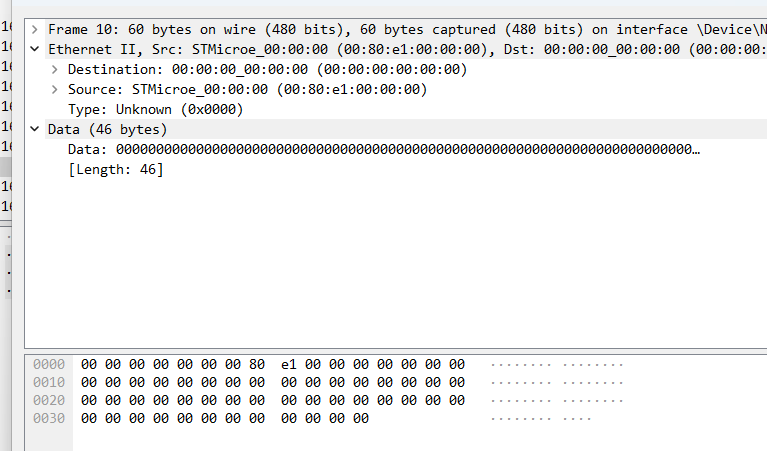
在王师兄飞线以后，按道理硬件上的问题已经全部解决，但是还是ping不通。继续单步调试吧，在low\_level\_input()函数里面发现了每次进入接收函数都是正常的，但为什么就是ping不通，那只剩下发送了。打开了TxBuffer的内存，发现





这一帧以太网帧完全正常没问题啊，于是只能考虑电脑接收端的情况了。这里使用wireshark抓包软件进行分析

可以接收到STM32发来的数据，但是本该属于目的MAC地址的地方全变成了0，王师兄又看了下，发现发送函数前少了一步清除Cache缓存。。。

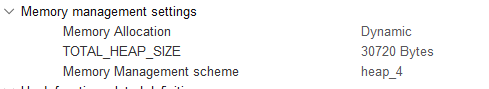


分析到这就合理了，内存中的以太网帧数据是没有错的，但是用ETH外设发送的时候，CPU直接读取Cache里面的值，而Cache里面的值是没有更新的！所以出现了前面全是0的情况。又是Cahce的问题，所以这里需要在发送前清除一下缓存，或者直接DCache透写，即强迫CPU读取内存。



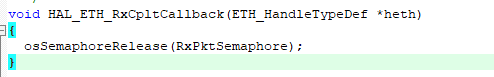
1. FreeRTOS + LWIP

在多线程情况下，LWIP的数据处理部分是会被创建到一个线程里面去。CubeMX里面开启FreeRTOS，配置一般选择默认就足够我们使用。这里分配的堆栈给大一点，否则任务过多就会产生SEGMENT FAULT。特别是在开启LWIP的情况下。

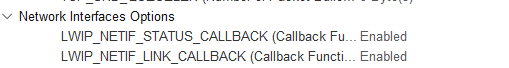




有操作系统下需要使能中断，前面我们说LWIP数据处理是放在任务里面。这个任务并不是一直运行，而是获取到信号量才会运行。中断回调函数里面会释放信号量，所以要把其打开。

