# 进展日志：

1、20190814：V10的版本（由汇聚节点程序版本裁剪）存在问题如下：多个任务中始终只执行第一个任务，其他任务不执行，尚未查出原因；（现在查明就是内存不足，即分配给操作系统的内部不足，可以在FreeRTOSConfig.h中修改configTOTAL\_HEAP\_SIZE大小）

2、20190814：V10版本在原先汇聚节点程序版本上重新裁剪（解决了V10版本的问题，其实不需要重新裁剪），增加串口输入任务（允许发送中断，实现后台发送数据的功能）

（1）目前串口中断优先级是7，属于RTOS管理的，即存在该中断被关闭（例如临界区），从而丢失接收数据的隐患；如果把中断优先级提高（即不被RTOS管辖内），则中断函数中不能使用RTOS的API函数；

（2）串口接收是通过DMA方式和串口总线空闲中断（实现自动断帧）来完成的；

3、20190815：

（1）实现串口的收发都是通过DMA方式，其中发送并没有使用中断（包括串口中断和DMA中断）；接收使用了DMA方式和串口总线空闲中断（实现自动断帧）；

（2）在串口接收DMA方式在串口总线空闲中断中需要与DMA串口发送一样清楚完成标志位，否则不会进行第二次DMA的接收；

4、20190816：

（1）更新了业务数据上传方式，不再区分长短消息；

（2）已经完成传感器业务数据接收及串口转发；

5、20190820：

（1） 完成了串口的编写，并调试了串口协议；

（2）目前连续两天以上拷机测试，程序正常；

6、20190821：

（1）把malloc的代码删除，节省了RAM空间，目前不到90K；

（2）传感器侧的参数设置：直接增加0xFE的类型，增加到6个参数（新增频点和门限）；

6、20190822：

（1）改正寄存器中黑白名单数量的寄存器赋值（原来的BlackSensorNode\_List.Nodenum\*2,其实不用乘以2）；

7、20190903：

（1）针对硬件新版进行了修改：只是SX1280的IO和中断需要修改；

（2）门限代理改为6个字节：高字节在前，低字节在后；

（3）代理里面增加两个字节的备用（也即参数设置）；

（4）写代理和代理格式进行修改（首字节改为传感器类型）；

（5）修改经常进入硬件错误死循环的问题：具体描述参考见“问题记录”的第4点；

8、20190906：

1、0903版本在连续三天运行，没有出现死机现象（串口在工作），但是出现了SX1280不接收数据的现象；

2、在SX1280\_SENS\_Pro函数中，空闲和default都设置接收状态；

3、修改了串口中断中消息队列发送函数中第三个参数，不应该是0；

4、在0906+1版本中，SX1280RxData\_Task中修改了等待中断的超时时间由最大值改为30s，同时修改了调用函数SX1280\_SENS\_Pro的策略。在0906+2中，SX1280\_SENS\_Pro函数中default中增加了时间lorasteptime变量的判断。

9、20190929

1、ID由4位改为位6位

2、修改了白名单结构；

3、修改了微功率传感器参数设置方式：说明如下：

在数据规约中的5.5中修改如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型值 | 类型名称 | 长度 | 类型 | 单位 |
| 1 | 数据采集周期 | 4 | 无符号整型 | s |
| 2 | 数据上送周期 | 4 | 无符号整型 | s |
| 3 | 控制周期长度 | 4 | 无符号整型 | 上送周期 |
| 4 | 业务频点编号 | 1 | 无符号整型 |  |
| 5 | 振荡时间范围 | 1 | 无符号整型 | 5ms |
| 6 | 时隙编号 | 4 | 无符号整型 | 从1开始 |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| .. |  |  |  |  |
| 0x3FFF | 数据上送周期+控制周期长度+业务频点编号+振荡时间范围+时隙编号 | 14 |  |  |

其中，301模块根据时隙编号计算成延时时间下发给微功率传感器

4、修改了代理，及部分串口协议。

10、20191005：

1、增加了串口4的设置功能，即在原来调试通道上增加串口协议功能；

2、增加串口发送数据时的交互，即串口发送数据后，要求对方回应，否则需要重发，保证数据不丢失；

3、增加flash的参数保存

4、增加串口协议中的参数设置与保存；

11、20191014

1、301通过串口设置参数给203模块；

2、测试了flash的数据保存；

12、20191010

1、根据20191008的协议，修改了参数设置：

（1）业务周期单位由s改为ms;

