ECx00U&EGx00U 系列

QuecOpen I2C 开发指导

LTE Standard 模块系列

版本：1.1

日期：2021-12-08

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。

隐私声明

为实现移远通信产品功能，特定设备数据将会上传至移远通信或第三方服务器（包括运营商、芯片供应商或您指定的服务器）。移远通信严格遵守相关法律法规，仅为实现产品功能之目的或在适用法律允许的情况下保留、使用、披露或以其他方式处理相关数据。当您与第三方进行数据交互前，请自行了解其隐私保护和数据安全政策。

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
-	2020-12-03	Chaos HUANG	文档创建
1.0	2021-09-28	Chaos HUANG	受控版本
1.1	2021-12-08	Chaos HUANG/ Evan MENG	更新枚举 ql_errcode_i2c_e 的成员（第 2.3.1.3 章节）。

目录

表格索引

图片索引

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 ECx00U 系列和 EGx00U QuecOpen®模块在 App 侧的 I2C 开发指导，包括 I2C API 函数、I2C 开发示例及功能调试。

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 I2C API

2.1. 头文件

I2C API 的头文件为 *ql_i2c.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_I2cInit()	初始化 I2C 总线。
ql_I2cWrite()	向 I2C 总线写入数据。从设备的寄存器地址长度为 8 位。
ql_I2cRead()	从 I2C 总线读取数据。从设备的寄存器地址长度为 8 位。
ql_I2cRelease()	释放 I2C 总线。
ql_I2cWrite_16bit_addr()	向 I2C 总线写入数据。从设备的寄存器地址长度为 16 位。
ql_I2cRead_16bit_addr()	从 I2C 总线读取数据。从设备的寄存器地址长度为 16 位。

2.3. 函数详解

2.3.1. ql_I2cInit

该函数用于初始化 I2C 总线。

- **函数原型**

```
ql_errcode_i2c_e ql_I2cInit(ql_i2c_channel_e i2c_no, ql_i2c_mode_e Mode)
```

- **参数**

i2c_no：
[In] I2C 总线编号。详情请参考***第 2.3.1.1 章***。

Mode：
[In] I2C 的工作模式。详情请参考***第 2.3.1.2 章。***

- **返回值**

详情请参考***第 2.3.1.3 章***。

2.3.1.1. ql_i2c_channel_e

I2C 总线编号枚举定义如下：

```
typedef enum
{
    i2c_1 = 0,
    i2c_2 = 1,
}ql_i2c_channel_e;
```

- **成员**

成员	描述
i2c_1	I2C 总线编号为 1
i2c_2	I2C 总线编号为 2

备注

在 QuecOpen 方案下，EC600U 系列模块提供一个 I2C 接口，总线编号只能为 1；EC200U 系列、EG500U-CN 和 EG700U-CN 模块提供两个 I2C 接口，总线编号可为 1 或 2。

2.3.1.2. ql_i2c_mode_e

I2C 工作模式枚举定义如下：

```
typedef enum
{
    STANDARD_MODE = 0,
    FAST_MODE = 1,
} ql_i2c_mode_e;
```

- **成员**

成员	描述
STANDARD_MODE	标准模式，传输速度为 100 kbps
FAST_MODE	快速模式，传输速度为 400 kbps

2.3.1.3. ql_errcode_i2c_e

I2C 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_I2C_SUCCESS = QL_SUCCESS,

    QL_I2C_INIT_ERR                 = 1|QL_I2C_ERRCODE_BASE,
    QL_I2C_NOT_INIT_ERR,
    QL_I2C_INVALID_PARAM_ERR,
    QL_I2C_WRITE_ERR                = 5|QL_I2C_ERRCODE_BASE,
    QL_I2C_READ_ERR,
    QL_I2C_RELEASE_ERR,
}ql_errcode_i2c_e;
```

- **成员**

成员	描述
QL_I2C_SUCCESS	函数执行成功
QL_I2C_INIT_ERR	I2C 初始化失败
QL_I2C_NOT_INIT_ERR	I2C 未初始化
QL_I2C_INVALID_PARAM_ERR	参数无效
QL_I2C_WRITE_ERR	向 I2C 写入数据失败
QL_I2C_READ_ERR	从 I2C 读取数据失败
QL_I2C_RELEASE_ERR	释放 I2C 总线失败

2.3.2. ql_I2cWrite

该函数用于向 I2C 总线写入数据，从设备的寄存器地址长度为 8 位。

- **函数原型**

