OCPU FAQ

FAQ 就是 Frequently Asked Questions（常见问题），本文档中记录一些客户经常遇到的问题，后续客户遇到问题，推荐用户先看看 FAQ，看自己的问题是否符合 FAQ 中的情况，如果符合的话 FAQ 中是有相应解答的，这样就不用人工服务了，节省了大家的时间。

SDK 初次使用部分

1、SDK 版本

1.1、客户从哪里获取我们 SDK 版本包和相关资料？

可以从我司 FAE 或者代理商获取到 SDK 的版本包和相关资料，亦可发邮件到 support@quectel.com 获取。

1.2、客户获取 SDK 版本包时需要注意哪些？

1）提供模块的软件版本号（ATI 可查询）给我司 FAE，SDK 版本包和软件版本包是需要对应的。

1.3、模块型号（按照软件版本分）

我司模块分为标准版本、纯 open cpu 版本、二合一版本
注：我司 NB 模组(BC26/BC66/BC20)A06 版本(包括 A06)开始都为二合一版本
标准模块:只支持标准 AT 命令，客户外接 MCU 通过串口向模块发送 AT 指令，实现信息交互。
eg：BC26A01-A03
纯 OCPU 版本:模块不支持标准 AT,客户可以在模块上进行二次开发，如果需要发送 AT 命令，也是通过 OCPU 传给模块底层处理。
eg:BC26A04-A05
二合一版本:同时支持标准模块和 OPEN CPU.
eg:BC26A06

特别提醒：

1、 不同软件版本对应不同的 SDK，不可以混用。
2、 NA/NB/NC/ND/NE 版本的模块软件版本不能通用，但 SDK 版本是一样的。
3、 不同版本支持的功能项不一致，客户需要按照自己项目需求选型模块,具体支持功能项可查询我司 RN

文件

2、SDK 编译、基本配置

2.1、GCC 编译器的版本号？

arm-none-eabi-gcc-4.8.3

2.2、如何安装 GCC 编译器

由于 NB 项目我们已经把 GCC 编译器打包到 tools 文件夹下，用户不需要再安装配置 GCC。

注：如果用户需要自己安装 GCC，可以参考我司 2G 模组配置

2.3、编译出错哪里查看错误信息？

客户查看 log 文件，路径：build\gcc\build.log

2.4、编译其他的 example 在哪里更改宏

\make\gcc\gcc_makefile

```
#-------------------------------------------------------------------------------
# Configure GCC installation path, and GCC version.
# To execute "arm-none-eabi-gcc -v" in command line can get the current gcc version
#-------------------------------------------------------------------------------
GCC_INSTALL_PATH=tools\gcc\win\gcc-arm-none-eabi
GCC_VERSION=4.8.3


C_PREDEF=-D __EXAMPLE_TCPCLIENT__

#-------------------------------------------------------------------------------
```

2.6、如何关闭 OCPU，使模块在标准模式运行

2.6.1、重新烧录软件版本(FW)

如果没有软件版本包，可以联系我司 FAE。

2.6.2、烧录 APP bin

在\custom\config\custom_sys_cfg.c 中更改为 APP_DISABLE,重新编译，烧录。

```
00044: /*************************************************************************/
00045: /* Configure the enable of application.                                  */
00046: /* The possible values are 'APP_DISABLE' and 'APP_ENABLE', and the       */
00047: /* default value is 'APP_ENABLE'.                                        */
00048: /*    APP_ENABLE: the application will work after downloading into module*/
00049: /*    APP_DISABLE:the application will not work after downloading.In this*/
00050: /*                case, the module still work as a STANDARD module.      */
00051: /*************************************************************************/
00052: static const ST_AppEnable appEnableCfg = {
00053:     APP_ENABLE
00054: };
```

2.6.3. A06 以前版本关闭 OCPU

切回AT口的处理步骤如下：

1） 使用SDK中默认的main.c编译APP，并烧录到模组。

2） 烧录APP后重启，通过串口发送命令(“AT+EPORT=1,connl,0\r\n”)配置UART0为AT口。

3） 通过串口发送命令(“AT+EPORT=3,0,9\r\n”)配置UART0波特率115200。

4） 烧录A05固件版本，注意在烧录版本之前，不能按reset，不然APP运行，会把步骤2中配置的AT口切回到UART3。

5） 烧录成功后即可以通过UART0(CHA)发送AT命令。

2.7、设置 debug 口模式，抓取 genie log

在\custom\config\custom_sys_cfg.c 中

```
/*************************************************************************/
/* Define the working mode of serial debug port (UART2).                 */
/*                                                                       */
/* The serial debug port (UART2) may work as a common serial port, as    */
/* well as work as a special debug port that can debug some issues       */
/* underlay application.                                                 */
/* Under the special debug mode, please refer to the document            */
/* "genie_Operation_UGD" for the usage of debug port                     */
/* now only supports BASIC MODE*/
/*************************************************************************/
static const ST_DebugPortCfg debugPortCfg = {
    //BASIC_MODE     // Set the serial  port  to a common serial port
    ADVANCE_MODE    // Set the serial  port to a special debug port
};
static const ST_DebugPortSet debugPortSet = {
    PORTNAME_UART2,       // Set the serial debug port (UART2) to GKI log .Can choose UART0/UART1/UART2
    PORTNAME_UART1       // Set the serial debug port (UART1) to HSL log .Can choose UART0/UART1/UART2
};
```

注：

1、 当 debug 口设置成 ADVANCE_MODE 时，APP 中不能通过 Ql_UART_Open 打开 debug 口输出文本 log。

2、如果当用户没有多余的串口同时抓 GKI log 或者 HSL log，可以只抓其中一项。

2.8、是否需要配置看门狗

NB 项目目前不需要配置看门狗，后续会删除程序中相关代码。

- NB_OPENCPU 项目并没有要求客户外加看门狗，因芯片内部有硬件看门狗，目前还未发现有程序 dump 不能重启现象。当程序出现卡住情况，模组会在 30s 后重启。
- 异常重启：比如指针错误，数据越界都会直接 dump，如果没有接 GKI 工具，会直

2.9、提前配置 GPIO

在\custom\config\ custom_gpio_cfg.h 中

```
#if 0 // If needed, config GPIOs here
GPIO_ITEM(PINNAME_SPI_MISO,       PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_SPI_MOSI,       PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_SPI_SCLK,       PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_SPI_CS,         PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_NETLIGHT,       PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_RI,             PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_DCD,            PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_CTS_AUX,        PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_RTS_AUX,        PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_GPIO1,          PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_RXD_AUX,        PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_TXD_AUX,        PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_GPIO2,          PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_GPIO3,          PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_GPIO4,          PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_GPIO5,          PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_RXD_DBG,        PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)
GPIO_ITEM(PINNAME_TXD_DBG,        PINDIRECTION_OUT,    PINLEVEL_LOW,  PINPULLSEL_PULLUP)

#endif
```

作用：

从模块开机到 APP 跑起来，大约需要 800ms，时间会比较久，如果客户想在内核期间就配置 gpio 的电平状态，可以通过这个配置表来实现。

2.10、增加一个 task

在\custom\config\ custom_task_cfg.h 中

```
00050: /*-------------------------------------------------------------------------------------------------------
00051:   |        Task Entry Function | Task Id Name    | Task Stack Size (Bytes) | Default Value1 | Default Value2 |
00052:   *-------------------------------------------------------------------------------------------------------*/
00053: TASK_ITEM(proc_main_task,       main_task_id,    10*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
00054: TASK_ITEM(proc_reserved1,       reserved1_id,    5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
00055: TASK_ITEM(proc_reserved2,       reserved2_id,    5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
00056: TASK_ITEM(proc_subtask1,        subtask1_id,     5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
00057:
```

注意：

1） 默认的三个任务（proc_main_task,proc_reserved1,proc_reserved2），建议客户不更改，顺序也不移动，因为我们用户库(app_start.lib)中会用到这三个任务，如果更改会影响代码运行。