（2）控制周期字节由4个改为2个；

（3）增加通信指令指示：

（4）控制请求分：所有控制参数，通信指令参数，业务控制参数；

（5）控制请求应答：分通信指令参数、业务控制参数；

（6）数据格式问题：控制请求不按规约；控制请求应答中的通信指令参数不按规约，业务控制参数按规约；其他帧都按照规约；目前传感器端只需要实现通信指令参数；

2、代理需要跟着修改，并且要flash保存；

12、20191021

1、设置上下限值按照数据规约的参数设置走；（未测试）

2、为适应非中继型汇聚节点，301模块的业务数据串口改为com1，而com2继续保留（只是从无线接收的业务数据不从com2（改为com1）输出给203模块）；

3、通信参数（rsp）和控制参数（rsp\_end）同时回给微功率传感器；

4、操作系统任务总堆栈由60K扩大到70K，否者串口4的发送任务不执行；

5、目前的RAM使用在105K左右，最大128K，另外有64K的还没有用。

6、修改写代理函数（修改了串口协议，读写入黑白名单需要区分带不带参数），只是修改WhiteList\_ADDID函数中，如果某个参数不为零则覆盖原参数，否者不覆盖；

7、修改微功率协议中控制请求回应函数，需要根据查询参数是否等于0，来决定是否下发该参数；

8、修改了串口协议，增加了上下限设置（按照数据规约）

13、20191027：

1、本程序作为泰州和河北第一批汇聚中301模块的程序版本；

14、20191111

1、修改串口协议，增加传感器注册请求与响应处理。（未测试）

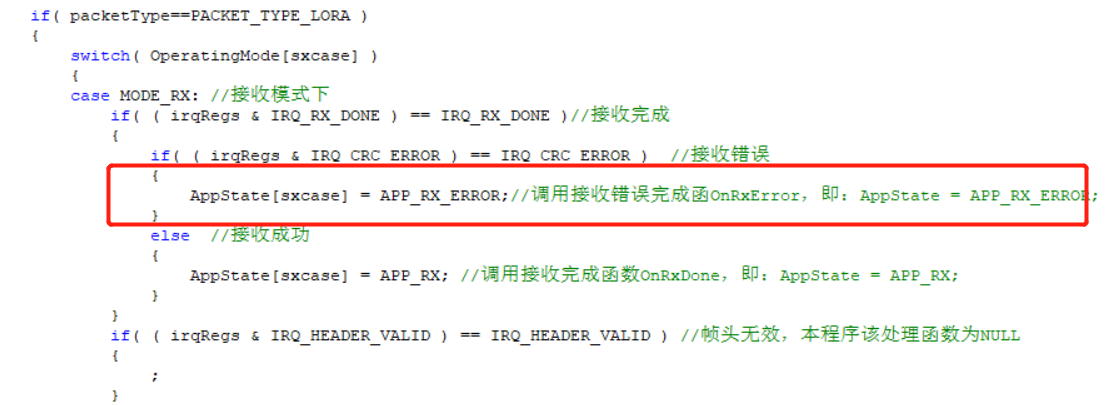
2、问题：原先校验和出错的问题，只是因为中断开启中没有CRC校验

uint16\_t RxIrqMask = IRQ\_RX\_DONE ;

只需改为如下：

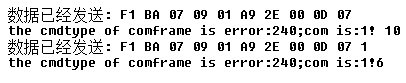
uint16\_t RxIrqMask = IRQ\_RX\_DONE |IRQ\_CRC\_ERROR ;