```
ql_errcode_i2c_e ql_I2cWrite(ql_i2c_channel_e i2c_no, uint8_t slave, uint8_t addr, uint8_t *data, uint32_t length)
```

- **参数**

i2c_no：
[In] I2C 总线编号。详情请参考***第 2.3.1.1 章***。

slave：
[In] I2C 从设备地址。

addr：
[In] I2C 从设备的寄存器地址。

data：
[In] 写入的数据。

length：
[In] 写入数据的长度。单位：字节。

- **返回值**

详情请参考***第 2.3.1.3 章***。

2.3.3. ql_I2cRead

该函数用于从 I2C 总线读取数据，从设备的寄存器地址长度为 8 位。

- **函数原型**

```
ql_errcode_i2c_e ql_I2cRead(ql_i2c_channel_e i2c_no, uint8_t slave, uint8_t addr, uint8_t *buf, uint32_t length)
```

- **参数**

i2c_no：
[In] I2C 总线编号。详情请参考***第 2.3.1.1 章***。

slave：
[In] I2C 从设备地址。

addr：
[In] I2C 从设备的寄存器地址。

buf：
[Out] 读取的数据。

length：
[In] 读取数据的长度。单位：字节。

- **返回值**

详情请参考***第 2.3.1.3 章***。

2.3.4. ql_I2cRelease

该函数用于释放 I2C 总线。若需要重新初始化同一个 I2C 总线，须调用该函数释放 I2C 总线，再重新调用 *ql_I2cInit()*进行初始化。

- **函数原型**

```
ql_errcode_i2c_e ql_I2cRelease(ql_i2c_channel_e i2c_no)
```

- **参数**

i2c_no：
[In] I2C 总线编号。详情请参考***第 2.3.1.1 章***。

- **返回值**

详情请参考*第 2.3.1.3 章*。

2.3.5. ql_I2cWrite_16bit_addr

该函数用于向 I2C 总线写入数据，从设备的寄存器地址长度为 16 位。

- **函数原型**

```
ql_errcode_i2c_e ql_I2cWrite_16bit_addr(ql_i2c_channel_e i2c_no, uint8_t slave, uint16_t addr, uint8_t *data, uint32_t length)
```

- **参数**

i2c_no：
[In] I2C 总线编号。请参考*第 2.3.1.1 章*。

slave：
[In] I2C 从设备地址。

addr：
[In] I2C 从设备的寄存器地址。

data：
[In] 写入的数据。

length：
[In] 写入数据的长度。单位：字节。

- **返回值**

详情请参考*第 2.3.1.3 章*。

2.3.6. ql_I2cRead_16bit_addr

该函数用于从 I2C 总线读取数据，从设备的寄存器地址长度为 16 位。

- **函数原型**

```
ql_errcode_i2c_e ql_I2cRead_16bit_addr(ql_i2c_channel_e i2c_no, uint8_t slave, uint16_t addr, uint8_t *buf, uint32_t length)
```

- **参数**

i2c_no：
[In] I2C 总线编号。详情请参考***第 2.3.1.1 章***。

slave：
[In] I2C 从设备地址。

addr：
[In] I2C 从设备的寄存器地址。

buf：
[Out] 读取的数据。

length：
[In] 读取数据的长度。单位：字节。

- **返回值**

详情请参考***第 2.3.1.3 章***。

3 I2C 开发示例

本章节主要介绍如何在 App 侧使用上述的 API 进行 I2C 开发以及简单的调试。

3.1. App 侧的 I2C 开发示例

ECx00U 系列和 EGx00U QuecOpen 模块的 SDK 代码中提供了 I2C 的示例文件 *I2C_demo.c*，文件路径为：*components\ql-application\i2c*。由于 I2C 需要使用从设备进行通讯，示例中默认使用型号为 GC032A 的摄像头进行功能演示。文件中的相关函数解析如下：

- *ql_i2c_demo_init()*：创建一个任务，运行 I2C 示例前需调用该函数；
- *ql_i2c_demo_thread()*：任务的执行函数，用于实现 I2C 的初始化及读写功能。

若需要运行该示例，只需在 *ql_init_demo_thread()*线程中调用 *ql_i2c_demo_init()*，并且在模块上连接 GC032A 摄像头，如下图所示：