2） 如果客户需要新增 task，可以在后面按照格式增加。

3） 如果客户新增的 task 中会用到 file system，则 stack size 必须设置最少 5K，防止栈溢出。

3、APP 下载

3.1、下载工具

我司 NB 模块的下载工具统一使用 QFLASH。目前推荐版本 QFLASH4.10 。

3.2、QFlash 下载流程

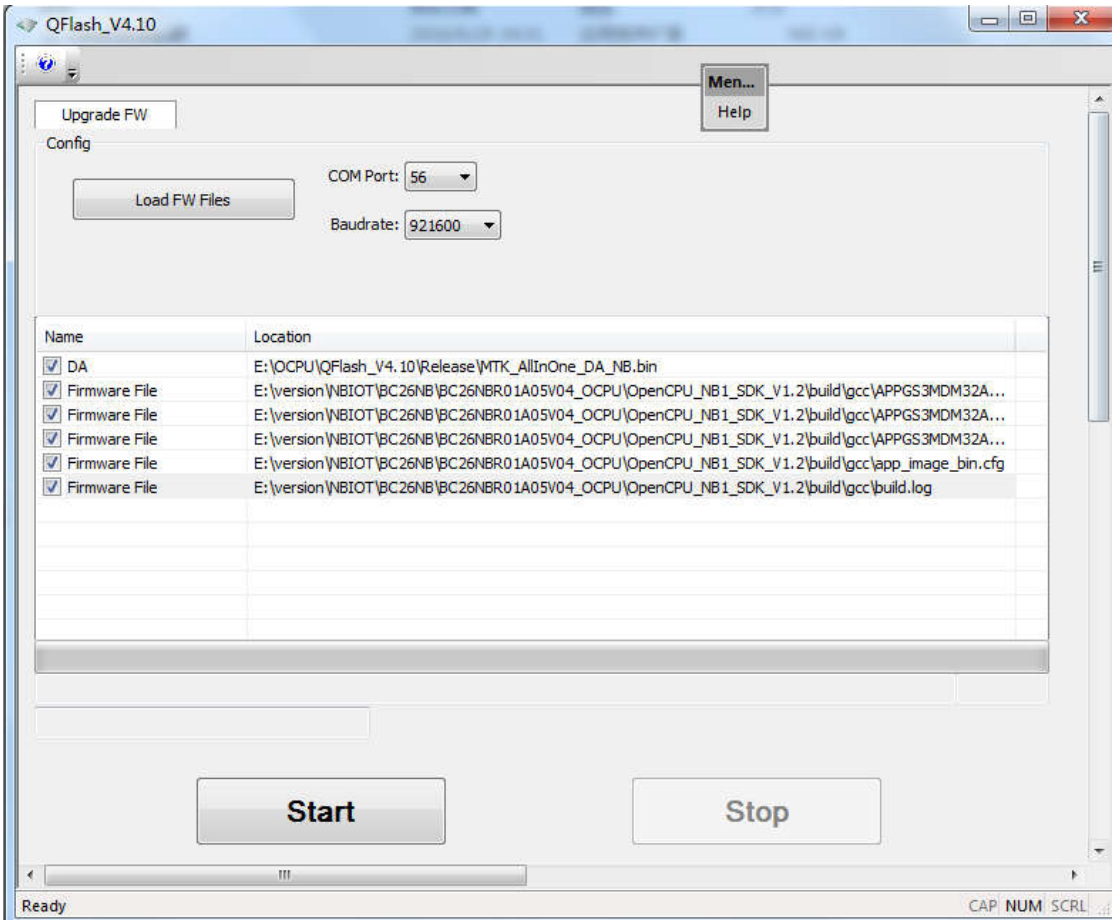
以我司 TE-B 开发板为例：

1） 打开 QFLASH,加载下载文件，点击 START

2） powerkey 拉低超过 500ms

注：如果模组已经在跑 APP bin（非第一次烧录 APP），按 reset 下载。

3.3、QFlash 工具参数配置

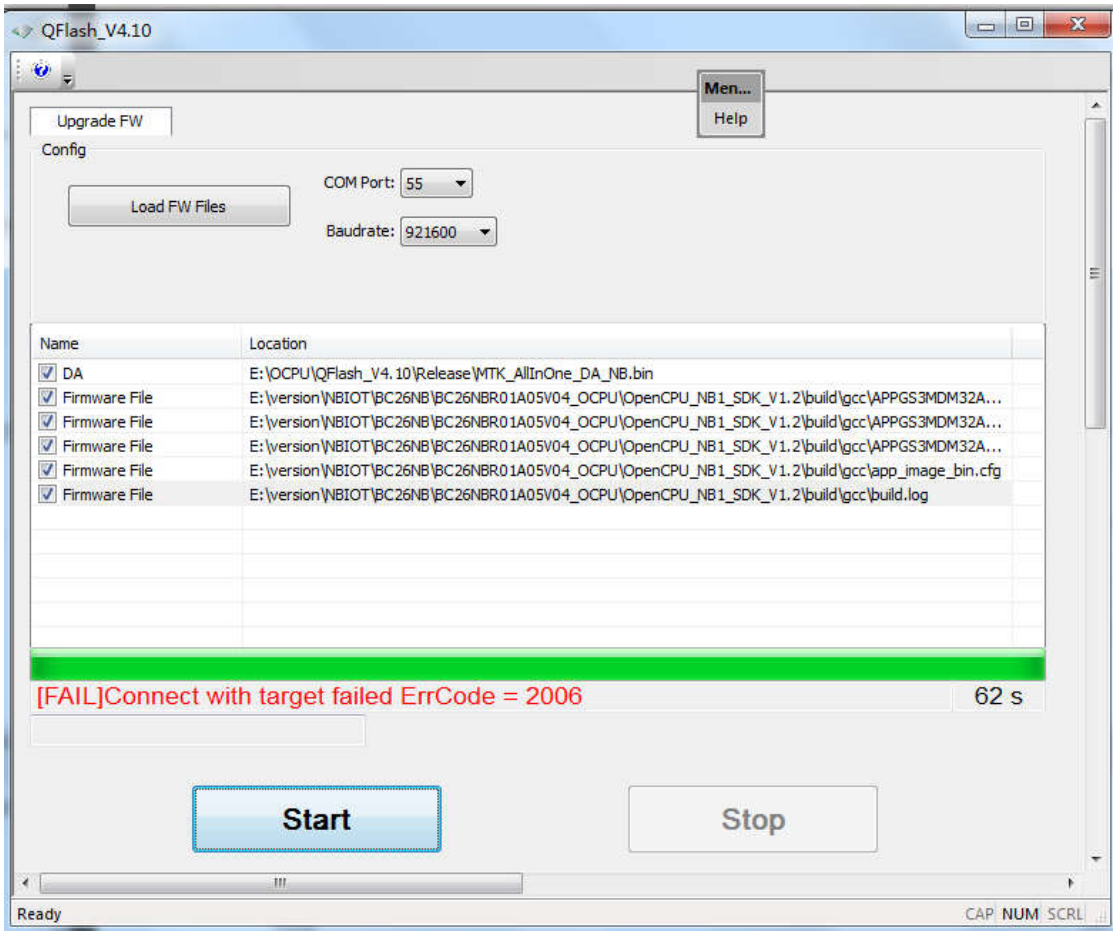


波特率

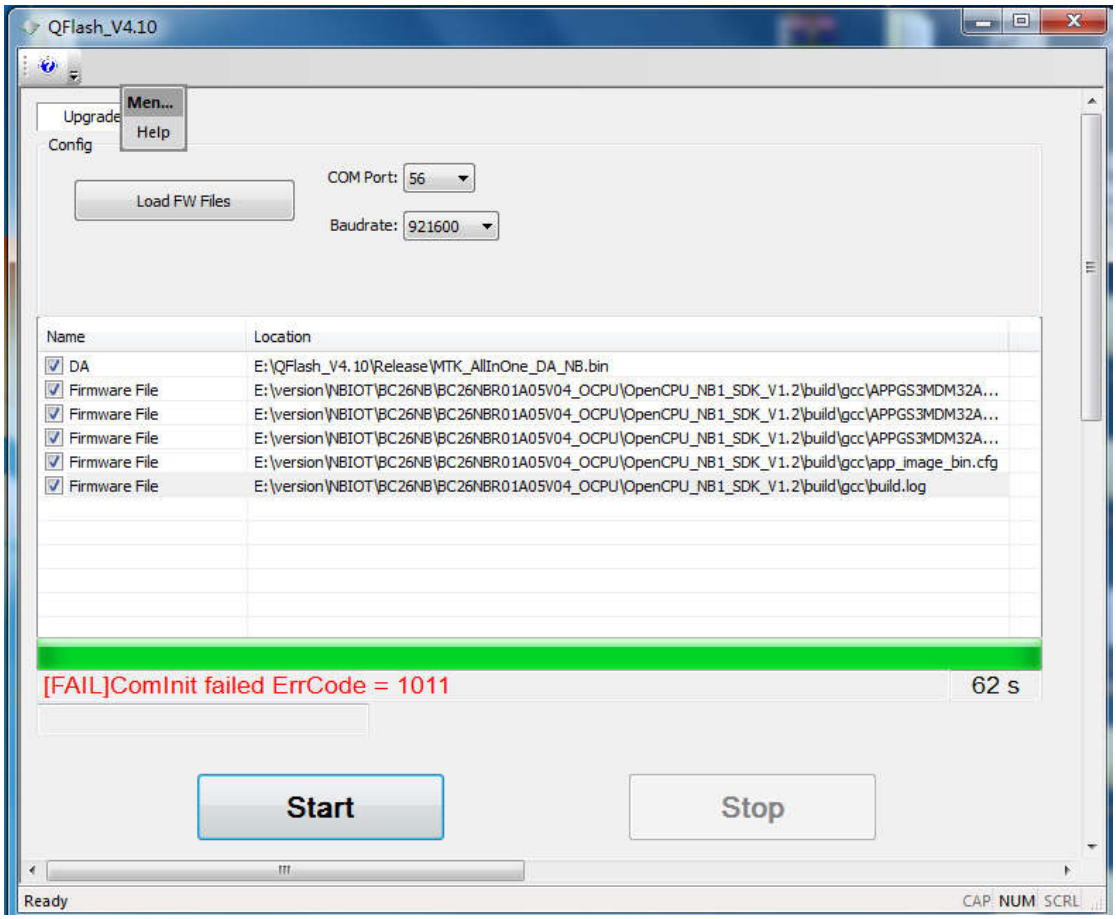
模块支持 9600-921600，一般推荐 921600，另客户选择波特率时，也需要考虑自身设计硬件电路支持波特率上限。

3.4、常见 ERROR

3.4.1、串口选择错误



3.4.2、串口被占用



常用工具介绍

1、抓包工具(genie)

1.1、工具作用

genie 是一个在 PC 端的工具，当模块出现网络异常或者 DUMP 时，通过应用层的 APP LOG 无法定位问题时，就需要客户抓取相应的 genie log 帮助分析。通常以下几类问题需要抓取 genie log.