则将报CRC校验错，即在函数SX1280ProcessIrqs()中，如下图：



3、增加了看门狗程序；

4、问题：读取传感器代理时出错；（已修改）

5、问题：串口提示接收数据错误；

15、20191120

1、屏蔽一些串口打印，方便串口命令查询黑白名单；

2、问题：删除黑白名单有问题，在删除函数中如果只有一个名单，则没有判断ID是否相符就直接删除了；（已修改）

3、增加了发送告警数据时，查询非白黑名单时也增加发送注册请求；

4、暂时取消了Flash的存储（1、取消初始化的读；2、在STMFLASH\_Write函数直接返回，省却一个个屏蔽该函数）；

5、电感取消，程序选择LDO方式；

15、20191209

1、灯的状态：（1）LED1是在业务信道收一次数据（还未校验），则亮灭切换；（2）LED0是在串口向203模块发送一次，则亮灭切换；注意在第一次接收传感器数据后串口会连续发送两次数据（注册与业务数据），故LED0似乎不闪烁；

2、1280的数据接收任务，超时由原来的5s改为最大值：这样如果长期接收不到数据，则就不会发送喂狗，则就使得看门狗超时；

3、看门狗策略调整：（1）1280接收任务阻塞时间，改为最大时间，如果接收到数据，则发送对应事件标志；（2）串口接收任务，如果接收到数据，发送事件标志；（3）看门狗任务查询到1280接收任务标志&串口接收任务标志，才喂狗；（4）看门狗任务连续运行20次以内每次喂狗，20次以后则不喂狗，这样就会触发重启；

4、只能先发数据，再发注册，因为缓冲会冲突；

16、20191210

1、BlackList\_CheckID函数中，如果名单数量超限不应该返回false；

2、LED0的闪烁由原来的串口发送203改为从203接收到回应命令后闪烁；LED0是离JTAG最远端的那个；

1. V20\_20200114
2. 在20191210版本基础上进行改进：基于异频组网的架构下，串口协议与Relay\_Master一致；
3. 目前RAM使用情况如下:RAM1：总128K，已使用110K；
4. V20\_20200115
5. 完成IO判断进行串口发送，从调试信息看正常，上传数据正常，需要用示波器进一步确认；
6. 完成微功率传感器的注册流程：目前RAM总使用110.95K，其中RAM1使用87K（128K），RAM2使用23K（64K）
7. 完成了串口发送的同步仲裁：（原来主要是打印花费太多时间导致IO一直被拉低）；
8. 完成了微功率传感器参数设置与查询，已单测；
9. 完成黑白名单设置功能。
10. 完成网络命令应答：名单设置与参数设置都需要应答

7、汇聚的config在flash中的保存，目的是保存汇聚ID。

8、目前内存使用情况：（RAM1和RAM2都打开的情况）

Total RO Size (Code + RO Data) 60008 ( 58.60kB)

Total RW Size (RW Data + ZI Data) 113612 ( 110.95kB)

Total ROM Size (Code + RO Data + RW Data) 60420 ( 59.00kB)

1. V20\_20200216

1、黑白名单数据结构统一修改为传感器节点列表结构：

（1）增加注册状态字节和黑白属性字节；

2、修改与列表结构有关的程序；

3、增加了心跳功能；

4、增加部分网络指令（非私有）

1. V20\_20200218
2. 按照《3-新架构master通信接口\_20200207》，即把原来串口接发存网络包的数据也需要打包成参数设置格式；

2、针对串口缓冲区长度改为1500后，程序会进入硬件错误死循环（即所谓B处）的问题：

（1）在没有改动 前提下：修改USART2\_REC\_LEN和USART2\_TX\_LEN为800字节，同时增加Uart2Tx\_MessageData\_Q\_NUM即消息队列数量（由5改为10），程序仍能正常运行，表明不是操作系统占内存的问题，即进一步增加configTOTAL\_HEAP\_SIZE是没有用的；

（2）改USART2\_REC\_LEN和USART2\_TX\_LEN为960字节为临界点，超过后串口会有错误提示：“Error:..\FreeRTOS\tasks.c,2806”

（3）把USART2\_REC\_LEN和USART2\_TX\_LEN改为1450后，把main.c中的串口收发任务（共6个）中原定义在函数中的局部变量数组uint8\_t BSbuf改为全局变量，即可；而且不能6个函数单独用6个全局变量，至少串口1,2的4个收发任务合用收发各两个全局变量，否则会导致串口4的接收不正确。（目前是这个现象，原因暂不明）。