```
} « end ql_pin_cfg_init »

static void ql_init_demo_thread(void *param)
{
    QL_INIT_LOG("init demo thread enter, param 0x%x", param);
    /*Caution: GPIO pin must be initialized here, otherwise the pin status cannot be determined*/
    ql_pin_cfg_init();

#if 0
    ql_gpio_app_init();
    ql_gpioint_app_init();
#endif

#ifdef QL_APP_FEATURE_I2C
    /*因为i2c通过camera来演示功能，所以打开i2c demo时不可同时打开camera demo*/
    /*because I2C uses the camera as a demonstration, so i2c demo can not be opened when the camera demo open */
    ql_i2c_demo_init();
#endif


#ifdef QL_APP_FEATURE_LEDCFG
    //ql_ledcfg_app_init();
#endif

#ifdef QL_APP_FEATURE_AUDIO
    //ql_audio_app_init();
#endif

#ifdef QL_APP_FEATURE_TTS
    //ql_tts_task_init();
```

图 1：创建 I2C 任务并运行 I2C 示例

I2C 从设备寄存器地址长度默认为 1 个字节（8 位），按照示例，将宏 **demo_for_8bit_or_16bit** 选择为 1，则保持默认长度 1 个字节；若使用寄存器地址长度为 2 个字节（16 位）的 I2C 从设备，则需要将宏选择为 0。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "ql_api_osi.h"
#include "ql_log.h"
#include "ql_api_camera.h"
#include "ql_i2c.h"
#include "I2C_demo.h"

/*===========================================================================
 * Macro Definition
 ===========================================================================*/

#define QL_APP_I2C_LOG_LEVEL            QL_LOG_LEVEL_INFO
#define QL_APP_I2C_LOG(msg, ...)        QL_LOG(QL_APP_I2C_LOG_LEVEL, "QL_APP_I2C", msg, ##__VA_ARGS__)
#define QL_APP_I2C_LOG_PUSH(msg, ...)   QL_LOG_PUSH("QL_APP_I2C", msg, ##__VA_ARGS__)

#define QL_I2C_TASK_STACK_SIZE      1024
#define QL_I2C_TASK_PRIO            APP_PRIORITY_NORMAL
#define QL_I2C_TASK_EVENT_CNT       5

#define SalveAddr_w_8bit        (0x42 >> 1)
#define SalveAddr_r_8bit        (0x43 >> 1)

/*reserved*/
#define SalveAddr_w_16bit       (0xff >> 1)
#define SalveAddr_r_16bit       (0xff >> 1)

#define demo_for_8bit_or_16bit  (1)       //1:test 8bit register address     0:test 16bit register address
/*===========================================================================
 * Struct
 ===========================================================================*/

/*===========================================================================
 * Enum
 ===========================================================================*/
```

图 2：I2C 从设备寄存器位数选择

3.2. I2C 功能调试

ECx00U 系列和 EGx00U QuecOpen 模块可通过使用移远通信 LTE OPEN EVB 进行 I2C 功能调试。LTE OPEN EVB 详情，请参考*文档 [2]*。按照*第 3.1 章*运行 I2C 示例，如下图所示：

```
74:
75: void ql_i2c_demo_thread(void *param)
76: {
77: #if demo_for_8bit_or_16bit
78:     /*test 8bit register address*/
79:     uint8_t read_data = 0;
80:     uint8_t data = 0xaa;
81:
82:     /*operate the camera for the example*/
83:     ql_CamInit(320, 240);
84:     ql_CamPowerOn();
85:
86:     ql_I2cInit(i2c_1, STANDARD_MODE);
87:     while(1)
88:     {
89:         QL_APP_I2C_LOG("I2C read_data = 0x%x", read_data);
90:         ql_I2cRead(i2c_1, SalveAddr_r_8bit, 0xf0, &read_data, 1);
91:         QL_APP_I2C_LOG("I2C read_data = 0x%x", read_data);
92:         ql_I2cWrite(i2c_1, SalveAddr_w_8bit, 0x55, &data, 1);
93:         read_data = 0;
94:         ql_rtos_task_sleep_ms(200);
95:     }
96:
97: #else
98:
99:     /*test 16bit register address*/
00:     uint8_t read_data = 0;
01:     uint8_t data = 0xff;
02:
```

