# 测温项目数据接收中继软件调试记录

## 2019/10/23

### 1 程序中使用rt\_free报错（李磊）

|  |
| --- |
| **问题/现象详细描述：**关于问题或现象的详细文字描述 |
| ((((rt\_ubase\_t)rmem) & (RT\_ALIGN\_SIZE - 1)) == 0) assertion failed at function:rt\_free, line number:555 |
| 出此错误时，程序中新添加了如下部分： |
| 其中vPort\_s2j\_init()函数中，将rt\_free进行了赋值给其他变量 |
| **分析原因及解决办法：**怀疑是不是rt\_malloc()函数或者rt\_free()函数不能使用，但是单独实验这两个函数，发现并没有报错。 |
| 进行但不调试，发现程序是出错在Get\_JsonFile()函数中的这一步： |
| 往上查找程序发现，如果打不开这个文件，就不会有机会给string分配内容空间，也就不能使用free进行释放堆内存 |
| 修改程序如下： |

### 2 创建文件，重启之后文件不存在（李磊）

|  |
| --- |
| **问题/现象详细描述：**在程序中创建新的ConfigFile.json文件，在重新开机之后，文件就又需要重新创建，查看目录下，发现确实已经创建好了： |
| **分析原因及解决办法：**将创建文件的程序注释，重新开机，查看是否有刚才已经创建好的文件，发现并没有 |
| 突然想起来，每次开机程序开始都会格式化Flash，可能是这个导致的，所以将格式化Flash的程序注释掉，重新执行。 |
| 然后重新下载，程序好使了。 |

### 3 ModbusSlavePollThreadEntry 进程堆溢出（王刚）

|  |
| --- |
| **问题/现象详细描述：**添加了设置Modbus通信参数的相关内容，运行后提示堆溢出 |
|  |
| **分析原因及解决办法：**修改该进程的堆大小由原来的512修改为1024 |
| 修改后程序正常运行 |

### 4 文件系统中存在json文件时，finsh运行不正常（李磊）

|  |
| --- |
| **问题/现象详细描述：** 当文件系统中不存在ConfigFile.json配置文件时，程序运行正常，finsh命令行可以正常使用，但是当在文件系统中创建ConfigFile.json配置文件时，程序开始运行之后没有死机，但是finsh命令行不能正常使用。finsh命令行卡在如下界面： |
| \ | /  - RT - Thread Operating System  / | \ 4.0.2 build Oct 23 2019  2006 - 2019 Copyright by rt-thread team  sys freq: 180000000  pclk2 freq: 90000000  spi configuration  finsh />mount success!  Get ConfigFile.json Success. |
| **分析原因及解决办法：**在Modbus相关的线程任务中，怀疑是其轮询没有延时，导致一直不释放CPU，而且其优先级较高导致(优先级为10)。 |
| eMBInit(MB\_RTU, g\_ModbusSlaveAddress, g\_ModbusUartNumber, g\_ModbusBandrate, MB\_PAR\_NONE);  eMBEnable();  while(1)  {  eMBPoll(); //FreeModbus从机不断查询  } |
| thread pri status sp stack size max used left tick error  -------- --- ------- ---------- ---------- ------ ---------- ---  TaskData 11 suspend 0x00000164 0x00000400 39% 0x00000004 000  TaskTest 9 suspend 0x0000007c 0x00000200 24% 0x00000005 000  ModbusSl 10 suspend 0x00000084 0x00000400 57% 0x00000003 000  tshell 20 running 0x00000268 0x00001000 21% 0x00000001 000  tidle0 31 ready 0x00000044 0x00000400 08% 0x0000000a 000  main 10 suspend 0x00000084 0x00000800 32% 0x0000000a 000 |
| 所以在程序轮询后，加入一个短暂的延时，如下：  while(1)  {  eMBPoll(); //FreeModbus从机不断查询  rt\_thread\_mdelay(1);  } |
| 问题解决！但是考虑将Modbus轮询任务的优先级降低，而不加延时，导致其他任务可以进行抢占CPU，是否可行，对此进行试验尝试。 |
| 经过将Modbus轮询任务优先级改为20，去除任务死循环中的延时，发现，程序还可以正常执行，finsh也可以正常使用。 |
| 根据上述情况，猜测就是由于其优先级高，一直不去释放CPU，导致只有高优先级可以进行抢占，而低优先级无法执行导致的。 |
| **疑问：**之前Modbus任务的优先级为10，而且死循环中一直没加延时，为什么之前的程序没有出现这种错误。是否是由于其他高优先级任务在抢占了CPU之后，其他高优先级中有延时进行释放CPU，导致CPU重新调度，有执行到低优先级任务，所以之前没碰到该问题？ |