1. V20\_20200224

1、20200218版本中第2（3）点中提到的串口4接收不正确，是以为config.h中系统堆大小由75K改为85K即可；

2、针对20200218版本中第2点中的程序会进入硬件错误死循环（即所谓B处）的问题：即把main.c中的串口收发任务（共6个）中原定义在函数中的局部变量数组uint8\_t BSbuf改为全局变量，即可。原因是函数中的用到局部变量太大导致系统问题（任务切换可能会有些问题）。且该问通过定位进入B处前的程序代码也不行（问题记录4中提到的两个方法）。

1. V20\_20200225

1、修改了由串口2调整为串口1作为通信口用（适用只有一个203（Relay\_Client））；

2、修改名单在调试口的打印；

3、修改了传感器数据注册和告警数据的发送；

1. V20\_20200226

1、修改了网络数据不需要新协议；

2、测试看门狗，在让Relay\_Client通道无法同步工作的情况下，第一次可能需要4分钟（20）,后续就是2分钟；

3、在config中增加Relay\_Client的工作状态标志，即在串口没有收到包含有本汇聚节点ID的信息时，串口不发送任何数据给Relay\_Client；

4、如果名单满了，不能再发传感器注册申请。

5、修改了串口发送大数据的问题：发送任务函数中变量taski类型错误；

6、通道号由2改为3；同时修改了心跳的错误（通道号，和起始帧需要两个字节）

1. V20\_20200227

1、串口2发送任务查询到串口空闲时，没有立即锁定串口(拉高电平)，而是再确认是否有数据发送，然后再锁定串口进行发送，如果其他通道在这个期间发起串口发送则会引起错误；

已解决：目前通过先查询队列是否信息，然后再查串口是否空闲，空闲则立即锁定串口；

1. 下发路由表的功能procNetCmdSensorRoute函数，通道号需要改为3，同时根据命令情况需要增加上行错误应答，后都修改为正确应答；

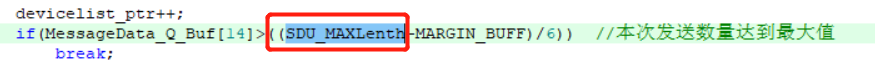
3、黑白名单操作：根据命令情况需要增加上行错误应答，后都修改为正确应答；

4、列表结构中增加注册不通过次数，在接收到业务数据时，判断该ID在列表中如果属于注册不通过，则需查看注册不同次数，超限则发起注册，不超则次数加1；（未测试，可通过修改程序把注册通过改为不通过，则可以看出是否会再次注册）

1. V20\_20200228

1、修改了下属传感器通信参数查询的BUG，即有效参数数量与被查询数量不一致时的问题；以及最大扰动的参数类型被屏蔽掉的问题；

2、修改查询设备列表函数procNetCmd\_C5\_DeviceListQuery中的BUG：



3、增加了私有指令C4查询设备参数（与下属设备通信参数同）；（未测试）

1. V20\_20200303

1、修改微功率传感器通信参数里面的延时时间参数，不需要进行本地计算而是直接转发即可；（已测试，同时单独配置业务周期时多发了一个字节的参数类型）

2、在对微功耗传感器进行通信参数设置时，通过memcpy方式进行参数传递，即上面下发（接收时也用memcpy）什么模式就转发什么模式，不进行大小端转换。（已测试）

3、在对微功耗传感器进行控制参数设置时，通过memcpy方式进行参数传递，不进行大小端转换。暂时节点网络没有控制参数的下发；（未测试）

4、添加黑名单后，同时需要把该黑名单的注册标志清零；

5、把列表数量由1000改为10后，串口出现问题：（1）主要原因是DMA所用到的串口数据变量（接收发送缓冲区）被分配到CCM（0x10000000起始64K）的RAM2中了；两种解决办法：把RAM2禁用（显然不合理）；把usart.o所用的变量指定分配到RAM1中(0x20000000起始)，参考“问题记录中CCM的使用”（顺便把FREERTOS的变量优先放在RAM2中）；