1) 模块异常重启/死机
2) 网络相关异常，找网、注网、TCP 通讯等

3) 定位内核问题/调试内核代码

1.2、工具操作步骤

参考文档《Quectel_BC26_OPENCPU 模块 Genie log 操作指导 V1.0.pdf》

2、串口工具(QCOM)

2.1、工具作用

串口调试工具，作为上位机软件，客户可以通过 QCOM 来和模块进行数据交互。抓取 APP 层的应用 log，辅助分析问题。

2.2、 异常错误

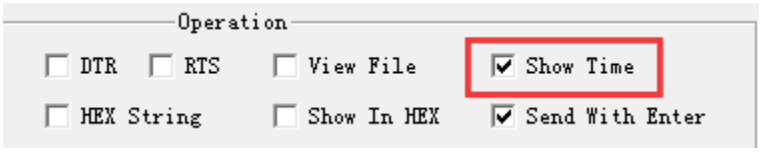
2.2.1、串口选择错误

2.2.2、串口被占用

2.2.2、没有勾选回车换行，导致 AT 不通。

2.3 、注意

1） 建议客户在使用 QCOM 工具时，勾选如下对话框，使 log 增加时间戳，便于分析问题。



2）如果需要跑压力测试，可以把 log 保存在本地 txt 文档中，避免 log 丢失。

SYSTEM 部分

1、 RIL

1.1、RIL 还未初始化成功就操作 AT 命令返回-5

注：我们的 Ql_SecureData_Read/Ql_SecureData_Store 接口都是通过操作 AT 命令来实现的，所以需要等到 RIL 初始化后才能调这些接口。RIL 初始化不仅仅是上报 MSG_ID_RIL_READY 的 URC，还需要调用 Ql_RIL_Initialize();才算初始化完成。

1.2、对于我们没有增加的 AT 命令，客户如何处理

客户可以参考我们已经实现的 RIL 接口和 AT 命令手册，自己封装一个接口

2、 URC

2.1、某些 URC 在 SDK 中并没有实现，如何处理

客户可以参考我们已经实现的 URC 处理接口和 AT 命令手册，自己封装

2.2、URC 消息处理

默认 URC 消息往主 task 中发送，所以主 task 一定需要增加接收消息的接口（Ql_OS_GetMessage）,用户也可在 RIL_URC.c 中更改 URC 消息发送的 task id,往其他 task 发送 URC 消息，建议不要更改。

2.3、收到 type（msg.param1）为 101 的 URC 如何处理

这类 URC 被称为未定义的 URC，即我们 opencpu 收到这类 URC 时，并没有找到相应 URC 处理的接口，对此进行解析处理。
用户如果需要解析这类 URC，可以自己来增加对应的接口（在 RIL_URC.c 中），如果不需要关心这类 URC，也可以直接忽略。

2.4、URC 上报的条件

URC 的上报都是有一定条件的，比如网络状态的“\r\n+CEREG:”，需要网络状态改变时触发。其他 URC 也是类似。

3、MESSAGE

3.1、s32 Ql_OS_GetMessage(ST_MSG* msg)

这是一个很重要的接口，在 opencpu 中，可以通过这个接口来处理当前 task 收到的消息。
它有如下的特点：
1、 可以处理当前 task 或者其他 task 发来的消息
2、 TIMER、EINT、ADC 事件的处理，也要依赖于在这个接口中根据不同的消息来调用 callback
3、 当 task 阻塞在这个接口的时候，任务会被挂起
4、 task 只有来消息时，会继续执行一次，其余时间都在这个接口处阻塞。

3.2、发送消息时，一定要注册 taskid

想知道某个 task 的 id,有如下方式：
1. 在\custom\config\ custom_task_cfg.h 中，查找对应的 task 配置

```
/*--------------------------------------------------------------------------------------------------
 |      Task Entry Function | Task Id Name   | Task Stack Size (Bytes) | Default Value1 | Default Value2 |
 *--------------------------------------------------------------------------------------------------*/
TASK_ITEM(proc_main_task,       main_task_id,   10*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
TASK_ITEM(proc_reserved1,       reserved1_id,   5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
TASK_ITEM(proc_reserved2,       reserved2_id,   5*1024, DEFAULT_VALUE1, DEFAULT_VALUE2)
```

2. 通过接口（Ql_OS_GetActiveTaskId）来获取当前的 task id

注：如果往一个不存在或者错误的 task id 发送消息，可能会导致系统异常。

3.3、每个 task 的消息队列，最大只能保存 30 条消息

用户实际使用时，如果有频繁的消息需要处理，尽量不要在 task 中做阻塞或者 Ql_Sleep 动作。

4、EVENT、MUTEX

4.1、MUTEX 不支持重复锁的机制

4.2、MUTEX 创建后不会释放。

6、TASK

6.1、TASK 中不能及时处理业务

如果调用阻塞接口或者 Ql_Sleep 接口，task 会阻塞，导致无法继续处理其他业务或者消息

6.2、task 调度

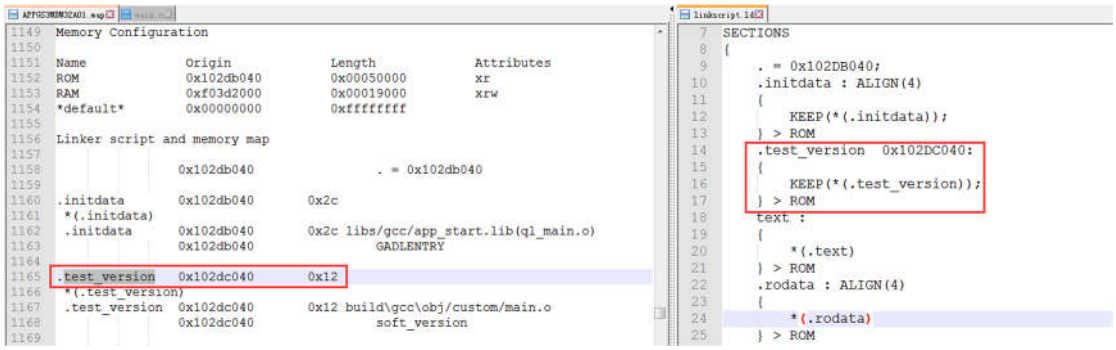
如果任务中不增加接口（Ql_OS_GetMessage）来挂起任务，而是采用如下的方式来做 while（1）循环，会由于当前 task 占用大量的系统资源，导致模块死机。如果客户需要使用这样的轮询方式，可以在 while（1）中增加延迟接口。

```
void proc_subtask1 (s32 taskId)
{
    while(TRUE)
    {
     // Ql_Sleep(5);
    }
}
```

8、其他

8.1、客户想在 ROM 中的某个地址写入数据

更改我们的 linkscript.ld 文件如下：



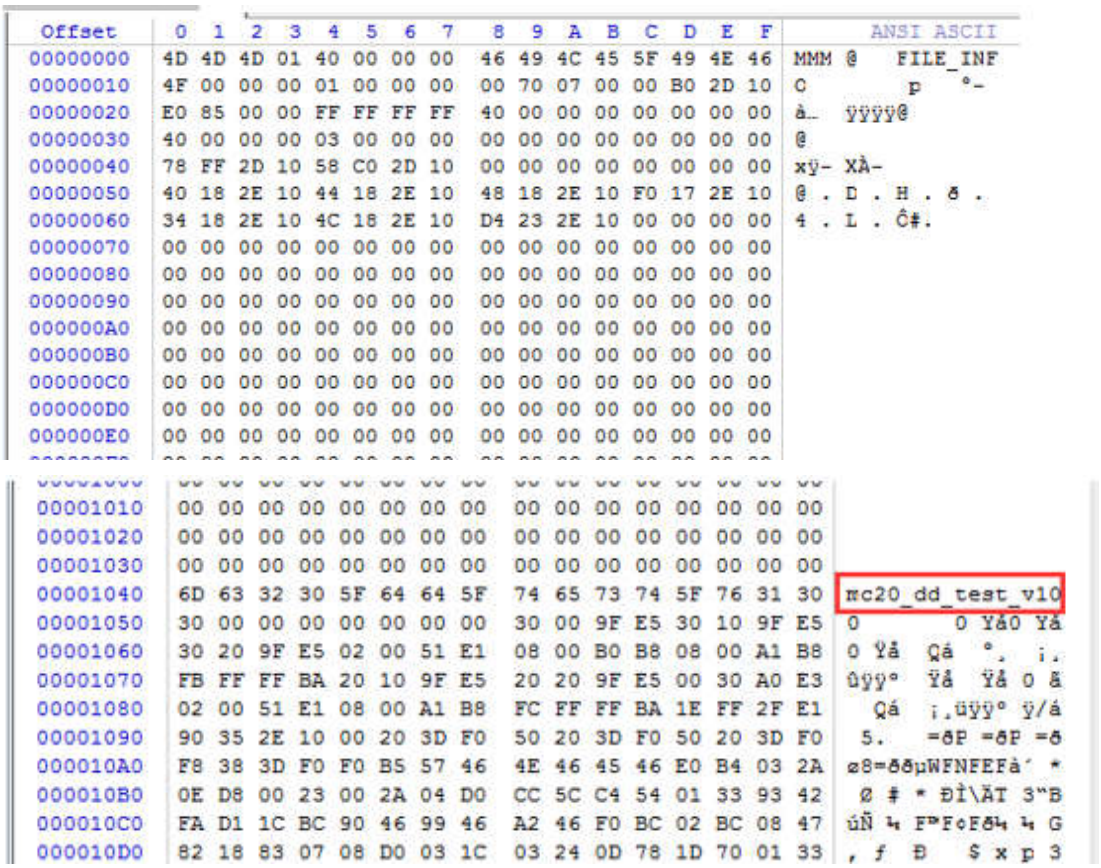
```
Memory Configuration

Name             Origin             Length             Attributes
ROM              0x102db040         0x00050000         xr
RAM              0xf03d2000         0x00019000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map

                 0x102db040                 . = 0x102db040

.initdata        0x102db040         0x2c
 *(.initdata)
 .initdata       0x102db040         0x2c libs/gcc/app_start.lib(ql_main.o)
                 0x102db040                 GADLENTRY

.test_version    0x102dc040         0x12
 *(.test_version)
 .test_version   0x102dc040         0x12 build\gcc\obj/custom/main.o
                 0x102dc040                 soft_version
```

```
SECTIONS
{
    . = 0x102DB040;
    .initdata : ALIGN(4)
    {
        KEEP(*(.initdata));
    } > ROM
    .test_version  0x102DC040:
    {
        KEEP(*(.test_version));
    } > ROM
    text :
    {
        *(.text)
    } > ROM
    .rodata : ALIGN(4)
    {
        *(.rodata)
    } > ROM
```

在代码中定义段，如下：

```
__attribute__((section(".test_version")))
const char soft_version[]    = "mc20_dd_test_v100";
```

