ECx00U&EGx00U 系列

QuecOpen 串口 API 参考手册

LTE Standard 模块系列

版本：1.0

日期：2021-08-30

状态：受控文件

上海移远通信技术股份有限公司（以下简称“移远通信”）始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 5108 6236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，请随时登陆网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

移远通信提供该文档内容以支持客户的产品设计。客户须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，移远通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，尽管移远通信采取了商业范围内的合理努力来提供尽可能好的体验，但本文档和其所涉及服务是在“可用”基础上提供给您的。移远通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

使用和披露限制

许可协议

除非移远通信特别授权，否则我司所提供硬软件、材料和文档的接收方须对接收的内容保密，不得将其用于除本项目的实施与开展以外的任何其他目的。

版权声明

移远通信产品和本协议项下的第三方产品可能包含受移远通信或第三方材料、硬软件和文档版权保护的相关资料。除非事先得到书面同意，否则您不得获取、使用、向第三方披露我司所提供的文档和信息，或对此类受版权保护的资料进行复制、转载、抄袭、出版、展示、翻译、分发、合并、修改，或创造其衍生作品。移远通信或第三方对受版权保护的资料拥有专有权，不授予或转让任何专利、版权、商标或服务商标权的许可。为避免歧义，除了正常的非独家、免版税的产品使用许可，任何形式的购买都不可被视为授予许可。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，移远通信有权追究法律责任。

商标

除另行规定，本文档中的任何内容均不授予在广告、宣传或其他方面使用移远通信或第三方的任何商标、商号及名称，或其缩略语，或其仿冒品的权利。

第三方权利

您理解本文档可能涉及一个或多个属于第三方的硬软件和文档（“第三方材料”）。您对此类第三方材料的使用应受本文档的所有限制和义务约束。

移远通信针对第三方材料不做任何明示或暗示的保证或陈述，包括但不限于任何暗示或法定的适销性或特定用途的适用性、平静受益权、系统集成、信息准确性以及与许可技术或被许可人使用许可技术相关的不侵犯任何第三方知识产权的保证。本协议中的任何内容都不构成移远通信对任何移远通信产品或任何其他硬软件、设备、工具、信息或产品的开发、增强、修改、分销、营销、销售、提供销售或以其他方式维持生产的陈述或保证。此外，移远通信免除因交易过程、使用或贸易而产生的任何和所有保证。

免责声明

1) 移远通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。

2) 移远通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。

3) 移远通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则移远通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，移远通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。

4) 移远通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2021-03-02	Braden HE	文档创建
1.0	2021-08-30	Braden HE	受控版本

目录

表格索引

1 引言

移远通信 ECx00U 系列和 EGx00U 模块支持 QuecOpen®方案；QuecOpen®是基于 RTOS 的嵌入式开发平台，可简化 IoT 应用的软件设计和开发过程。有关 QuecOpen®的详细信息，请参考***文档 [1]***。

本文档主要介绍在 QuecOpen®方案下，ECx00U 系列和 EGx00U 模块的串口 API、相关示例及功能调试。模块当前支持软件配置 3 路物理串口和 3 路 USB 虚拟串口，其中物理串口的 RX 默认使用 DMA 模式，TX 默认使用 FIFO 模式。

1.1. 适用模块

表 1：适用模块

模块系列	模块
ECx00U	EC200U 系列
	EC600U 系列
EGx00U	EG500U-CN
	EG700U-CN

2 串口 API 介绍

2.1. 头文件

串口 API 的头文件为 *ql_uart.h*，位于 *components\ql-kernel\inc* 目录下。若无特别说明，本文档所述头文件均位于该目录下。

2.2. 函数概览

表 2：函数概览

函数	说明
ql_uart_set_dcbconfig()	配置串口属性
ql_uart_get_dcbconfig()	获取串口属性配置
ql_uart_open()	打开指定串口
ql_uart_close()	关闭指定串口
ql_uart_write()	向指定串口写入数据
ql_uart_read()	通过指定串口读取数据
ql_uart_register_cb()	注册串口事件回调函数