1. V20\_20200312

1、把看门狗事件标志中，策略由1280接收到数据改为1280正确接收到业务数据；同时增加串口发送事件标志，用以监测串口发送任务是否正常；

2、增加私有指令的3通道重启功能；

3、增加3通道存储设备列表的功能：（出现硬件错误死循环问题）

（1）首先如果列表存在RAM2的区域，列表数超100就会硬件死循环；

（2）把列表存在RAM1的区域（修改SCT文件），列表数一定数量（750）以内正常，超过则有问题，原因可能是：经计算是整个数据长度超过扇区的128K（4倍大小问题，即数据长度超过了32K）；

4、增加私有命令C1通道工作参数查询，但是回复是空指令；

5、出现死机问题：描述：LED0正常5s闪烁一次（离JTAG最远），表明看门狗任务正常，但是串口打印表明串口发送队列满（心跳数据发不出），表明串口1发送任务不正常，同时LED1灯不闪烁，表明SX1280接收数据不正常；

分析：（1）堆栈内存不足，导致某些任务在运行一段时间后会挂掉；（2）有内存或者硬件接口被共享使用，导致低优先级的任务不被运行；

测试：（1）把OS堆栈由85K改为90K，名单数量由750改为700个；

6、优化了程序：（1）把任务堆栈大小为256不变；（2）SX1280的控制和业务两个任务添加了vTaskDelay；（3）串口1的接收任务延时改为10；

7、修改了debug口查询列表的错误（有switch语句没有break）;

8、SenBS\_CH1\_DataPro函数中最后返回需要是6；

9、在SX1280的业务和控制通道处理接收数据时，需要剔除长度为0的数据；

1. V20\_20200317

1、器件列表中有ID为全0（或者全F）的微功率传感器；

这是因为接收到了全为0的数据；（已修改，在处理接收数据函数中剔除长度为0的数据）

2、把看门启动放在程序的一开始，同时在接收到正确的业务数据后才发送看门狗监控标志（EVENTBIT\_SenBS\_CH1\_TASK）

3、增加了任务堆栈的监控；

4、看门狗喂狗时间改为接近6分钟；

5、增加调试串口清设备清单的命令（未完成）

1. V20\_20200320 V2.0.3.25

1、增加调试串口清设备清单的命令（完成）

2、在SX1280HalWaitOnBusy函数中，添加BUZY管脚超时打印；

3、USART1\_Snd\_Task中接收队列数据等待时间由原来的portMAX\_DELAY改为0；

4、增加注册任务，方便在一个SDU中包含多个ID，但同时没有增加该任务的看门狗事件；

5、当前RAM使用：145.71K

6、发送注册消息队列时需先查询串口心跳是否正常；

7、删除SPI1的初始化；

8、if( ( irqRegs & IRQ\_HEADER\_VALID ) == IRQ\_HEADER\_VALID ) //帧头无效 修改为APP\_RX\_ERROR，原本是不处理的；

9、降低在SX1280\_SENS\_Pro函数中，APP\_RX\_ERROR情况下的延时：由1000改为0；

10、在SX1280接收数据任务和控制任务中读写SX1280函数中添加临界段代码保护；

11、if(lorasteptime[sxcase]>100\*1000)//超过100s，重新继续设置为接收模式，同时把timeout和error的情况下都把lorasteptime[sxcase]清0；

12、确定为正式版本号V2.0.3.25

13、SX1280RxData\_Task等待事件标志不再是portMAX\_DELAY，而是60\*1000；

1. V2.0.3.26

1、修改版本号，支持私有命令查询版本号。

2、调整串口抢占时的随机延时时间；（原来的方法有些时间太长了）

3、North\_UL\_isEnable函数中增加临界代码保护；

4、串口1任务系统等待时间有10改为5；串口发送任务发送完成后添加等待500us；

5、修改一下串口速率

6、

问题描述：串口1没有发送数据导致汇聚不断重启，平台看不到微功率传感器数据；

现象分析：

（1）两个LED灯闪烁表明SX1280业务员数据接收正确，看门狗任务正常；

（2）调试口打印正常，平台查询通道状态时调试口有输出，表明下行通道正常；

（3）IO2看不到波形，表明串口1发送任务未正常执行。

（4）调试口没有报串口发送队列缓冲满报警，说明没有持续写队列操作，而SX1280读正常，则只有一种可能，即RelayClient\_status（接收心跳标志）没有被置位，而下行心跳数据正常，则表明才标志参数置位逻辑有问题；