图 3：运行 I2C 示例

先将编译版本烧录到模块中，再使用 USB 转串口线将 LTE OPEN EVB 的 USB 端口与 PC 连接，并连接 coolwatcher 工具抓取 AP Log，如下图所示：

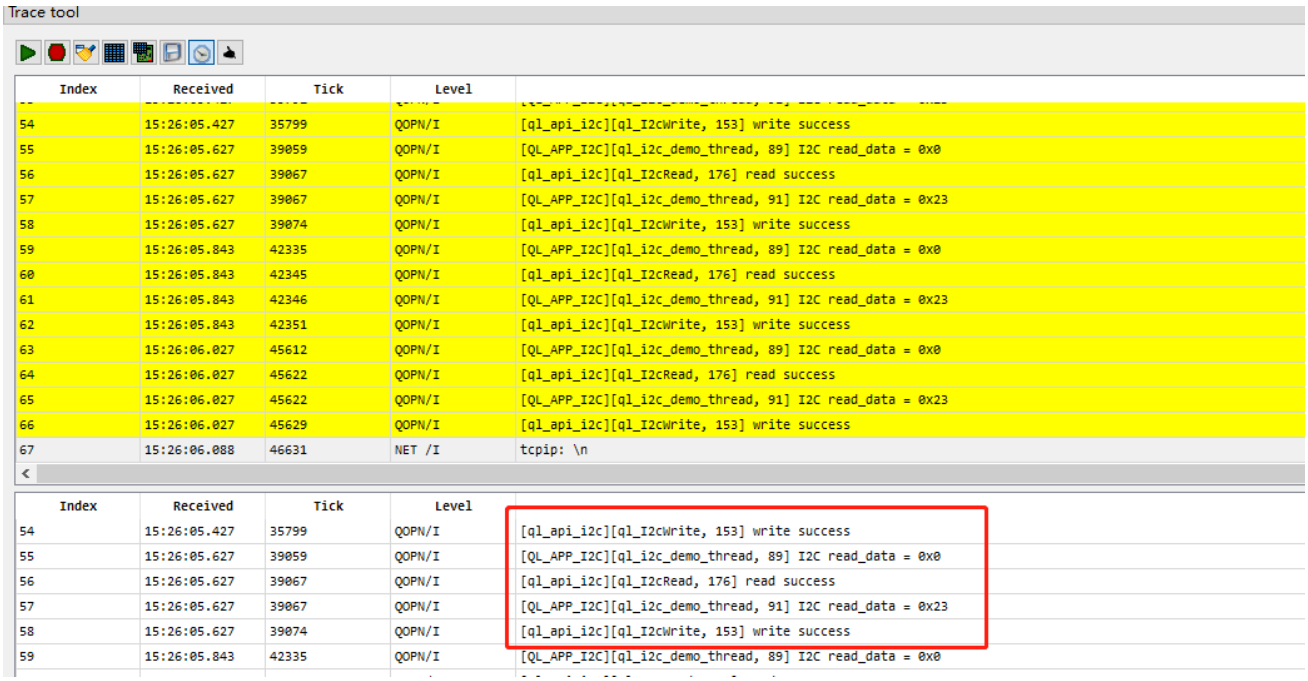

图 4：coolwatcher 抓取 Log

如上图所示，对比读写前后的两组数据，*ql_I2cWrite()*返回“**write success**”，*ql_I2cRead()*返回实际从 I2C 从设备读取的数据，表明 I2C 通讯成功。

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
[2] Quectel_LTE_OPEN_EVB_User_Guide

表 4：术语缩写

缩写	英文全称	中文全称
AP	Access Point	接入点
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
I2C	Inter-Integrated Circuit	内置集成电路
RTOS	Real-Time Operating System	实时操作系统
SDK	Software Development Kit	软件开发工具包