2.3. 函数详解

2.3.1. ql_uart_set_dcbconfig

该函数用于配置串口属性。通过该函数配置的参数需调用 *ql_uart_open()*重新打开串口后生效。

- **函数原型**

```
ql_uart_errcode_e ql_uart_set_dcbconfig(ql_uart_port_number_e port, ql_uart_config_s *dcb)
```

- **参数**

port：
[In] 串口编号。请参考***第 2.3.1.1 章***。

dcb：
[In] 串口属性配置。请参考***第 2.3.1.2 章***。

- **返回值**

请参考***第 2.3.1.3 章***。

2.3.1.1. ql_uart_port_number_e

模块当前支持软件配置 3 路物理串口和 3 路 USB 虚拟串口。串口编号枚举信息定义如下：

```
typedef enum
{
    QL_PORT_NONE = -1,
    QL_UART_PORT_1,
    QL_UART_PORT_2,
    QL_UART_PORT_3,
    QL_USB_PORT_AT,
    QL_USB_PORT_MODEM,
    QL_USB_PORT_NMEA,
    QL_PORT_MAX,
}ql_uart_port_number_e;
```

- **参数**

参数	描述
QL_UART_PORT_1	串口 1
QL_UART_PORT_2	串口 2

QL_UART_PORT_3	串口 3
QL_USB_PORT_AT	USB AT 端口
QL_USB_PORT_MODEM	USB modem 端口
QL_USB_PORT_NMEA	USB NMEA 端口

2.3.1.2. ql_uart_config_s

串口属性配置结构体定义如下：

```
typedef struct
{
    ql_uart_baud_e baudrate;
    ql_uart_databit_e data_bit;
    ql_uart_stopbit_e stop_bit;
    ql_uart_paritybit_e parity_bit;
    ql_uart_flowctrl_e flow_ctrl;
}ql_uart_config_s;
```

- **参数**

类型	参数	描述
ql_uart_baud_e	*baudrate*	波特率设置，默认为 115200 bps。请参考***第 2.3.1.4 章***。
ql_uart_databit_e	*data_bit*	数据位设置，默认为 8 位。请参考***第 2.3.1.5 章***。
ql_uart_stopbit_e	*stop_bit*	停止位设置，默认为 1 位。请参考***第 2.3.1.6 章***。
ql_uart_paritybit_e	*parity_bit*	校验位设置，默认无校验。请参考***第 2.3.1.7 章***。
ql_uart_flowctrl_e	*flow_ctrl*	流控设置，默认不开启。请参考***第 2.3.1.8 章***。

2.3.1.3. ql_uart_errcode_e

串口错误码表示函数是否执行成功，若失败则返回错误原因。枚举信息定义如下：

```
typedef enum
{
    QL_UART_SUCCESS                 = 0,
    QL_UART_EXECUTE_ERR             = 1|QL_UART_ERRCODE_BASE,
    QL_UART_MEM_ADDR_NULL_ERR,
```

```
    QL_UART_INVALID_PARAM_ERR,
    QL_UART_OPEN_REPEAT_ERR,
    QL_UART_NOT_OPEN_ERR          = 5|QL_UART_ERRCODE_BASE,
} ql_uart_errcode_e
```

- **参数**

参数	描述
QL_UART_SUCCESS	函数执行成功
QL_UART_EXECUTE_ERR	函数执行失败
QL_UART_MEM_ADDR_NULL_ERR	参数地址为 NULL
QL_UART_INVALID_PARAM_ERR	参数无效
QL_UART_OPEN_REPEAT_ERR	串口重复打开
QL_UART_NOT_OPEN_ERR	串口未打开