复现：0通道断电超过2分钟（监测心跳数据超时时间）后重新上电，现象重复；

解决：（1）改进RelayClient\_status（接收心跳标志）置位逻辑；

1. V2.0.3.27
2. 针对3.26版本问题，修改如下：

（1）RelayClient\_status的处理，在接收到心跳后即置位该参数；

（2）看门狗策略：

原策略是在5s内同时有3事件（SX1280接收到业务数据，串口发送，串口接收）发生才喂狗，这个概率有些小；而看门狗是20分钟内不喂狗即重启；

由于SX1280接收到业务数据后会立即进行串口发送，串口接收心跳是1s周期的，故5s内同时发生在20分钟内还是可行的，因为20分钟没有收到SX1280业务数据也可以重启了；

1. 把串口发送任务中，发送完成后延时1ms再释放串口，看是否有撞包问题；
2. V2.0.3.30

1、在名单中注册未通过的ID进行重新注册时，没有查询心跳是否正常；

2、查询接收心跳时，时间戳重新赋值有问题（变量拷贝错误）；

3、看门狗策略调整：

(1)原策略是在5s内同时有3事件（SX1280接收到业务数据，串口发送，串口接收）发生才喂狗，这个概率有些小；而看门狗是N(WATCHDOG\_MaxTime参数决定)分钟内不喂狗即重启；

(2)由于SX1280接收到业务数据后会立即进行串口发送，串口接收心跳是1s周期的，故5s内同时发生在N(WATCHDOG\_MaxTime参数决定)分钟内还是可行的，因为N(WATCHDOG\_MaxTime参数决定)分钟没有收到SX1280业务数据也可以重启了；

(3)V2.0.3.30中改为N(WATCHDOG\_MaxTime参数决定)分钟，3事件只要都发生过一次即可（在读事件标志时，标志不清零，只有在3件事都发生后清零）

1. V2.0.4.02
2. 从V2.0.3.26~30这几个版本中因为函数North\_UL\_isEnable中增加临界代码保护有逻辑问题，即在if(North\_UL\_islock()==true)成立时没有退出临界保护，导致如果串口仲裁IO长为低时，程序一直在执行串口1发送任务，无法调度到其他任务中；
3. 串口仲裁IO线上拉电阻如果为10K，接上示波器探头，则即使只有301模块的情况，也容易North\_UL\_islock()==true成立：（1）如果把IO初始为推完输出，则不会（从示波器看，数据传输频率会下降）；（2）如果把上拉电阻改为3.9K则也不会；
4. V2.1.4.02

1、本版本作为第一批正式版本发布：

（1）串口1；

（2）看门狗时间10分钟：即该时间内接收到正确的业务数据，串口接收心跳数据或命令数据，串口发送数据这3个任务必须都执行过一次；

（3）重复注册的时间设置：30次业务周期，

（4）名单数量：700；

（5）板间监控心跳：30秒；

（6）版本号：硬件V1.0.0.0 软件：V2.1.4.2

1. V2.1.4.16

1、增加IO3（与LED1功能同）

2、修改重复注册的时间策略：原先是用业务数据周期，但是不同传感器的业务数据周期不同，导致重复注册时间不同，先修改为绝对时间；

3、修改了频点设置方式，不再用频点号查询列表方式改为根据频点号公式计算；

4、发送注册请求串口数据时，数据长度有误（没有减2）；

1. V2.1.4.16（直连主节点）（服务于合作公司：合智、方天）

1、在版本V2.1.4.16基础上修改；

2、串口波特率115200\*4；

3、没有下行心跳（看门狗事件，D\_DID, RelayClient\_status等需要修改）；

4、不需要串口竞争；

5、增加串口接收缓冲区清零；

6、所有串口数据需要按照《3-新架构master通信接口\_20200207.docx》格式；

1. V2.3.0.0（直连主节点）（服务于合作公司：合智、方天）

1、私有命令C0的保存：（1）需要处理所有通道；（2）保存对象字节位置似乎有误；（和汇聚版本V2.1.5.13一致；）

# 当前版本的问题：

1、暂时节点网络没有控制参数的设置与查询

2、20200303：

（1）程序问题描述：LED0正常5s闪烁一次（离JTAG最远），表明看门狗任务正常，但是串口打印表明串口发送队列满（心跳数据发不出），表明串口1发送任务不正常，同时LED1灯不闪烁，表明SX1280接收数据不正常；

