ECx00U&EGx00U 系列

QuecOpen SMS API

参考手册

LTE Standard 模块系列

版本：1.0

日期：2021-09-02

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	Marvin NING	文档创建
1.0	2021-09-02	Marvin NING	受控版本

目录

表格索引

1 引言

移远通信 LTE Standard ECx00U 系列、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、EGx00U QuecOpen®模块 SDK 中提供的 SMS API 使用方法和功能。SMS 接口主要包括短信发送、接收短信、获取短信中心号码、获取短信储存状态等功能。

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 SMS 数据结构及 API 介绍

2.1. 头文件

SMS API 的头文件为 *ql_api_sms.h*，位于 SDK 包的 *ql-sdk\components\ql-kernel\inc* 目录下。若无特别说明，本文档所提到的头文件均在该目录下。

2.2. 函数概览

表 2：函数概览

函数	说明
ql_sms_get_init_status()	获取 SMS 初始化状态
ql_sms_send_msg()	发送文本短信
ql_sms_send_pdu()	发送 PDU 短信
ql_sms_read_msg()	读取文本或 PDU 类型的短信
ql_sms_read_msg_list()	读取短信列表
ql_sms_delete_msg()	删除短信
ql_sms_get_center_address()	获取短信中心号码
ql_sms_set_center_address()	设置短信中心号码
ql_sms_set_storage()	设置短信存储位置
ql_sms_get_storage()	获取短信存储位置
ql_sms_get_storage_info()	获取(U)SIM 卡与移动设备中的短信存储状态
ql_sms_callback_register()	设置短信回调函数

2.3. SMS 相关 API

2.3.1. ql_sms_get_init_status

该函数用于获取 SMS 初始化状态。只有初始化完成，才能调用 SMS 相关函数。

- **函数原型**

```
ql_sms_errcode_e ql_sms_get_init_status(uint8_t nSim, uint8_t *status)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

status：
[Out] SMS 初始化状态。
0 未完成初始化
1 初始化完成

- **返回值**

详见*第 2.3.1.1 章*。

2.3.1.1. ql_sms_errcode_e

SMS 相关 API 执行错误码枚举定义如下：

```
typedef enum
 {
   QL_SMS_SUCCESS                    = 0,
   QL_SMS_ERROR                      = 1 | (QL_COMPONENT_SMS << 16),
   QL_SMS_NOT_INIT_ERR,
   QL_SMS_PARA_ERR,
   QL_SMS_NO_MEMORY_ERR,
   QL_SMS_SEM_CREATE_ERR,
   QL_SMS_SEM_TIMEOUT_ERR,
   QL_SMS_NO_MSG_ERR,
 }ql_sms_errcode_e
```

- **参数**

参数	描述
QL_SMS_SUCCESS	函数执行成功
QL_SMS_ERROR	函数执行失败
QL_SMS_NOT_INIT_ERR	SMS 业务未初始化
QL_SMS_PARA_ERR	参数错误
QL_SMS_NO_MEMORY_ERR	内存申请失败
QL_SMS_SEM_CREATE_ERR	信号量创建失败
QL_SMS_SEM_TIMEOUT_ERR	信号量等待超时
QL_SMS_NO_MSG_ERR	无可读取短信

2.3.2. ql_sms_send_msg

该函数用于发送文本短信。

- **函数原型**

```
ql_sms_errcode_e ql_sms_send_msg(uint8_t nSim, char *phone_num, char *data, ql_sms_code_e code)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

phone_num：
[In] 接收方手机号码。

data：
[In] 消息内容。小于 140 个字节。

code：
[In] 短信编码方式。详见***第 2.3.2.1 章***。
0 GSM-7 bit
1 UCS2

- **返回值**

详见*第 2.3.1.1 章*。

2.3.2.1. ql_sms_code_e

短信编码方式定义如下：

```
typedef enum
{
  GSM = 0,
  UCS2 = 1,
}ql_sms_code_e
```

- **参数**

参数	描述
GSM	GSM 编码
UCS2	UCS2 编码，发送的短信包含中文时使用

2.3.3. ql_sms_send_pdu

该函数用于发送 PDU 类型的短信。

- **函数原型**

```
ql_sms_errcode_e ql_sms_send_pdu(uint8_t nSim, char *pdu)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

pdu：
[In] PDU 短信。

- **返回值**

详见*第 2.3.1.1 章*。

2.3.4. ql_sms_read_msg

该函数用于读取文本或 PDU 类型的短信。

- **函数原型**