2.3.1.4. ql_uart_baud_e

波特率信息枚举定义如下：

```
typedef enum
{
    QL_UART_BAUD_2400          = 2400,
    QL_UART_BAUD_4800          = 4800,
    QL_UART_BAUD_9600          = 9600,
    QL_UART_BAUD_14400         = 14400,
    QL_UART_BAUD_19200         = 19200,
    QL_UART_BAUD_28800         = 28800,
    QL_UART_BAUD_33600         = 33600,
    QL_UART_BAUD_38400         = 38400,
    QL_UART_BAUD_57600         = 57600,
    QL_UART_BAUD_115200        = 115200,
    QL_UART_BAUD_230400        = 230400,
    QL_UART_BAUD_460800        = 460800,
    QL_UART_BAUD_921600        = 921600,
    QL_UART_BAUD_1000000       = 1000000,
    QL_UART_BAUD_1843200       = 1843200,
    QL_UART_BAUD_2000000       = 2000000,       //only support uart2/3
    QL_UART_BAUD_2100000       = 2100000,
    QL_UART_BAUD_3686400       = 3686400,       //only support uart2/3
```

```
    QL_UART_BAUD_4000000    = 4000000,      //only support uart2/3
    QL_UART_BAUD_4468750    = 4468750       //only support uart2/3
}ql_uart_baud_e;
```

- **参数**

参数	描述
QL_UART_BAUD_2400	波特率为 2400 bps
QL_UART_BAUD_4800	波特率为 4800 bps
QL_UART_BAUD_9600	波特率为 9600 bps
QL_UART_BAUD_14400	波特率为 14400 bps
QL_UART_BAUD_19200	波特率为 19200 bps
QL_UART_BAUD_28800	波特率为 28800 bps
QL_UART_BAUD_33600	波特率为 33600 bps
QL_UART_BAUD_38400	波特率为 38400 bps
QL_UART_BAUD_57600	波特率为 57600 bps
QL_UART_BAUD_115200	波特率为 115200 bps
QL_UART_BAUD_230400	波特率为 230400 bps
QL_UART_BAUD_460800	波特率为 462800 bps
QL_UART_BAUD_921600	波特率为 921600 bps
QL_UART_BAUD_1000000	波特率为 1000000 bps
QL_UART_BAUD_1843200	波特率为 1843200 bps
QL_UART_BAUD_2000000	波特率为 2000000 bps（仅支持串口 2 和串口 3）
QL_UART_BAUD_2100000	波特率为 2100000 bps
QL_UART_BAUD_3686400	波特率为 3686400 bps（仅支持串口 2 和串口 3）
QL_UART_BAUD_4000000	波特率为 4000000 bps（仅支持串口 2 和串口 3）
QL_UART_BAUD_4468750	波特率为 4468750 bps（仅支持串口 2 和串口 3）

2.3.1.5. ql_uart_databit_e

数据位枚举信息定义如下：

```
typedef enum
{
    QL_UART_DATABIT_7 = 7,
    QL_UART_DATABIT_8 = 8,
}ql_uart_databit_e;
```

- **参数**

参数	描述
QL_UART_DATABIT_7	数据位为 7 位（暂不支持）
QL_UART_DATABIT_8	数据位为 8 位

2.3.1.6. ql_uart_stopbit_e

停止位枚举信息定义如下：

```
typedef enum
{
    QL_UART_STOP_1   = 1,
    QL_UART_STOP_2   = 2,
}ql_uart_stopbit_e;
```

- **参数**

参数	描述
QL_UART_STOP_1	停止位为 1 位
QL_UART_STOP_2	停止位为 2 位

2.3.1.7. ql_uart_paritybit_e

校验位枚举信息定义如下：

```
typedef enum
{
    QL_UART_PARITY_NONE,
    QL_UART_PARITY_ODD,
```

```
    QL_UART_PARITY_EVEN,
}ql_uart_paritybit_e;
```

- **参数**

参数	描述
QL_UART_PARITY_NONE	无校验位
QL_UART_PARITY_ODD	启用奇校验
QL_UART_PARITY_EVEN	启用偶校验

2.3.1.8. ql_uart_flowctrl_e

流控枚举信息定义如下：

```
typedef enum
{
    QL_FC_NONE = 0,
    QL_FC_HW,
}ql_uart_flowctrl_e;
```

