海思 open 开发过程常见疑问 V1.0

1）Q：关于管脚是否可以模式切换问题？比如没有用到 SPI 口，如何将 SPI 作为 GPIO 使用？

A：（1）如下 mode 模式可配置。

3.2.2.2. GPIO List

Table 2: BC35 Multiplexing Pins

Pin No.	Pin Name	RESET	MODE1	MODE2	MODE3	MO
1	PINNAME_SPI_CS	I/PN	SPI_CS	GPIO	EINT	
5	PINNAME_SPI_SO	I/PN	SPI_SO	GPIO	EINT	
6	PINNAME_SPI_CLK	I/PN	SPI_CLK	GPIO	EINT	
7	PINNAME_SPI_SI	I/PN	SPI_SI	GPIO	EINT	
8	PINNAME_UART_TX3	I/PN	UART_TX3	GPIO	EINT	
9	PINNAME_UART_RX3	I/PN	UART_RX3	GPIO	EINT	

（2） 配置步骤。

3.2.2.4. GPIO Usage

The following shows how to use the multifunctional GPIOs:

Step 1: GPIO initialization. *Call ql_gpio_init* function sets the specified pin as the GPIO function, and initializes the configurations, which includes direction, level and pull selection.

Step 2: GPIO control. When the pin is initialized as GPIO, the developers can call the GPIO related API functions to change the GPIO level.

Step 3: Release the pin. If developers do not want use this pin no longer, and need to use this pin for other purposes (such as PWM, EINT), they must call *ql_gpio_uninit* to release the pin first. This step is optional.

3.2.2.5.1. ql_gpio_init

This function enables the GPIO function of the specified pin, and initializes the configurations, which includes direction, level and pull selection.

- **Prototype**

```
QOCPU_RET ql_gpio_init  (     Enum_PinName pinName,
                              Enum_PinDirection dir,
                              Enum_PinLevel   level
                         )
```

- **Parameters**

2）Q:ql_uart_open 的回调函数，是否时一帧数据接收完毕才回调，其一帧接收的最大长度是多少？

A：最新的 SDK 划分发送和接受最大是 1500 字节。物理串口是在判断一帧的机制是 100ms 内物理串口没有收到数据，就说明一帧数据接收完成了。这时候就会调用初始化带入的处理函数。

NOTES

The receive buffer size of physic UART is 1500 bytes. The receive buffer and send buffer size of virtual UART are 2763 bytes respectively.

3.2.1.3. API Functions

3）Q:Backup Critical Data 共有 10 个 block，每个 block 多大，一次可写入和读出的最大长度时多少，是否支持任意位置存取？

A：neul_kv_set 和 eul_kv_get 来操作。一个 block 理论上是 65355 个字节，但是实际总共存储内存是 8k，10 个 block 内部计划分出 5k 的空出来，每个 block512；下个版本这里将会做限制，每个 block 最大一次性存取 512 字节。

不支持任意位置存取，只能从首字节读取存取数据长度内的数据，单个存取也是要占用这个 id 的，所以说这个 id 内的数据不支持单个自己读取。

4）Q:Allocate a block of memory from internal RAM 动态分配内存，提供用来动态分配的内存池的大小是多少？

A：支持电信 iot 平台的 RAM 最大剩余 11k，对用户说 10k，因为这个统计会有误差
支持移动 onet 版本的 RAM 最大 18k，对用户说 16k。

5)Q:osDelay(100)这个是 delay 多久，1S 还是 100ms，系统时钟中断频率是 10ms 还是 1ms？

A:单位是 ms，为 100ms。

6）Q:在 vuart_recieve_handle 等回调函数中能否用信号量等待来解决共享资源互斥问题，因为中断一般只能发送、不能等待,还是必须用 non_os_enter_critical()、non_os_exit_critical()来解决。

A：中断中不能调用信号量等待。一般全局变量在中断或者不通任务间调用，理论上要进入极限状态，不解决怕有问题，加两个函数是为了程序的健壮性和稳定性。如因为 m0 核貌似支持中断抢断。

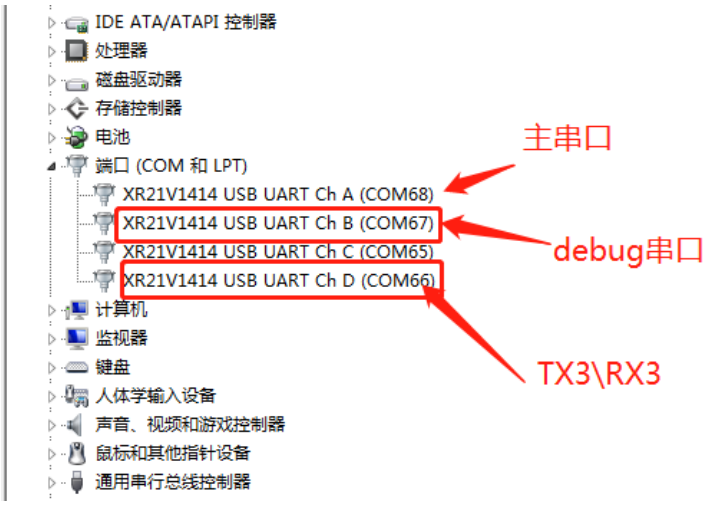
7) Q:有关 UART 问题，需要一个物理主串口与终端通信；一个串口打印用户 APP debug 信息；疑问是模组本身内核 debug 信息通过哪个口输出，比如抓 log 等.