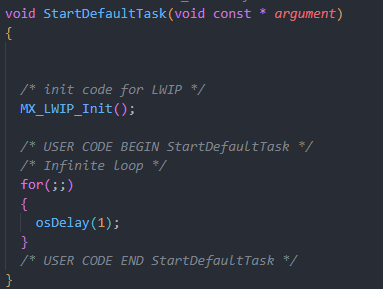


LWIP里面也是使用默认配置即可，这里面把这两个使能，可以开启网线的热插拔功能。建议使用静态IP，DHCP分配时间长，而且每次IP还不固定。

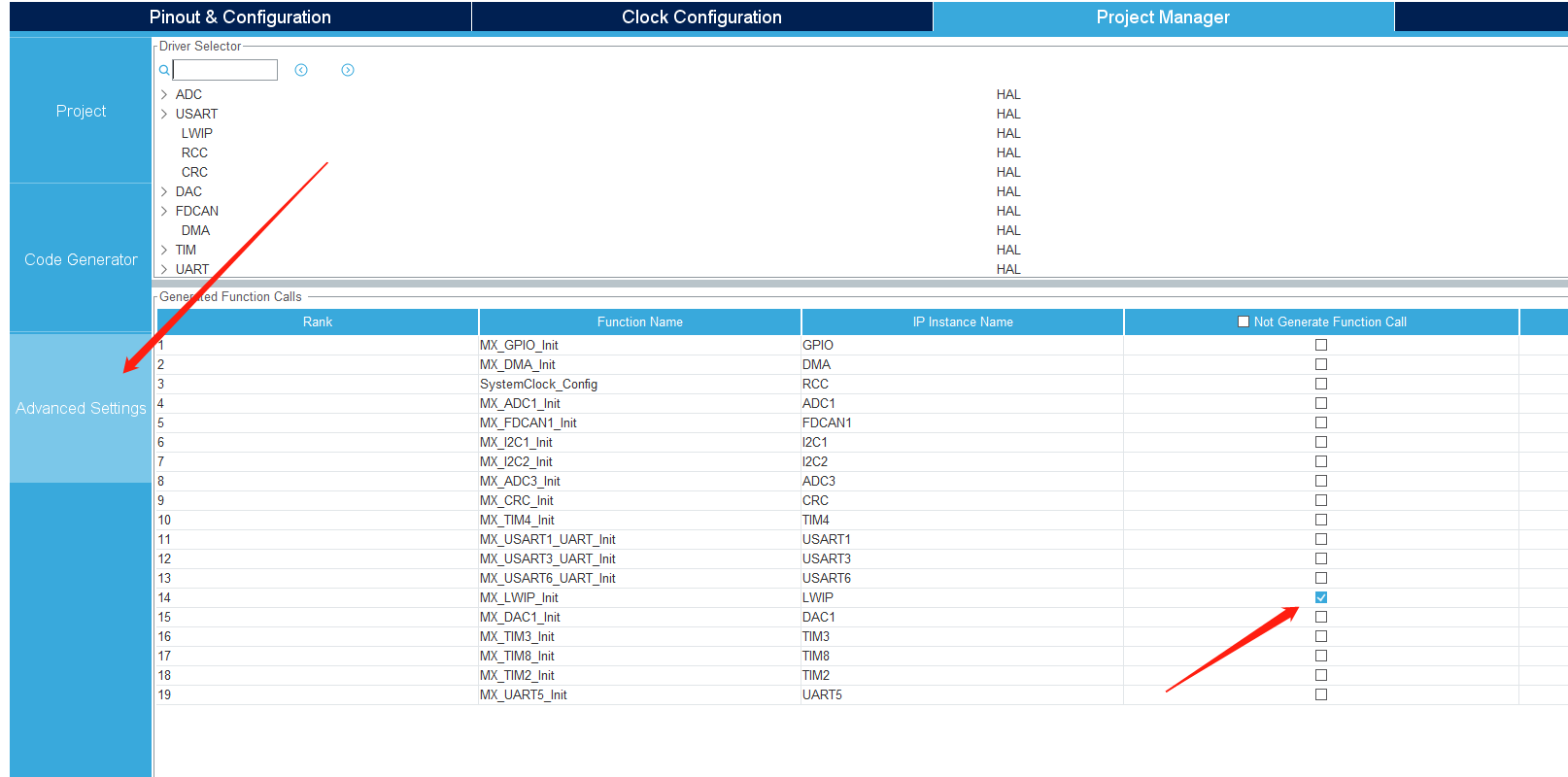


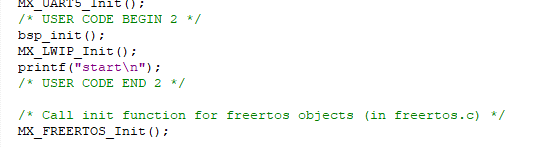
* 1. 关于CubeMX的坑

由于笔者这里所用的CubeMX的版本为5.3.0，按照网上教程到这里应该说就已经能跑通了。但是在实际TCP通信的时候，却老是卡死在一个很底层的地方，因为有个指针是空。经过单步调试发现，是LWIP的内存块没有初始化（具体可看LWIP的内存管理相关的资料），导致申请的指针为空。具体来说就是mem\_init() 和memp\_init()没有被调用，而这两个函数的调用逻辑是MX\_LWIP\_Init()-> tcpip\_init()->lwip\_init()->mem\_init() 这说明MX\_LWIP\_Init()没有被调用，经过检查发现CubeMX自动生成的代码里面，这个函数被放在了空闲线程里面，而我们socket那个线程的优先级是要高于空闲线程，FreeRTOS的实时性，会导致系统优先调用socket线程，而其又找不到TCP块，所以卡死。



那我们不把其放在空闲线程内即可，LWIP配置不显式生成MX\_LWIP\_Init()，自己在主函数freertos初始化之前手动加上即可。





* 1. 一些小问题

在开启TCP的时候不要开代理，否则就会出现ping的通但是TCP连接不了的情况。具体原因笔者没去了解，推测可能是端口占用？