- **参数**

参数	描述
QL_FC_NONE	禁用硬件流控
QL_FC_HW	启用硬件流控

2.3.2. ql_uart_get_dcbconfig

该函数用于获取串口属性配置。

- **函数原型**

```
ql_uart_errcode_e ql_uart_get_dcbconfig(ql_uart_port_number_e port, ql_uart_config_s *dcb)
```

- **参数**

port：
[In] 串口编号。请参考***第 2.3.1.1 章***。

dcb：
[Out] 串口属性配置。请参考***第 2.3.1.2 章***。

- **返回值**

请参考***第 2.3.1.3 章***。

2.3.3. ql_uart_open

该函数用于打开指定串口。

- **函数原型**

```
ql_uart_errcode_e ql_uart_open(ql_uart_port_number_e port)
```

- **参数**

port：
[In] 串口编号。请参考***第 2.3.1.1 章***。

- **返回值**

请参考***第 2.3.1.3 章***。

2.3.4. ql_uart_close

该函数用于关闭指定串口。

- **函数原型**

```
ql_uart_errcode_e ql_uart_close(ql_uart_port_number_e port)
```

- **参数**

port：
[In] 串口编号。请参考***第 2.3.1.1 章***。

- **返回值**

请参考***第 2.3.1.3 章***。

2.3.5. ql_uart_write

该函数用于向指定串口写入数据。

- **函数原型**

```
int ql_uart_write(ql_uart_port_number_e port, unsigned char *data, unsigned int data_len)
```

- **参数**

port：
[In] 串口编号。请参考***第 2.3.1.1 章***。

data：
[In] 写入数据。

data_len：
[In] 写入数据的长度。单位：字节。

- **返回值**

负数　　函数执行失败，错误码请参考***第 2.3.1.3 章***。
其它值　实际数据的长度

2.3.6. ql_uart_read

该函数用于通过指定串口读取数据。

- **函数原型**

```
int ql_uart_read(ql_uart_port_number_e port, unsigned char *data, unsigned int data_len)
```

- **参数**

port：
[In] 串口编号。请参考***第 2.3.1.1 章***。

data：
[Out] 读取数据。

data_len：
[In] 读取数据的长度。单位：字节。

- **返回值**

负数　　函数执行失败，错误码请参考***第 2.3.1.3 章***。
其它值　实际读取数据的长度

2.3.7. ql_uart_register_cb

该函数用于注册串口事件回调函数。

- **函数原型**

```
ql_uart_errcode_e ql_uart_register_cb(ql_uart_port_number_e port, ql_uart_callback uart_cb)
```

- **参数**

port：
[In] 串口编号。请参考***第 2.3.1.1 章***。

uart_cb：
[In] 待注册的回调函数。若为 NULL，则系统为轮询模式。详情请参考***第 2.3.7.1 章***。

- **返回值**

请参考***第 2.3.1.3 章***。

2.3.7.1. ql_uart_callback

该函数为串口事件回调函数，用于通知客户端读取数据。

- **函数原型**

```
typedef void (*ql_uart_callback)(uint32 ind_type, ql_uart_port_number_e port, uint32 size)
```

- **参数**

ind_type：
[In] 串口事件类型。APP 侧可处理的事件包括：

QUEC_UART_RX_RECV_DATA_IND	接收数据事件
QUEC_UART_RX_OVERFLOW_IND	接收数据缓存溢出事件
QUEC_UART_TX_FIFO_COMPLETE_IND	发送 FIFO 完成事件

port：
[In] 事件来源的串口编号。请请参考***第 2.3.1.1 章***。

size：
[In] 串口接收数据大小。单位：字节。

- **返回值**

无

3 示例

本章节主要介绍在 APP 侧如何使用上述 API 进行串口开发及简单调试。示例中默认使用串口 2 进行测试。

3.1. 开发示例

移远通信 ECx00U 系列和 EGx00U QuecOpen SDK 代码中提供了串口 API 的示例文件，即 *components\ql-application\peripheral\uart_demo.c*。