A:模组有三个串口:我们 demo 中的 app_debug 用的是 uart1，这个串口在波特率小于 57600 时数据接收唤醒；debug 串口是独立的，上电就会默认配置 log 输出，建议用户终于统一使用模组的 debug 口进行 log 打印，debug 调用函数在 qmisc.h 中；还有一路串口请参照 demo 中的 example_uart.c. 选 uart3 作为 app_debug 输出，UART1 与终端通信主串口。

8).Q:有关 FOTA 升级问题，目前标准固件与用户 APP 合并形式。若差分升级，APP 差分文件与固件差分文件合并进行 DFOTA 升级？还是如何实现。

A：open 开发烧写目前最终生成 fwpkg 文件;差分文件就是取最新需要升级 open 生成的 fwpk 固件包与烧写进模组的 open 生成的 fwpkg 固件包做差分包即可；升级步骤与标准固件差分包一致；

9）BC95JB TE-B EVVB 串口映射说明如下：



10）海思 V150 平台内核说明如下：
Boudica 芯片拥有 3 个 CPU，分别为安全核（S 核）、协议核（C 核）、应用核（A 核），采用共享内存方式实现核间通信。
S 核：负责启动 A、C 核，拥有擦除和写内置 flash 的权限。
C 核：当前运行 FreeRTOS（后面会切 LiteOS），主要包含 NB-IoT 及 LWIP 协议栈
A 核：当前运行 FreeRTOS（后面会切 LiteOS），主要负责 AT 命令解析、外置 flash 读写以及其他上层应用、协议等（ LWM2M、FOTA ）。

The module contains 3 processor cores: Application Processor, Protocol Processor and Security Processor.
Security Processor : ARM Cortex-M0 core @ 51.75 MHz
Protocol Processor : ARM Cortex-M0 core @ 51.75 MHz
Application Processor :ARM Cortex-M0 core @ 51.75 MHz

11）一旦程序跑飞了，内核看门狗在 30S 内会产生复位动作。30s 的这段时间 A 核处于未知状态，是不会有 log 出来。重点查数据接收处理那块的内存逻辑；因海思平台开发内存空间较小，虚拟串口不可频繁大数据操作交互。

12）Q：UART3 能够作为与外设进行交互数据串口吗，类似 UART1 一样。与 UART1 有什么差异。

A:UART3 就是比主串口 UART1 少了一个唤醒功能；UART3 禁止了睡眠，回头用了 UART3 使用低功耗的要加一个函数，初始化完串口加个这个函数 osAddStopClocksVeto();但对整体功耗有影响，内核不会进入深度睡眠；若不用,立马执行 osRemoveStopClocksVeto()；深休眠后 uart3 不支持唤醒。

13）Q：进入入网后就会卡住，自动重启问题。

A：根本原因是可能用户函数里使用了大的局部变量，导致任务堆栈溢了；App_debug 和系统的 debug 打印口都不支持大数据打印，只做程序调试定位的作用，建议 buffer 控制在 400 字节，大了会造成模组内存的资源不够，导致程序执行出错；超过这个字节数的数据如需打印请直接使用 ql_uart_write，直接在用户处理程序打印；虚拟串口过来的大数据和网虚拟串口发的大数据一定要加长度判断，超过长度不能打印，不然程序必挂

14）Q：客户使用 open UART3 打印 APP debug 会出现乱码现象及不稳定。

A：由于未打开压力域，串电导致。找到 ql_bank_io 函数接口，打开电压域。备注：ql_io_bank_open(IO_BANK_L1,VDD_IO_LEVEL_3V3);初始化加个这个,在 ql_io_bank.h 这个没有用 gpio 的初始化外设需要开下电压域

```
    {
        QDEBUG_NORMAL("uart port1 init error");
    }
    if(ql_uart_init(UART_PORT3) != QOCPU_RET_OK)
    {
        QDEBUG_NORMAL("uart port3 init error");
    }
    if(ql_io_bank_open(IO_BANK_L1,VDD_IO_LEVEL_3V0) != QOCPU_RET_OK )
    {
        QDEBUG_NORMAL("user open IO_BANK_L1 err\r\n");
    }

    uart3_create_queue();
    uart1_config.baudrate=115200;
    uart1_config.data_bits=UART_DATA_BITS_8;
    uart1_config.parity=UART_PARITY_NONE;
    uart1_config.stopbits=UART_STOP_BITS_1;

    uart3_config.baudrate=115200;
    uart3_config.data_bits=UART_DATA_BITS_8;
    uart3_config.parity=UART_PARITY_NONE;
    uart3_config.stopbits=UART_STOP_BITS_1;

    if( ql_uart_open(UART_PORT1, &uart1_config, uart1_recieve_handle) != QOCPU_RET_OK )
    {
        QDEBUG_NORMAL("user open  err\r\n");
    }
    if( ql_uart_open(UART_PORT3, &uart3_config, uart3_recieve_handle)!=QOCPU_RET_OK )
    {
        QDEBUG_NORMAL("user open  err\r\n");
    }
    APP_DEBUG("\r\n<-- OpenCPU: UART Example -->\r\n");
    for(;;)
```

15）Q：任务调度器是 1ms 执行一次？还是 10ms？另外用户需要超时等待判断时的时间计时的需求：需要一个定时器来计时。

A：任务调度器是 1ms 执行一次；osTimerNew 创建一个定时器，osTimerStart 启动它。这个当定时时间<16ms 进不了深休眠，>16ms 的话模组会先进深睡然后定时到了在唤醒。

16）具体中断功能的引脚是可以将模组从 PSM 唤醒的。