分析：（1）该问题确认为：在读写SX1280过程中被中断（或者任务被调度），在中断函数（或者高优先级任务）中又对该SX1280进行了读写引起了该问题，因为在第一次读写SX1280过程未完成，SX1280是把BUSY管脚拉高（表示忙），所以在中断函数（或者高优先级任务）中再次读写操作时，查询BUSY管脚一直为高导致陷入死循环。该现象通过有意设计上述流程的测试程序中可以容易复现；

解决：（1）在读写SX1280的代码前后增加临界代码保护，该策略在测试程序中已经验证有效，同时该策略在实际程序中也经过了十几天的拷机，未再出现死机问题；

# 发布新版本时需要注意的问题：

1、看门狗允许及看门狗设置的时间。

2、串口选择：串口2，串口1；

3、重复注册的时间设置

4、名单数量

5、在SX1280HalWaitOnBusy函数中，添加BUZY管脚超时打印；

6、板间心跳监控时间：目前是2分钟；

7、版本号

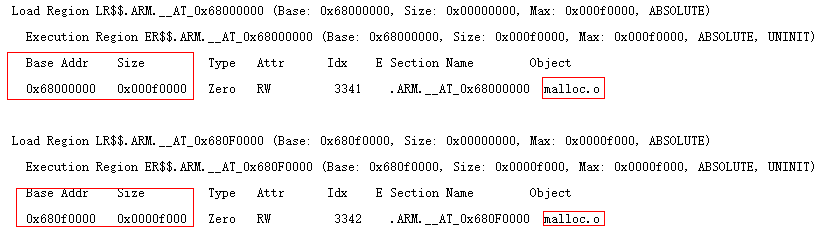
# 问题记录：

## 1 STM32F407内存问题

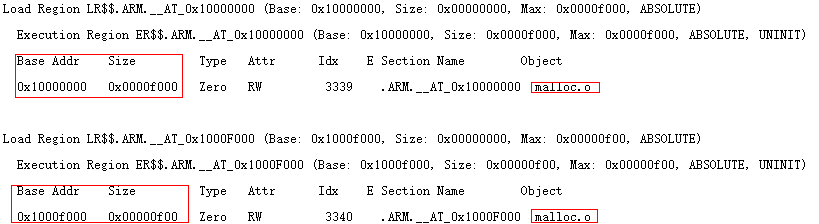
1、STM32F407VET6集成共有128K(普通SRM)+64K(CCM)+4K(??RTC)=192K

2、正点原子的malloc.c中内存管理分配情况如下：预定义并占用了内存

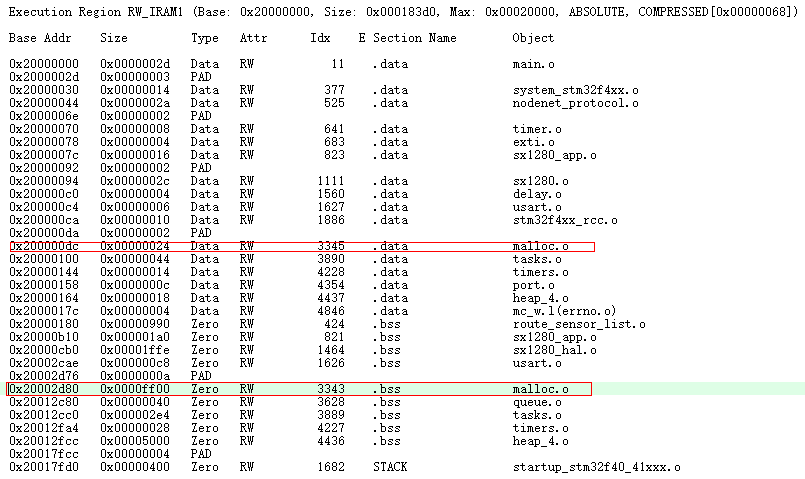
（1）外部SRAM：起始地址0x68000000，分配了960K，对应MAP文件如下：从MAP看，malloc.c实际占用了960K(0XF0000)+60K（0Xf000）=1020K，而实际使用了0字节；