```
ql_sms_errcode_e ql_sms_read_msg(uint8_t nSim, uint8_t index, char *buf, uint16_t buf_len, ql_sms_format_e format)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

index：
[In] 短信索引号。

buf：
[Out] 用于接收短信的数组指针。

buf_len：
[In] 数组长度。

format：
[In] 短信格式。文本或 PDU 格式。详见***第 2.3.4.1 章***。

- **返回值**

详见***第 2.3.1.1 章***。

2.3.4.1. ql_sms_format_e

短信格式定义如下：

```
typedef enum
{
  PDU = 0,
  TEXT = 1,
}ql_sms_format_e
```

- **参数**

参数	描述
PDU	PDU 格式
TEXT	文本格式

2.3.5. ql_sms_read_msg_list

该函数用于读取短信列表。

- **函数原型**

```
ql_sms_errcode_e ql_sms_read_msg_list(uint8_t nSim, ql_sms_format_e format)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

format：
[In] 短信格式。详见***第 2.3.4.1 章***。

- **返回值**

详见***第 2.3.1.1 章***。

2.3.6. ql_sms_delete_msg

该函数用于删除短信。

- **函数原型**

```
ql_sms_errcode_e ql_sms_delete_msg(uint8_t nSim, uint8_t index)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

index：
[In] 需删除短信的索引号。

- **返回值**

详见*第 2.3.1.1 章*。

2.3.7. ql_sms_get_center_address

该函数用于获取短信中心号码。

- **函数原型**

```
ql_sms_errcode_e ql_sms_get_center_address(uint8_t nSim, char* address, uint8_t len)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

address：
[Out] 用于接收短信中心号码的数组指针。

len：
[In] 数组长度。

- **返回值**

详见*第 2.3.1.1 章*。

2.3.8. ql_sms_set_center_address

该函数用于设置短信中心号码。若无特殊需求，不建议更改短信中心号码。

- **函数原型**

```
ql_sms_errcode_e ql_sms_set_center_address(uint8_t nSim, char* address)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

address：
[In] 字符串格式的短信中心号码，需添加“+86”的前缀。

- **返回值**

详见***第 2.3.1.1 章***。

2.3.9. ql_sms_set_storage

该函数用于设置短信存储位置。

- **函数原型**

```
ql_sms_errcode_e ql_sms_set_storage(uint8_t nSim, ql_sms_stor_e mem1, ql_sms_stor_e mem2, ql_sms_stor_e mem3)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

mem1：
[In] 已读取和删除消息的存储位置。详见***第 2.3.9.1 章***。

mem2：
[In] 待写入和发送消息的存储位置。详见***第 2.3.9.1 章***。

mem3：
[In] 已接收消息的首选存储位置。详见***第 2.3.9.1 章***。

- **返回值**

详见***第 2.3.1.1 章***。

2.3.9.1. ql_sms_stor_e

短信存储位置枚举定义如下：

```
typedef enum
{
  ME = 1,
  SM = 2,
}ql_sms_stor_e
```

- **参数**

参数	描述
ME	移动设备
SM	(U)SIM 卡

2.3.10. ql_sms_get_storage

该函数用于获取短信存储位置。

- **函数原型**

```
ql_sms_errcode_e ql_sms_get_storage(uint8_t nSim, ql_sms_mem_info_t *mem_info)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

mem_info：
[Out] 短信存储位置。详见***第 2.3.10.1 章***。

- **返回值**

详见***第 2.3.1.1 章***。

2.3.10.1. ql_sms_mem_info_t

短信存储位置结构体定义如下：

```
typedef struct
 {
   ql_sms_stor_e mem1;
   ql_sms_stor_e mem2;
   ql_sms_stor_e mem3;
 }ql_sms_mem_info_t
```

- **参数**

类型	参数	描述
ql_sms_stor_e	*mem1*	已读取和删除消息的存储位置，详见***第 2.3.9.1 章***
ql_sms_stor_e	*mem2*	待写入和发送消息的存储位置，详见***第 2.3.9.1 章***
ql_sms_stor_e	*mem3*	已接收消息的首选存储位置，详见***第 2.3.9.1 章***

2.3.11. ql_sms_get_storage_info

该函数用于获取(U)SIM 卡与移动设备中短信存储状态。

- **函数原型**

```
ql_sms_errcode_e ql_sms_get_storage_info(uint8_t nSim, ql_sms_stor_info_s *stor_info)
```