该文件函数解析如下：

- *ql_uart_app_init()* //创建任务，运行示例前需调用该函数。
- *ql_uart_demo_thread()* //任务的执行函数，简单实现了串口的配置、打开和读、写数据，默认采用中断回调读取数据。
- *ql_uart_notify_cb()* //串口事件的回调函数，简单实现了数据的读取，可通过 coolwatcher 打印接收内容。

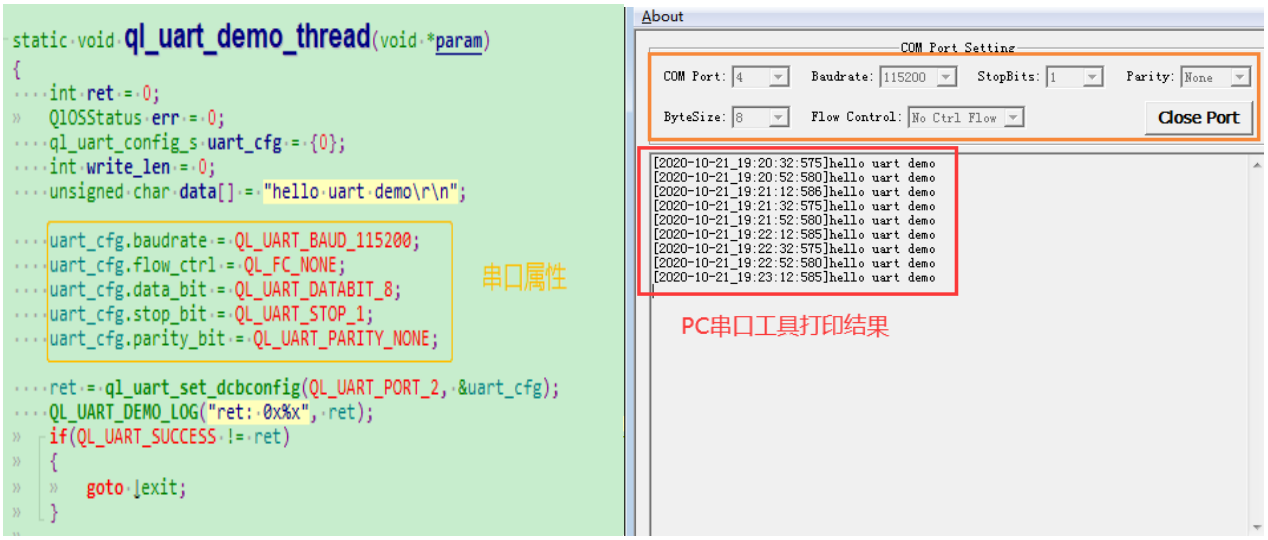
若需要运行该示例，则只需在线程 *ql_init_demo_thread* 中调用 *ql_uart_app_init()*，如下图所示：

```
static void ql_init_demo_thread(void *param)
{
    QL_INIT_LOG("init demo thread enter, param 0x%x", param);

    ql_gpio_app_init();
    ql_gpioint_app_init();
    ql_ledcfg_app_init();

#ifdef QL_APP_FEATURE_AUDIO
    ql_audio_app_init();
#endif
#ifdef QL_APP_FEATURE_LCD
    ql_lcd_app_init();
#endif
    ql_nw_app_init();
    //ql_datacall_app_init();
    ql_osi_demo_init();

#ifdef QL_APP_FEATURE_FILE
    ql_fs_demo_init();
#endif

    //ql_dev_app_init();
    ql_adc_app_init();
    ql_uart_app_init();
#ifdef QL_APP_FEATURE_HTTP
    //ql_http_app_init();
#endif
```