（2）内部CCM：起始地址0x10000000，分配了60K，对应MAP文件如下：从MAP看，malloc.c实际占用了60K(0XF000)+3K（0Xf00）=63K，而实际使用了0字节；



（3）内部普通SRAM：起始地址0x20000000，分配了60K，对应MAP文件如下：从MAP看，malloc.c实际占用了63K(0XFF00)+36（0X24）=63K，而整个程序使用了约96K的SRAM；



3、从上面三图可看出，总RAM使用情况如下：

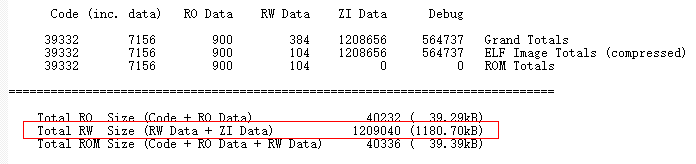
普通SRAM占用了96K：其中包括malloc.c用了63K，其他程序使用了33K，还剩下128-96=32K可用；

CCM占用了63K：全部为malloc.c使用；

外部SRAM占用了1020K：全部为malloc.c使用；

共计:96+63+1020=1179。

对应编译统计结果如下：即1180.70K，与前面计算的1179K基本吻合，



以上测试都是基于0422版本；

20190821：

1、把malloc的文件夹下函数取消后（用不到内存分配相关功能），整个程序在81K内存，全部是使用的是内部普通SRAM，也即内部CCM的60K内存没有被使用；

## 2 发送后如果对方回复太快，来不及接收。

（20190620：通过减少控制信道任务中事件堵塞时间，已解决）；

## 3 SX1280的SX1280GetPayload函数，如果接收实际长度大于接收缓冲区长度，则直接返回1，导致接收缓冲区数据没有更新；

现改为填充完缓冲区最大长度后返回；

2、轮询回复的时候，如果一帧需要多条数据，则最好能同一类型的数据；

3、黑白名单增加时需要先查表确认是否已经在其中；（20190620：已修改）

4、添加黑白名单时，需要同时删除对应白黑名单（20190620：已修改）；但是删除黑白名单时，是否要添加对应白黑名单（暂定删除时不需要添加）；

5、需要添加同类传感器的参数：业务周期，控制周期，报警上限；

6、测试增加白名单（20190624：已测试）

## 4 FREERTOS中的入队函数，如果入队的数组长度小于队列的长度，是如何操作的，是不是仍然按照队列的长度进行入队的；因为入队函数的参数中没有长度这个形变；

## 4 出现进入硬件错误死循环（即所谓B处）。

1、查错方法：

https://blog.csdn.net/electrocrazy/article/details/78173558

前提是：能停在死循环处，以便能查出进入死循环前的代码地址；

https://www.cnblogs.com/Ilmen/p/3356147.html

2、错误原因：

（1）本次错误是由于结构指针的问题，定义了一个结构指针，然后对该指针直接赋结构成员的值。应该是定义结构变量，然后才能直接赋值给结构成员。如果只是定义了指针，那么一定要把该结构类型的变量赋值给指针。

（2）上次错误主要是中断开启没有相应的中断函数；

（3）出错原因总结：

◆数组越界操作；

◆内存溢出，访问越界；

◆堆栈溢出，程序跑飞；

◆中断处理错误；

## 5 freeRTOS某个任务不被执行。

1、经测试，由于freertosconfig.h中系统所有总的堆大小太小的情况下，某些任务似乎不被执行；故需要扩大堆栈大小；例如本例程中堆栈已经到70K了，如果设置为60K，则串口4的发送任务就不被执行；

## 6 CCM的使用

1、参考资料：

（1）https://blog.csdn.net/u012625379/article/details/90517082 （修改SCT文件）

（2）http://www.stmcu.org.cn/module/forum/thread-604814-1-1.html （REERTOS用CCM）

2、参考本例程的SCT文件，如下图：