实际 bin 结果：



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	4D	4D	4D	01	40	00	00	00	46	49	4C	45	5F	49	4E	46	MMM @ FILE_INF
00000010	4F	00	00	00	01	00	00	00	00	70	07	00	00	B0	2D	10	O p °-
00000020	E0	85	00	00	FF	FF	FF	FF	40	00	00	00	00	00	00	00	à… ÿÿÿÿ@
00000030	40	00	00	00	03	00	00	00	00	00	00	00	00	00	00	00	@
00000040	78	FF	2D	10	58	C0	2D	10	00	00	00	00	00	00	00	00	xÿ- XÀ-
00000050	40	18	2E	10	44	18	2E	10	48	18	2E	10	F0	17	2E	10	@ . D . H . ð .
00000060	34	18	2E	10	4C	18	2E	10	D4	23	2E	10	00	00	00	00	4 . L . Ô#.
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001040	6D	63	32	30	5F	64	64	5F	74	65	73	74	5F	76	31	30	mc20_dd_test_v10
00001050	30	00	00	00	00	00	00	00	30	00	9F	E5	30	10	9F	E5	0 0 Ÿå0 Ÿå
00001060	30	20	9F	E5	02	00	51	E1	08	00	B0	B8	08	00	A1	B8	0 Ÿå Qá °¸ ¡¸
00001070	FB	FF	FF	BA	20	10	9F	E5	20	20	9F	E5	00	30	A0	E3	ûÿÿº Ÿå Ÿå 0 ã
00001080	02	00	51	E1	08	00	A1	B8	FC	FF	FF	BA	1E	FF	2F	E1	Qá ¡¸üÿÿº ÿ/á
00001090	90	35	2E	10	00	20	3D	F0	50	20	3D	F0	50	20	3D	F0	5. =ðP =ðP =ð
000010A0	F8	38	3D	F0	F0	B5	57	46	4E	46	45	46	E0	B4	03	2A	ø8=ððµWFNFEFà´ *
000010B0	0E	D8	00	23	00	2A	04	D0	CC	5C	C4	54	01	33	93	42	Ø # * ÐÌ\ÄT 3"B
000010C0	FA	D1	1C	BC	90	46	99	46	A2	46	F0	BC	02	BC	08	47	úÑ ¼ FFFcFð¼ ¼ G
000010D0	82	18	83	07	08	D0	03	1C	03	24	0D	78	1D	70	01	33	, ƒ Ð $ x p 3

注：客户最好还是不要固定段地址，如果前一个段和当前段空间预留不足，编译会提示超出范围，如果预留过大，会造成 ROM 的浪费。建议还是使用我们默认的动态管理方式。

8.2、我们模块是大端还是小端模式

我们 NB 平台的所有模块都是小端模式。

8.3、调用 Ql_Reset()重启模块需要拉低 PWRKEY 吗？

不需要，调用 Ql_Reset()接口后，模块的 pwrkey 引脚不管拉高还是拉低，都不影响重启。

8.4、如何添加用户自己的 lib 库

1、用 gcc 自带的 ar 工具把当前的所有.o 文件打包成库

```
arm-none-eabi-ar -r test.lib    *.o
```

2、makefile 中增加编译的库

```
USERLIB += libs/gcc/app_start.lib \
                libs/gcc/test.lib
```

8.6、编译时，发现报错 _sbrk 未定义

客户调用了 C 标准库的 strtod 接口，这些接口我们默认增加的 lib 库中缺少部分依赖接口的实现。客户可以参考源码自己来实现这类接口。
注：我们已经实现了部分的标准库接口，并使用 QL 开头，用户如果用到这类标准库，需要使用 QL 开头，而不能直接调用标准库。

Deep sleep 模式部分

1、Deep sleep 模式介绍

在嵌入式应用中，系统的功耗越来越受到人们的重视，这一点对于需要电池供电的便携式系统尤为明显，降低系统功耗，延长电池的寿命，就是降低系统的运营成本。系统功耗的最小化需要从软、硬件两方面入手，下面我们重点介绍软件实现。

2、模组进入 deep sleep 模式的条件

模组必须符合下面两个条件，才会进入 deep sleep 模式：
A: modem 侧进入 PSM
B：AP 侧进入 PSM

2.1、Modem 侧如何进入 PSM

Modem 侧进入 PSM 的唯一条件是 T3324 timeout。

2.2、AP 侧如何进入 PSM

当模组当前正在运行的所有 task 都处于 idle 状态时，模组 AP 侧就会自动进入 PSM 模式，如果想模组快速进入 idle，可以通过 Ql_OS_GetMessage 挂起 task。

3、进入 deep sleep 模式接口

为了降低功耗，模组默认是使能进入 deep sleep 模式的。如果模组中间后执行过 disable 操作，再次希望进入 deep sleep 模式，可以通过下面两种方式其一进入睡眠模式，需要强调的是系统并不会立即进入睡眠，而是依赖于当前网络及系统所有任务执行状态，只有当 AP 和 modem 侧都进入 PSM，模组才会进入 deep sleep。

2.1、AT+QSCLK(标准下推荐方案)

4.14. AT+QSCLK Configure Sleep Mode

The commands is used to configure UE sleep mode.

Please refer to ***Chapter 6*** for possible <err> values.

AT+QSCLK Configure Sleep Mode	
Test Command **AT+QSCLK=?**	Response **+QSCLK: (0-2)** **OK**
Read Command **AT+QSCLK?**	Response **+QSCLK: <n>** **OK**
Write Command **AT+QSCLK=<n>**	Response **OK** If there is any error, response: **ERROR** or **+CME ERROR: <err>**
Maximum Response Time	300ms

Parameter

<n>	0	Disable Sleep Mode
	1	Enable light sleep and deep sleep, wakeup by PSM_EINT (Falling Edge)
	2	Enable light sleep only, wakeup by Main UART

Example

```
AT+QSCLK=1
OK
```

特别提醒：

模式2，第一次串口数据会被丢弃，仅仅起唤醒作用。建议此模式下先发一个AT 来唤醒模块再继续其他业务。

2.2、Ql_SleepEnable(Opencpu 推荐方案)

```
/*****************************************************************
* Function:     Ql_SleepEnable
*
* Description:
*           Set the module into sleep mode at once
*
* Return:
*           QL_RET_OK indicates this function successes.
*           Ql_RET_NOT_SUPPORT this function not support.
*****************************************************************/
s32 Ql_SleepEnable(void);
```

在 opencpu 模式下，除了设置 AT 命令（AT+QSCLK），还可以通过接口（Ql_SleepEnable）使模块进入睡眠模式。

4、如何禁止模组 deep sleep 模式

●在标准模式下失能 deep sleep，可以通过 AT 命令（AT+QSCLK=0）。

●使用 Opencpu 方案，既可以通过 AT 命令（AT+QSCLK=0），也可以通过如下接口（Ql_SleepDisable）失能 deep sleep 模式。

```
/*****************************************************************
* Function:     Ql_SleepDisable
*
* Description:
*           Exit  the sleep mode
*
* Return:
*           QL_RET_OK indicates this function successes.
*           Ql_RET_NOT_SUPPORT this function not support.
*****************************************************************/
 s32 Ql_SleepDisable(void);
```

5、如何从 deep sleep 模式唤醒模块：

5.1、标准方案

●TAU
同时唤醒 AP 侧和 modem 侧
● PSM_EINT 引脚下降沿
仅仅唤醒 AP 侧

5.2、OPENCPU 方案

- TAU
  同时唤醒 AP 侧和 modem 侧
- PSM_EINT 引脚下降沿
  仅仅唤醒 AP 侧
- RTC 超时
  仅仅唤醒 AP 侧

特别提醒：

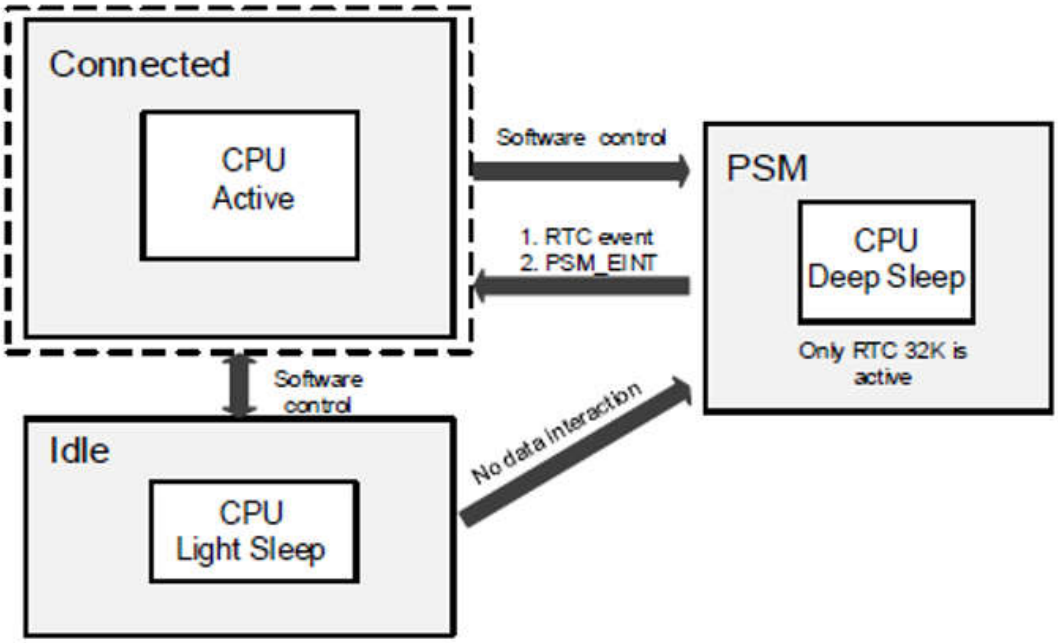
除了TAU外，其他唤醒方式仅仅是唤醒了模块的AP侧，这个时候，模组串口可以正常收发数据，APP正常运行，如果用户想要接收服务器下发数据，还需发送一包上行数据到服务器唤醒modem侧的PSM。