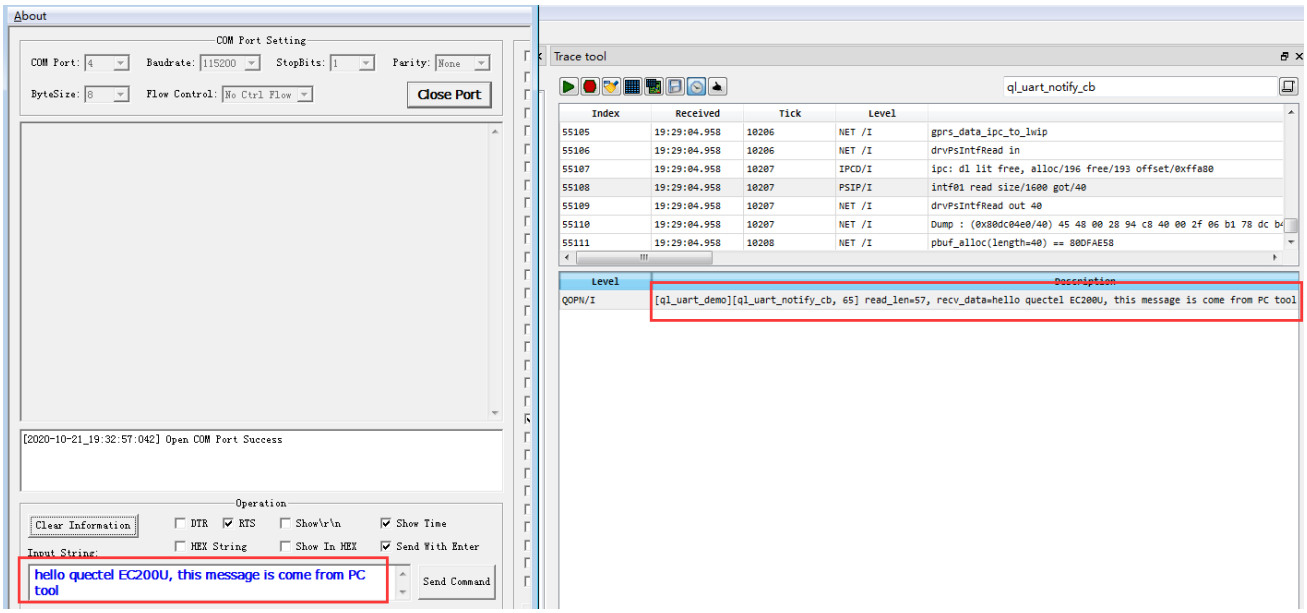
3.2. 功能调试

ECx00U 系列和 EGx00U QuecOpen 模块可通过移远通信 LTE OPEN EVB 进行串口功能调试。

将重新编译版本烧录到模块中，使用 USB 转串口线连接 LTE OPEN EVB 的 USB 口，并连接 coolwatcher 工具。在 PC 端打开串口工具（如 QCOM），按照示例中设置的串口属性进行连接，即可看到 PC 端的串口工具收到“hello uart demo”，如下图所示：



通过串口工具发送“hello quectel EC200U, this message is come from PC tool”后，可使用 coolwatcher 打印模块成功接收的串口数据，如下图所示。



4 附录 参考文档及术语缩写

表 3：参考文档

文档名称
[1] Quectel_ECx00U&EGx00U 系列_QuecOpen_CSDK_快速开发指导

表 4：术语缩写

缩写	英文全称	中文全称
API	Application Programming Interface	应用程序接口
APP	Application	应用
EVB	Evaluation Board	评估板
IoT	Internet of Things	物联网
LTE	Long-Term Evolution	长期演进
NMEA	NMEA (National Marine Electronics Association) 0183 Interface Standard	NMEA（美国国家海洋电子协会）0183 接口标准
PC	Personal Computer	个人计算机
RTOS	Real-Time Operating System	实时操作系统
RX	Receive	接收
SDK	Software Development Kit	软件开发工具包
TX	Transmit	发送
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发传输器
DMA	Direct Memory Access	直接存储器访问
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发传输器