## 2019/10/24

### 1 在测试中发现程序未按照期望输出（王刚）

|  |
| --- |
| **问题/现象详细描述：**由于之前测试做的不全面，导致调试打印不正常，还有自已粗心导致放设备ID号的地址出错。 |
|  |
| **分析原因及解决办法：**未从循环中调出来，应添加break语句，之前测试时，按最后一个地址测试，在测试过程中一定要注意边值及特殊点的测试。 |
|  |
|  |
|  |

## 2019/10/25

### 1 在线调试中波特率的值没问题，打印输出是不是期望的值。（王刚）

|  |
| --- |
| **问题/现象详细描述：**在线调试中波特率的值没问题，打印输出是不是预想的值。 |
|  |
| **分析原因及解决办法：**  打印输出函数，由于引号位置放错，导致输出不正常。要注意避免粗心。 |

### 2 程序在运行中提示堆接近堆地址的末尾。（王刚）

|  |
| --- |
| **问题/现象详细描述：**在线调试中波特率的值没问题，打印输出是不是预想的值。 |
| \0  \ | /  - RT - Thread Operating System  / | \ 4.0.2 build Oct 25 2019  2006 - 2019 Copyright by rt-thread team  sys freq: 180000000  pclk2 freq: 90000000  spi configuration  finsh />mount success!  g\_ModbusBandrate001 = 115200  g\_ModbusBandrate002 = 115200.  参数下发成功.  Get ConfigFile.json Success.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address.  warning: SaveConf?stack is close to end of stack address. |
| **分析原因及解决办法：**  可能是堆内存分配少了，应加大该堆内存。由1024改为1536  static rt\_uint8\_t SaveConfigThreadStack[1536]; |

### 3 测试过程中，发现程序总是优先进去轮询任务，而不是保存设置任务。（王刚）

|  |
| --- |
| **问题/现象详细描述：**测试过程中，发现程序总是优先进去轮询任务，而不是保存设置任务。 |
| 115200为默认的参数，9600为根据保持寄存器0000H配置的波特率。 |
| **分析原因及解决办法：**  可能是由于两个任务优先级相同，优先进入了轮询任务。解决方案为降低轮询任务的优先级。由原先的20改为22。  /\* Modbus从机轮训的任务优先级，栈空间，任务结构体及入口函数 \*/  #define THREAD\_MODBUS\_SLAVE\_POLL\_PRIO 22 |

## 2019/10/27

### 1 将配置参数保存成json文件，之后从json文件中读取数据，作为配置参数失败。（李磊）

|  |
| --- |
| **问题/现象详细描述：** 将变压器运输平台中保存配置参数到json文件中，每次开机时，从文件系统中读取json文件，并解析json文件，将文件中参数设置到程序中。将相关文件移植过来之后，发现每次能保存文件成功，开机之后也能读取json文件成功，但是，打印配置参数，发现参数还是不对，没有改变。 |
| **分析原因及解决办法：** 经过单步调试，发现，程序中保存文件能正常保存，读取文件时，也能正常读取到文件，但是在文件解析时，就会解析出错。 |
| 解析文件出错，可能有两种原因导致，一种为保存文件的程序有问题，导致保存的文件格式不正确；一种为解析文件的程序有问题，导致解析不正确。 |
| 由于文件保存在STM32的Flash中，无法将文件读取出来进行查看，所以将该部分保存，解析文件的相关文件，单独移到windows上，创建简单的C语言程序，进行单独测试。 |
| 先测试保存文件的程序，运行完毕之后，打开保存的json文件进行查看，发现，保存的json文件格式存在问题。如下： |
| "ConfigFile":[  {  "name": "Modbus\_Baudrate",  "parameter": 0,  "number": 0  },  {  "name": "Modbus\_SlaveAddress",  "parameter": 0,  "number": 1  },  {  "name": "Modbus\_UartNumber",  "parameter": 0,  "number": 2  }  ]  } |
| Json文件最前边缺一个‘{’，所以导致json文件在解析的时候，一直解析不正确。所以接下来去json文件保存的程序中进行查找问题。如下 |
|  |
| 在此处，将{补充上，然后重新编译程序调试。 |
| 文件保存格式正确，并且解析也正确。 |