6、模组模式切换

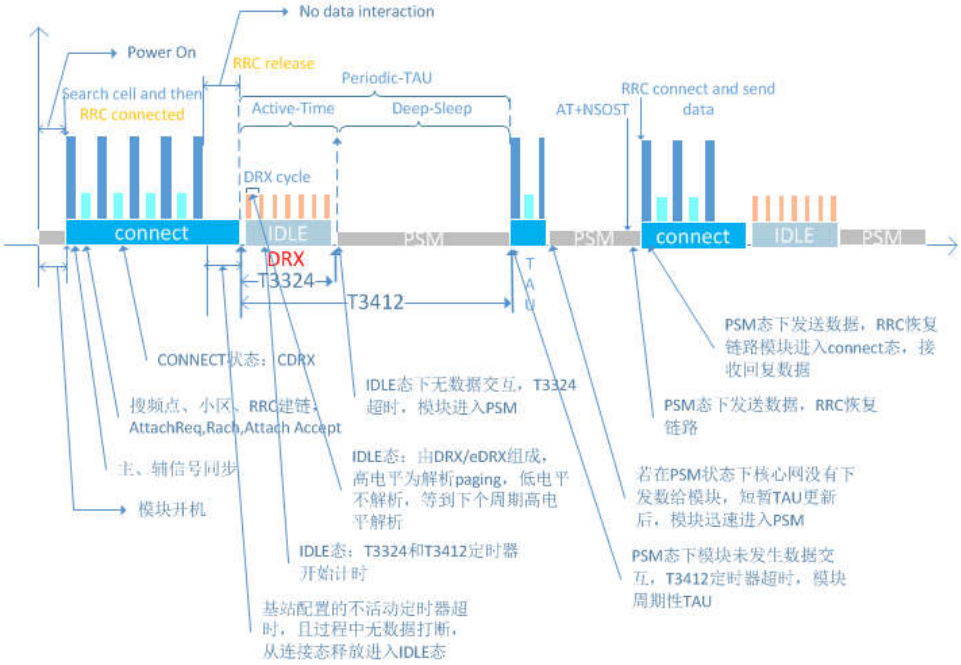
6.1、模组三种工作模式(Active、Idle、PSM)

模式	工作状态描述	
正常工作模式	Connected	连接状态：模块处于 Active（工作）模式，所有功能正常可用，可以进行数据发送和接收；模块在此模式下可切换到 Idle 模式或 PSM 模式。
	Idle	空闲状态：模块处于 Light Sleep（轻休眠）模式，网络处于 DRX/eDRX 状态，寻呼窗口内可接收寻呼。模块在此模式下可切换至 Connected 或 PSM 模式。
	PSM	省电状态：模块处于 Deep Sleep（深睡眠）模式，CPU 掉电，内部只有 RTC 工作；网络处于非连接状态，无法接收下行数据；模块在此模式下可切换至 Connected 模式。

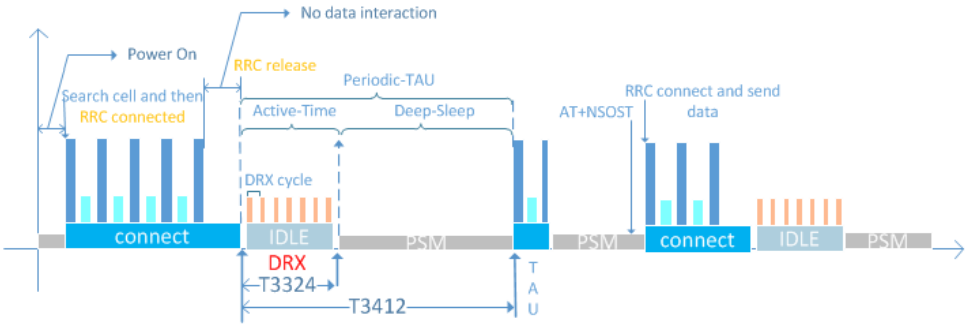
6.2、模式间切换



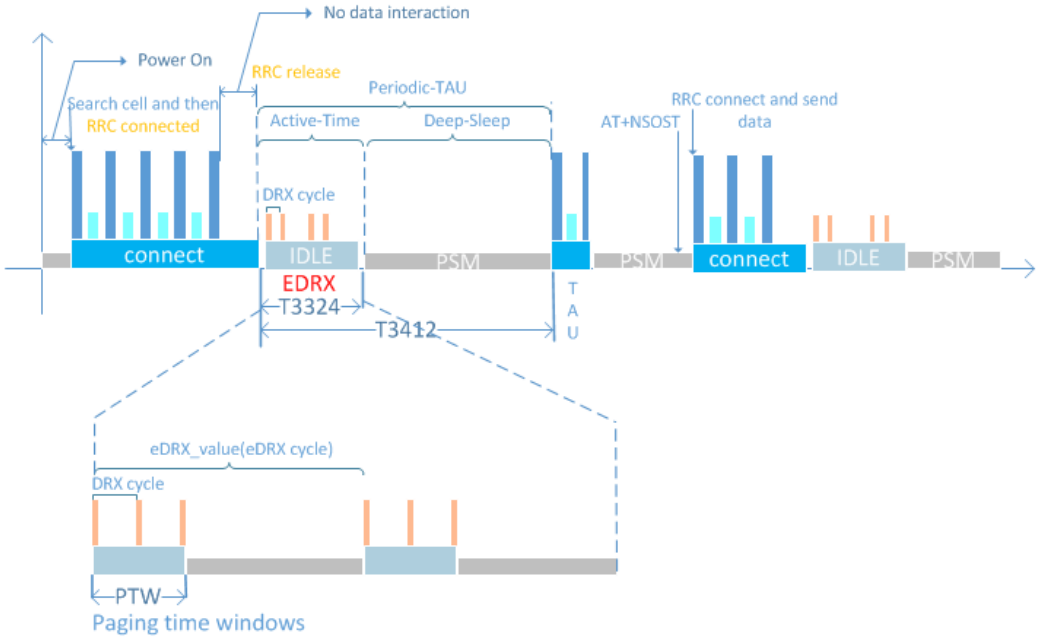
5.3、modem 侧各个状态流程



Modem侧完整的状态流程转换过程



模组支持DRX的情况



模组支持EDRX的情况

6、耗流

耗流值如下表所示，测试电源电压是 3.3V.

表 1：模块耗流

参数	模式	描述	最小值	典型值	最大值	单位
I_{VBAT}	PSM	睡眠状态		3.5		μA
	Idle	eDRX=81.92s, PTW=40.96s		288		μA

参数	模式	描述		最小值	典型值	最大值	单位
		@DRX=1.28s			541		μA
		@DRX=2.56s			434		μA
	Active [1)]	Single- tone (15kHz 子载波间隔)	B1 @23dBm		100	285	mA
			B3 @23dBm		107	308	mA
			B5 @23dBm		107	303	mA
			B8 @23dBm		113	325	mA
			B20 @23dBm		109	301	mA
		Single- tone (3.75kHz 子载波间隔)	B1 @23dBm		193	302	mA
			B3 @23dBm		215	335	mA
			B5 @23dBm		215	330	mA
			B8 @23dBm		224	344	mA
			B20 @23dBm		215	329	mA

备注

[1)] 仪器测试条件下的耗流值。

特别提醒：

实网下由于干扰和当地运营商DRX/EDRX等策略的不同，功耗会和实验室环境下有一定的差异，另上述测试数据客户可以参考我司硬件手册。

7、注意事项

7.1、deep sleep 模式，模块串口不能接收数据

7.2、deep sleep 模式，必须关闭 GP-TIMER 和 GPS

频繁的中断或者 GPRS 数据，会导致模块 AP 侧很难进入 PSM 模式或者进入后会马上唤醒，从而影响模组进入的 deep sleep。

7.3、目前用户设计中不建议使用 pwrkey 唤醒模组

7.4、 Opencpu 方案，task 中增加接口（Ql_OS_GetMessage）

当模块有消息需要处理时，会通过这个接口来处理，没有消息时 task 挂起，可以立刻让 AP 侧进入 deep sleep 模式。

7.5、准备进入 deep sleep 模式前关闭频繁中断（如 TIMER、EINT）

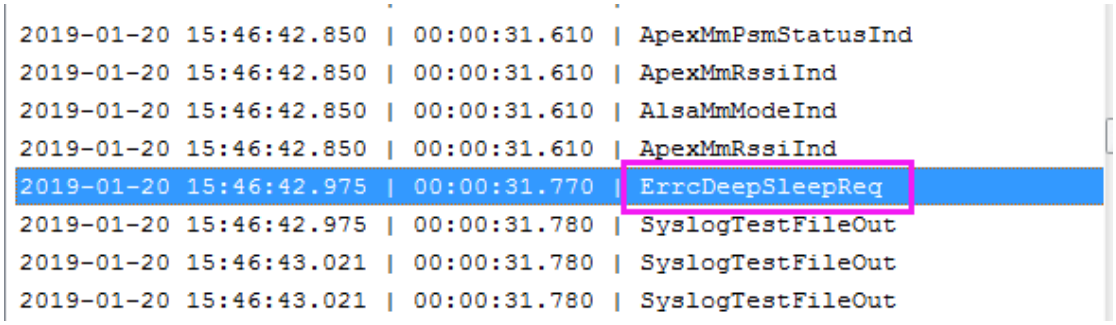
如果有频繁中断，模块将很难进入睡眠模式，或者会频繁唤醒，导致功耗降不下去。例，如果应用中增加了周期 1s 的 timer，模块功耗降不会明显下降，可以延长周期或者关闭 timer。

7.6、客户反馈在切 CFUN0 后，60s 内并没有进入 PSM 模式

如果网络正常注册后，切 CFUN0 马上返回 OK，但如果之前网络一直没有注册成功，这时候切 CFUN0，按照协议需要等到 RRC 连接超时（60-80s）才会返回 OK。然后模组进入 deep sleep 模式。

7.7、modem 侧进入 PSM 的判断方法

1、设置 AT+QNBIOTEVENT=1,1 ，如果 modem 侧进入 PSM，会上报 URC:
(\r\n+QNBIOTEVENT: “ENTER PSM")。
2、log 中收到如下信息



7.8、TCP 链路需要 closed 模组才会进入 deep sleep

7.9、模块进入 deep sleep 模式有什么具体特征

特别提醒：

我们测试的睡眠模式功率，是裸模块下的数据，如果客户在自己 PCB 测试，需要减去外设和供电电路的功耗。

8、opencpu 常见唤醒机制

在 opencpu 下，客户可以通过如下方式来唤醒模块：

1、rtc （客户在模块进入 deep sleep 前启动一个 rtc，比如定时 10min，模块会每隔 10min 唤醒一次，模块唤醒后，可以去进行相关业务，业务处理完，继续睡眠，等待下次唤醒）

2、PSM_EINT 引脚唤醒（客户通过触发 PSM_EINT 来唤醒模块，这个方式比较适合模块外部有 MCU 或者传感器,可以通过给 PSM_EINT 引脚一个下降沿唤醒模块

3、TAU 客户可以通过向运营商配置不同的 TAU 时间来唤醒模组。（这类应用一般可以在对于实时性不要非常低的场景，比如抄表）

FOTA 升级部分

1、简介

为了满足客户 APP BIN/CORE 的 OTA(over the air)升级功能，我司提供了 DFOTA 方案，客户需要制作一份升级包，把升级包放到 HTTP 服务商，通过我司的 AT 命令即可实现

远程升级方案。我司暂不提供服务器租赁服务，所以用户需要自己搭建 http 服务。

2、升级流程

以升级 APP 服务器为例：
1) 客户搭建 HTTP 服务器
2) 登录我司云平台制作差分(功能还在添加)
3）模块注册网络成功(**AT+CEREG?**)
4）发送 AT 命令(**AT+QFOTADL="http://www.quectel.com:100/update.zip"**)
5）模组自动完成下载、升级、重启

注：1、如果是升级固件，差分包请联系我司 FAE 获取。
2、升级包需要先小批测试基本功能和压力，确保正常后，才能大批升级
3、如使用我司两款以上产品，项目升级包要有固定的命名规则，避免混淆

SIM 卡部分

1、常见客户问题

1.1、SIM 卡欠费

如果怀疑 SIM 卡欠费，可以拨打运营商电话:

```
[2017-12-05_13:27:44:125] AT+QAUDCH=1
[2017-12-05_13:29:11:005]OK
[2017-12-05_13:29:13:095]AT+QMIC=0,8
[2017-12-05_13:29:13:095]OK
[2017-12-05_13:29:16:605]AT+CLVL=60
[2017-12-05_13:29:16:605]OK
[2017-12-05_13:29:17:572]AT+CLIP=1
[2017-12-05_13:29:17:572]OK
[2017-12-05_13:29:20:100]ATD10010;
[2017-12-05_13:29:20:100]OK
```

1.2、开机无法找到 SIM 卡

现象描述：开机后模块返回 +CPIN: NOT READY or NOT INSERTED。从以下几方面分析解决：
1) SIM 卡与卡座接触不良，可以尝试在 SIM 卡上增加垫片。
2) SMT 焊接不良，可以通过万用表测试模块与 SIM 卡焊盘之间的连通性。

3) SIM 卡已损坏，可以将 SIM 卡放在 EVB 或者手机上测试，确认是否正常。
4) 假如 VBAT 电源走线过于靠近 SIM 信号线，可能会由于 VBAT 电源线上的纹波太大，干扰到 SIM 卡各个信号线，导致 SIM 卡无法识别，可尝试切断附近的 VBAT 线，通过其他途径单独给模块供电，看问题是否消失。
5) 假如客户的 SIM 卡座和模块确实离得很远，走线也比较长，各个信号线也没有地屏蔽处理，很有可能导致 SIM 卡无法识别，可以尝试使用较短的飞线直接连接到卡座。
6) SIM 卡信号线上并联的 ESD 器件寄生电容需要不大于 50pF，过大可能会导致 SIM 卡无法识别，可尝试直接去掉该 ESD 看问题是否消失。
7) 天线摆放位置以及射频走线不合理也会干扰到 SIM 卡，从而导致 SIM 卡无法识别，客户可以使用 AT+CFUN=4,1 关闭模块射频发射和接收，确定 SIM 卡工作是否正常，若正常则表示 SIM 卡受到 RF 干扰。为了尽可能的消除 RF 干扰，SIM_DATA,SIM_CLOCK,SIM_RST,SIM_VDD 并联滤波电容。具体参考模块硬件设计手册。
8) 周边环境有干扰，确认测试现场周围是否有超强电/磁场存在，比如高压输电线、大功率无线设备等， 可以尝试使用一个屏蔽罩盖住 SIM 卡以及 SIM 卡各个信号线的走线，或者使用地屏蔽线处理 SIM 卡信号线，看问题是否消失。

1.3、使用过程中出现掉卡的问题

现象描述：模块开机返回+CPIN: READY, 过一会儿后模块返回 +CPIN: NOT READY。
可能原因：
1) RF 干扰，可以通过以下方法确认：
   a) 可以尝试使用 AT+CFUN=4,1 关闭模块射频发射和接收，看看问题是否仍然存在。
   b) 可以尝试把天线靠近 SIM 卡和 SIM 信号线，看看问题出现概率是否有所增加。
   c) 确认测试现场周围是否有超强电/磁场存在，比如高压输电线、大功率无线设备等。
   d) 可尝试通过并联 15~33PF 电容来滤除射频干扰。
2) 硬件设计存在问题。可以尝试如下办法确认：
   a) 假如 VBAT 电源走线过于靠近 SIM 信号线，当刚开机的时候纹波可能不是很大，模块能正常找到 SIM 卡，但是当模块开始注册并同步网络时，纹波增大可能直接干扰到 SIM 卡的信号线，导致掉卡。可尝试切断附件的 VBAT 线，通过其他途经单独给模块供电。
   b) 确认该卡工作是几伏，1.8V 还是 3V？有些情况下，1.8V 的卡更容易被干扰，可尝试使用 AT 命令 AT+QSIMVOL 锁定 3V 看问题是否依然存在。
3) SIM 卡质量比较差。可以尝试如下办法确认：
   a) 换张卡看看是否有同样的问题。
   b) 把问题卡放在 Quectel EVB 上确认是否有同样问题。

1.4、 哪些原因可能造成模块开机后无法注册网络？

1) 确认模块是否找到 SIM 卡（AT+CPIN?）；
2) 确认 SIM 卡是否欠费；
3) 确认模块工作频段（AT+QBAND）和工作模式（AT+CFUN）是否正确；
4) 确认模块射频信号是否正常（AT+CSQ）；

5) 查询模块 IMEI 号（AT+GSN），确认 IMEI 号是否合法。由于有些客户会改写 IMEI 号，导致部分区域网络认为此 IMEI 非法，从而禁止模块注册。

1.5、模块关机，给模块发短信，模块开机需要一定时间才能收到。这个时间能否缩短？

短信中心转发会判断模块的开关机状态，如果关机会有重发机制，所以这个时间由短信中心控制我们无法更改。当模块开机短信功能初始化完成，网络正常情况下就会在这个时间内收到短信中心重发的短信。

网络部分

1、TCP、UDP

我们模块一共支持最大同时建立 6 路 socket，所以用户最大可以同时建立 6 路的 TCP 或者 UDP。

1.3、参考用例

Example_tcpclient.c ------>TCP 长连接
Example_tcpdemo.c ------>TCP 短连接
Example_udpclient.c ------>UDP，模块作为客户端

1.4、常见客户问题

1.4.1、域名异常导致解析失败

问题现象：使用 Ql_IpHelper_GetIPByHostName/AT+QIDNSGIP 返回域名解析失败
问题分析：使用 nslookup 工具检查当前域名是否正常，如果异常反馈给客户的网络管理员

正常

异常

问题原因：客户的域名异常

1.4.2、模块使用电信 NB 卡建立 TCP 连接，发送 4-6 包，被对端 closed 问题

电信 NB 卡暂时不支持 TCP 业务，如果用户需要使用 TCP 业务，需把服务器 IP 加到 SIM 卡白名单中。

1.4.3、当模块建立 TCP 连接后，如果不 closed 当前的 socket，模组不会进入 deep sleep 模式

1.4.4、TCP/UDP 接收数据的方式有哪些？

1) buffer 模式：收到 server 端数据后，先保存到本地的 buffer 中，上抛一个 URC 提示客户来调用 read 动作
2) direct 模式：直吐模式，收到 server 端的数据，数据会跟着 URC 一起抛给用户。

1.4.5、 TCP 连接中模块的 IP 地址会如何变化？

模块和服务器建立 TCP 连接之后，模块的 IP 地址是不会变的，可以通过 AT+QILOCIP 获得。如果执行 AT+QICLOSE 后，再通过 AT+QIOPEN 建立连接 IP 地址也不会变。但是执行 AT+QIDEACT 后，再通过 AT+QIOPEN，IP 地址就会变化。原因是 AT+QICLOSE 只是把当前连接给关闭了，但是 PDP 场景还是激活的，而 AT+QIDEACT 则把当前场景给关闭了，再次连接时需要重新激活场景，激活场景的时候会重新获得 IP 地址，至于网络分配给模块的 IP 地址是根据运营商的配置而决定的。

1.4.6 模块 AT+QISEND 返回正确，但是为什么服务器还没有收到发送过去的数据？

AT+QISEND 返回正确的 length,表示模块上层发送的数据已经送入了底层 TCP socket 对应的 buffer 中，这并不代表数据已经发送到了服务器，如果想知道数据是否发送到了服务器可以通过 AT+QIACK 查询。

1.4.7 模块在哪些情况下会上报 CLOSE？

模块在下面三种情况下会上报 CLOSE：
1) 服务器主动关闭了 TCP 链接；
2) 网络异常发送 RST 包中断模块和服务器的 TCP 链接。
3)其他

1.4.8 客户反馈 TCP 发送数据到平台后，模块重启

客户通过串口使用我们的 mian.c 例程来建立 socket，模块默认 AT+QICFG="showlength"[,<show_length_mode>]的 mode 为 0，导致 URC 处理时，指针错误，模块重启。建议使用我们封装好的 RIL 接口(接口中默认设置为 1)或者客户在建立 socket 之前，自己设置 showlength 为 1。

2、QNTP

2.1、客户使用 AT+QNTP 返回-2

问题原因：网上公用的 NTP 服务器故障率很高，有些服务器连续连接会出现连接不上。
问题解决：推荐客户使用以下阿里云的 NTP 服务器。

```
ntp1.aliyun.com
ntp2.aliyun.com
ntp3.aliyun.com
ntp4.aliyun.com
ntp5.aliyun.com
ntp6.aliyun.com
ntp7.aliyun.com
```

注：客户可以购买收费的 NTP 服务器

2.2、网上免费 NTP 服务器不稳定，客户端应对方法推荐

1、尽量使用域名的 NTP 服务器，避免 IP 更改不能使用
2、可以设置多个 NTP 服务器，作为备用
3、可以自己搭建一个时间服务器，通过 TCP 方式获取当前的时间。

3、QNITZ

3.1、模块注网成功，并未发现有时间同步

部分地区运营商不支持该功能，具体可以咨询当地运营商

3.2、什么时候模块会上报时间同步的 URC

每次模块重新注册网络后，会上报时间同步的 URC
注：部分地区的运营商会连续上报两次 URC

3.3、NITZ 功能是否需要 GPRS 流量

不会产生数据流量

3.4、QNITZ 的 URC 上报

有时刚配置的需要先固定波特率 115200

4、MQTT

4.1、OPEN 下 MQTT 收发十六进制数据时特殊字符

发：对于 0D\0A\1A\00\08 等特殊字符，可以在发送的时候加上包的大小。
收：接收 0D\0A 外的其他字符没有问题，0D\0A 我们 ril 接口在处理的时候会把包分成两包，造成数据不对。目前推荐规避方式：注册和打开虚拟串口来接收匹配收到的 MQTTURC。

1、ONENET

5.1、常见客户问题

5.1.1、客户反馈建立 onenet 连接后，概率性会收到平台下发数据。

模组的 edrx 会导致下行数据有延迟，建议对于实时性要求比较高的应用场景，可以把 edrx 和 psm 先关闭，业务处理后，再打开。

5.1.2、onenet 自恢复功能

进入 PSM 前需要保存 onenet 一些重要参数，比如 msgid,不然做不到自恢复，用户可以通过调用接口(QI_SecureData_Store)保存到安全数据区。

6、LWM2M

7、其他

7.1、如何锁频点

[2017-07-12_10:28:35:853]AT+QOPS -----搜索当前附近基站信息
[2017-07-12_10:28:56:035]+QOPS: 2,"CHINA UNICOM GSM","UNICOM","46001"---联通
[2017-07-12_10:28:56:035]1,5504,2B53,37,40,111-----这些都是频点信息
[2017-07-12_10:28:56:097]2,5504,56E3,27,36,109
[2017-07-12_10:28:56:097]3,5504,2B55,34,35,123
[2017-07-12_10:28:56:097]4,5504,56E1,35,31,114
[2017-07-12_10:28:56:097]5,5504,44A3,2C,30,643
[2017-07-12_10:28:56:097]6,5504,56E2,34,28,121
[2017-07-12_10:28:56:097]7,5504,582B,1C,30,110
[2017-07-12_10:28:56:097]8,5504,47A7,35,28,122
[2017-07-12_10:28:56:097]9,5504,2871,1F,28,120
[2017-07-12_10:28:56:097]10,5504,2B54,1C,27,118
[2017-07-12_10:28:56:097]+QOPS: 3,"CHINA MOBILE","CMCC","46000"---移动
[2017-07-12_10:28:56:105]1,550A,37A8,30,47,48
[2017-07-12_10:28:56:105]2,550A,01FB,0F,38,82
[2017-07-12_10:28:56:105]3,550A,6D46,1E,57,5
[2017-07-12_10:28:56:105]4,550A,2BB9,3E,36,45
[2017-07-12_10:28:56:105]5,5665,01EB,0E,34,84
[2017-07-12_10:28:56:105]6,550A,2E3C,22,33,88
[2017-07-12_10:28:56:105]7,550A,5DF7,20,34,1
[2017-07-12_10:28:56:105]8,550A,2BB7,0F,39,90
[2017-07-12_10:28:56:105]9,550A,37A6,37,32,51
[2017-07-12_10:28:56:105]10,550A,3A40,2B,31,89
[2017-07-12_10:28:56:105]+QOPS: 1,"46020","46020","46020"
[2017-07-12_10:28:56:114]1,4103,0105,2C,5,1015
[2017-07-12_10:28:56:114]2,4103,0202,2D,8,1005
[2017-07-12_10:28:56:114]OK

[2017-07-12_10:30:12:461]AT+QLOCKF=2,0,114
---模块锁到 114 频点上，参数 2 的意思:开启锁频功能并开机自动切换到上次锁定的频点
[2017-07-12_10:30:12:461]OK
[2017-07-12_10:30:23:656]AT+QPOWD=0 ---关机
[2017-07-12_10:30:23:656]OK
[2017-07-12_10:30:47:326]AT+CGREG?
[2017-07-12_10:30:47:326]+CGREG: 0,1----已注册上网

[2017-07-12_10:30:47:326]OK
[2017-07-12_10:31:09:635]AT+QENG=1,0 ----查询当前小区

[2017-07-12_10:31:09:635]OK
[2017-07-12_10:31:11:752]AT+QENG?
[2017-07-12_10:31:11:752]+QENG: 1,0

[2017-07-12_10:31:11:752]+QENG: 0,460,01,5504,56e1,114,53,-83,59,59,5,12,x,x,x,x,x,x,x
------当前模块注册频点为 114
[2017-07-12_10:31:11:752]OK

外设部分

1、GPIO

2、ADC

2.1、ADC 使能后，没有 callback 上报

我们的 ADC 检测是通过发消息的机制来触发的，当 ADC 检测到数据，会发送消息到注册 ADC 的 task，在这个 task 中，客户必须调用接口（Ql_OS_GetMessage）来处理消息，才会上报 callback
注：软件 timer、普通 EINT、也是同样的机制，硬件 timer 和 fast eint 直接中断触发，不通过发送消息

2.2、ADC 是否可以采集大于 1.8V 的电压

可以，但是用户需要外部做分压处理，保证到达模块 ADC0 引脚的电平在 0-1800mv 之间，通过比例间接算出当前电压。

2.3、ADC 采样是多少位的 D/A

10bit

3、ENIT

3.1、模块支持的最小消抖时间是多少？

50ms,即模块只能检测到大于 50ms 的电平变化

3.2、我们的 EINT 是电平还是上升沿触发

都支持
两者区别：
电平触发：在高或低电平保持的时间内触发
边沿触发：由高到低或由低到高这一瞬间触发

3.3、注册 EINT 时返回-16

在注册 EINT 之前，这个引脚已经做其他功能，比如 GPIO，SD 等

注：用户受其他 MCU 外部中断配置影响，设置 EINT 之前，需要先初始化 GPIO，我们 opencpu 是不需要的。

4、PWM

4.1、如何输出指定频率 PWM 波形

PWM frequency=(pwmSrcClk / pwmDiv)/(lowPulseNum+highPulseNum).

PWM frequency：频率
pwmSrcClk：时钟源
pwmDiv：分频
lowPulseNum：低电平比例
highPulseNum：高电平比例

4.2、opencpu 中仅同时支持一路 PWM

4.3、目前哪些引脚支持 PWM 功能

SPI_CS、NETLIGHT、RTS_AUX、GPIO3

5、IIC

5.1、返回-34 错误

1、客户的硬件连接有问题，比如虚焊等
2、从器件地址错误
举个例子：

使用芯片的 datasheet 信息

Table 9.I2C Address

SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	SAD0	W/R
0	1	0	0	1	1	SAO	0/1

Table 10.SAD+Read/Write patterns

Command	SAD[6:1]	SAD[0]=SA0	R/W	SAD+R/W
Read	010011	0	1	01001101(4dh)
Write	010011	0	0	01001100(4ch)
Read	010011	1	1	01001111(4fh)
Write	010011	1	0	01001110(4eh)

应该使用的器件地址：我们的 IIC 是需要传入 8 位(0X4C)地址的，有些客户只传入 7 位(0X26）没有包含最后一个读写位。

5.2、代码中对于读写接口是否要写入不同的地址（根据最后一个读写位）

不需要，我们的代码会根据你调用的接口不同，来更改最后一位的值，用户只需要统一传入 0 或者 1 即可

5.3、如何实现控制多路的 IIC

我们最多支持 6 路的 IIC，客户初始化（仅一次）后，需要支持多少路 IIC 就配置几次即可。注意：如果客户需要挂载多路的 IIC，需要确定这几路的 IIC 器件地址必须不同，因为我们是根据不同的器件地址来进行操作的。

5.4、IIC 操作时，返回-34，从波形上面可以看出返回的 ACK 大约有 1v 左右

上拉电阻选择过大，把 10K 电阻更改成 4.7K

5.4、硬件 IIC 外挂 2 个设备，设备需要的速率不同所以需要 config 配置切换速率，切换时出现-310 错误

我们底层处理：同一个 channel 下可以有 6 个 owner，根据不同的器件地址 config 会分配不同 owner。当该器件地址已经存在时 return-1，但是可能-1 是溢出数据，所以实际 return 值是 255，本应该返回该器件地址已经存在的错误，但是由于没法匹配-1 的 if 判断导致错误的进入太多从机设备错误，即-310。实际上不影响功能使用。

6、SPI

6.1、使用我们 example_SPI，读取寄存器数据全为 0

客户硬件电路设计时，MISO 和 MOSI 引脚设置反了
注：一般芯片上标注 DO(MISO)、DI(MOSI)

6.2、使用我们的 SPI 时，写入 512 个字节后，读出来时，只有 256 个字节是成功的

查看 datasheet 发现芯片规定一页为 256 个字节

6.3、SPI 可以在 Init 时定义一个 clk 管脚，只是不要使用特定的 PINNAME_PCM_CLK。

然后通过操作 CS 管脚选择不同的设备，从而实现用硬件的管脚和硬件 spi 挂载多个设备。

SDK 中 Config 时 channel 只能是 1 应该是描述错误这个 channal 可以 0~254，但必须和 init 时一致。

6.4、SPI 时钟频率是否可以配置

可以配置，配置范围 30KHz——52MHz。实际选择频率也需要参考所选 SPI 芯片的 datasheet。

TIME

1、timer

1.1、简介

我们 NB 模块，提供了两种类型的 timer。一种是软件 timer，一种是快速 timer（GP_TIMER）。在 opencpu 中每个 task 可以最多同时使用 10 个软件 timer，一个工程只能有 1 个 GP_TIMER.软件 timer 是通过传递消息来调用的，GP_TIMER 直接通过中断来触发的，所以优先级要比软件 timer 高。

1.2、常见客户问题

1.2.1、软件 timer 定时不准的问题

因为软件 timer 是通过发送消息的方式来触发 callback，所以如果 timeout 后，当前 task 不能及时处理 message，将会导致定时延迟。

比如代码中有如下操作会造成延迟：

1、 Ql_Sleep()

2、 Mutex、EVENT 等阻塞接口

3、 复杂业务’

1.2.2、GP timer 频繁开关导致无法正常工作

1.2.3、GP timer 的 callback 中不能增加锁操作

MTK 规定禁止在 HISR 里获取互斥量时等待，比如做锁的操作

如果需要在 callback 中发送消息，请使用接口（Ql_OS_SendMessageFromISR）

2、RTC

1.1、简介

NB 模块支持一路 RTC 功能，用户可以通过设置 RTC 来定时触发 callback，当模组进入 deep sleep 模式后，可以通过 RTC 来唤醒模块

1.2、常见客户问题

1.2.1、RTC 定时器设置 2/4/6/8 小时，都会大约 90s 钟被唤醒

PSM 状态下，模块未发生数据交互，T3412 定时器超时，模块会周期性 TAU，这时会唤醒模块，唤醒的现象和 RTC 一样。

1.2.2、当模组从 deepsleep 模式唤醒后，再次 start rtc 会返回-4

如果是循环的 RTC，当模组从 deep sleep 唤醒后，实际上 RTC 数据还有保存，RTC timer 也还在运行，所以再次 start 会返回-4，提示用户，RTC 已经 start。如果需要再次 start，需要先 stop。

电源部分

1、常见客户问题

1.1、基本要求

模块的电源设计对其性能至关重要。BC26-OpenCPU 可使用低静态电流、输出电流能力达到0.5A 的LDO 作为供电电源，也支持Li-MnO2/2S 电池供电；其电源输入电压范围为2.1V~3.63V。模块在数传工作中，必须确保电源跌落不低于模块最低工作电压2.1V，否则模块会异常。

1.2、减少电压跌落

为了确保更好的电源供电性能，在靠近模块VBAT 输入端，建议并联一个低ESR（ESR=0.7Ω）的100uF 的钽电容，以及100nF、100pF 和22pF 滤波电容。同时，建议在靠近VBAT 输入端增加一个TVS管以提高模块的浪涌和ESD 承受能力。原则上，若VBAT 走线越长，则要求线宽越宽。VBAT 输入端参考电路如下图所示：

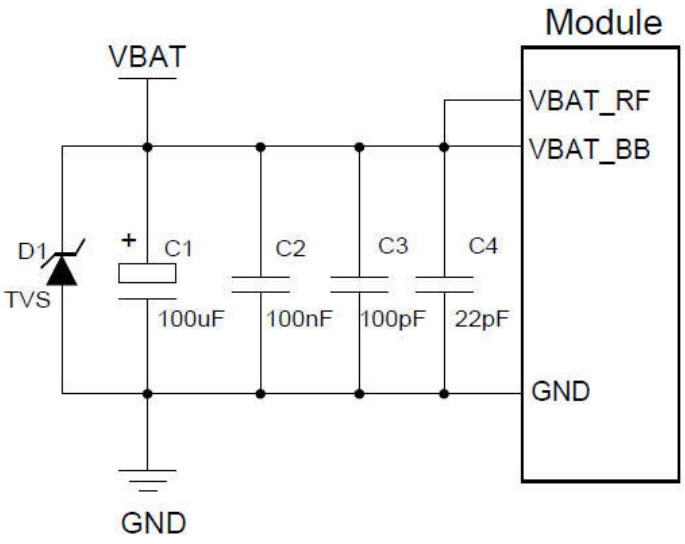


图 5：VBAT 输入端参考电路

1.3、模块在开机后，不断重启

如果用户的设备出现在开机后的一段时间内，频繁的重启，就需要考虑是否因为在注册网络阶段，由于电压跌落造成的，可以通过示波器来监控 VBAT 的电压。

注：有时为了测试客户设备，外部 VBAT 飞线，需要注意飞线不能过长。

1.5 能否将 5V 电压通过二极管降压后给模块供电？

不建议采用此方式，因为二极管压降随电流变化而变化，可能会因模块供电电压不稳定而导致模块工作异常。

串口部分

1、常见客户问题

1.1、串口数量

MC20/MC60 系列模块，由于 AUX 串口默认和 GNSS 的 UART 通讯，所以只有 MAIN 口和 DEBUG 口两个 UART。
其他 2G 模块：有三个串口：MAIN、DEBUG、AUX
注：为了方便 APP 和内核之间进行通讯，我们还提供了三个虚拟串口，其中虚拟串口 3 默认提供给 RIL 接口使用。所以用户如果需要和底层进行数据交互，可以使用虚拟串口 1 和虚拟串口 2。

1.2、各串口功能

MAIN 串口（UART1）：
1、 下载 APP BIN
2、 打印应用 log
3、 和外设进行通讯
Debug 串口（URAT2）
1、 抓取 catcher log（advance mode）
2、 打印应用 log （basic mode）
3、 和外设进行通讯（basic mode）
AUX 串口（UART3）
1、 打印应用 log
2、 和外设进行通讯
注：MC20/MC60 系列模块此口默认和 GPS 连接，用户不可控制。

虚拟串口 1（VIRTUAL_PORT1）
和底层进行数据交互

虚拟串口 2（VIRTUAL_PORT2）
和底层进行数据交互

虚拟串口 3（VIRTUAL_PORT3）

RIL 功能占用，用来交互 AT

1.3、串口 buffer 必须要读空

当应用层收到读取串口 buffer 的 callback 时，在 callback 中需要 while（1）读取 buffer 中的数据，直到把 buffer 数据读空，才返回,否则串口再次来数据将无法接收。可以参考我们 SDK 中的 main.c

1.4、底层 buffer 发送消息的条件

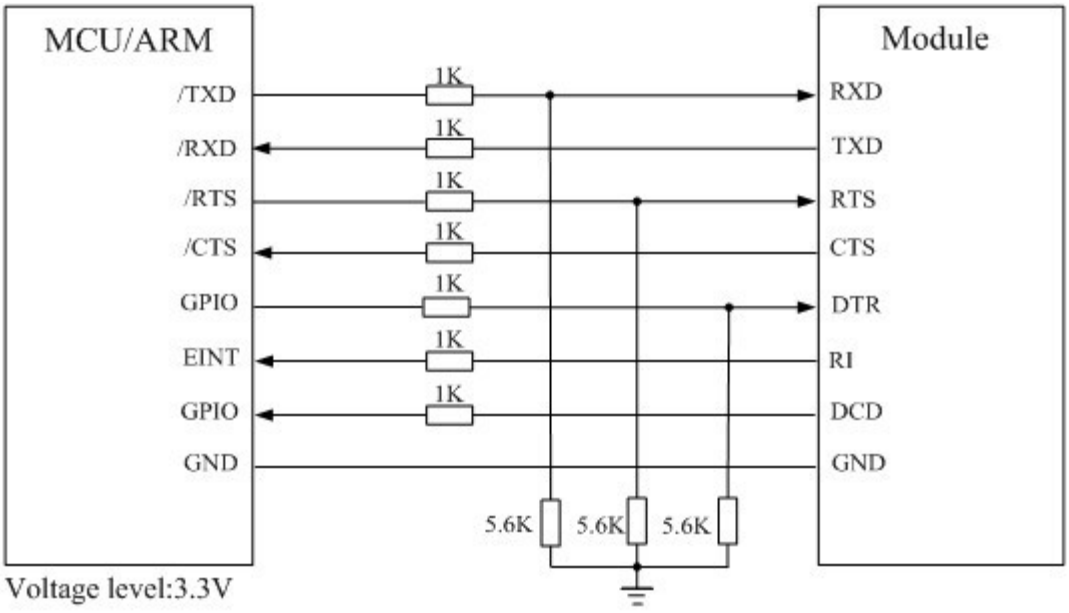
当收到的 buffer 数据超过最大阀值的 2/3 或者连续 4 个字节时间间隔没有收到数据时。

1.5、当模块通过 QI_Sleep_Enable 进入睡眠模式后，将不再能接收数据

如果用户需要发送数据给模块，可以先通过外部中断，唤醒模块，最好能增加个延迟，然后再发送数据，保证模块当前已经被唤醒。

1.6、串口电平如何匹配

1) 如果 MCU 的串口电平是 3.3V，匹配电路如图所示：



2) 如果 MCU 的串口电平是 3V，则将图 3 中的 5.6K 电阻换成 15K。

1.7、DEBUG 口出现乱码，或者异常输出

DEBUG 口开机默认会输出 SDK 版本信息。如果客户使用不同的波特率那么可能就会显示乱码，所以客户一般反馈的开机乱码可能跟此有关。
另外需要考虑 DEBUG 口的打印，接口 Ql_Debug_Trace 是默认从 debug 口输出的，如果使用不当可能造成乱码。

程序设计部分

1、常见客户问题

1.1、取值范围与打印问题

对应的数据类型需要用对应的类型打印。
例如客户 u32 的数据，范围是 0xFFFFFFFF，打印出来返回-1。客户使用%d、%ld 打印。u32 位无符号 int 型，打印应使用%u。

1.2、程序栈溢出

给 task 设置的栈空间小了会导致栈溢出，修改栈空间大小可解决，如果程序中需要用到大 buffer 空间，建议使用 malloc 来申请资源。。