- **参数**

nSim：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

stor_info：
[Out] 短信存储状态的结构体。详见***第 2.3.11.1 章***。

- **返回值**

详见***第 2.3.1.1 章***。

2.3.11.1. ql_sms_stor_info_s

短信存储状态的结构体定义如下：

```
typedef struct
{
    uint16_t usedSlotSM;
    uint16_t totalSlotSM;
    uint16_t unReadRecordsSM;
    uint16_t usedSlotME;
    uint16_t totalSlotME;
    uint16_t unReadRecordsME;
    ql_sms_stor_e    newSmsStorId;
 }ql_sms_stor_info_s
```

- **参数**

类型	参数	描述
uint16_t	*usedSlotSM*	(U)SIM 卡已存储的短信数量（已使用的存储空间）
uint16_t	*totalSlotSM*	(U)SIM 卡可存储短信总数量
uint16_t	*unReadRecordsSM*	(U)SIM 卡未读短信数量
uint16_t	*usedSlotME*	移动设备已存储的短信数量（已使用的存储空间）
uint16_t	*totalSlotME*	移动设备可存储短信总数量
uint16_t	*unReadRecordsME*	移动设备未读短信数量
ql_sms_stor_e	*newSmsStorId*	新消息存储位置，详见***第 2.3.9.1 章***

2.3.12. ql_sms_callback_register

该函数用于注册短信回调函数。

- **函数原型**

```
void ql_sms_callback_register(ql_sms_event_handler_t cb)
```

- **参数**

cb：
[In] 回调函数。详见***第 2.3.12.1 章***。

- **返回值**

无

2.3.12.1. ql_sms_event_handler_t

该函数定义短信相关事件回调函数。

- **函数原型**

```
typedef void (*ql_sms_event_handler_t)(uint8_t sim_id, int event_id, void *ctx)
```

- **参数**

sim_id：
[In] 所用的(U)SIM 卡。若模块只支持 1 个(U)SIM 接口，此参数只可设置为 0。
0 (U)SIM 卡 1
1 (U)SIM 卡 2

event_id：
[In] 短消息事件。详见***第 2.3.12.2 章***。

ctx：
[In] 回调函数的传参指针。

- **返回值**

无

2.3.12.2. ql_sms_event_id_e

短信相关事件枚举定义如下：

```
typedef enum
{
  QL_SMS_INIT_OK_IND =    1| (QL_COMPONENT_SMS << 16),
  QL_SMS_NEW_MSG_IND,
  QL_SMS_LIST_IND,
  QL_SMS_LIST_END_IND,
  QL_SMS_MEM_FULL_IND,
}ql_sms_event_id_e
```

- 参数

参数	描述
QL_SMS_INIT_OK_IND	SMS 初始化成功
QL_SMS_NEW_MSG_IND	新短消息
QL_SMS_LIST_IND	短消息列表
QL_SMS_LIST_END_IND	短消息列表已读取
QL_SMS_MEM_FULL_IND	短信存储空间已满

3 SMS 开发示例

本章节主要介绍在 APP 侧如何使用上述 API 进行 SMS 功能开发。

3.1. Demo 说明

ECx00U 系列、EGx00U QuecOpen SDK 代码中提供了 SMS 的示例，即 *sms_demo.c* 文件，位于 *components\ql-application\sms* 目录下。该文件中主要包含发送短信、读取短信、删除短信等功能。以下示例展示了如何使用 API 发送英文与中文短信：

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

#include "ql_api_common.h"
#include "ql_api_osi.h"
#include "ql_api_sms.h"
#include "ql_log.h"

ql_task_t sms_task = NULL;
ql_sem_t    sms_init_sem = NULL;

void user_sms_event_callback(uint8_t nSim, int event_id, void *ctx)
{
    switch(event_id)
    {
        case QL_SMS_INIT_OK_IND:
        {
            QL_SMS_LOG("QL_SMS_INIT_OK_IND");
            ql_rtos_semaphore_release(sms_init_sem);
            break;
        }
        case QL_SMS_NEW_MSG_IND:
        {
            ql_sms_new_s *msg = (ql_sms_new_s *)ctx;
            QL_SMS_LOG("sim=%d, index=%d, storage memory=%d", nSim, msg->index,
msg->mem);
            break;
```

```
            }
            case QL_SMS_MEM_FULL_IND:
            {
                ql_sms_new_s *msg = (ql_sms_new_s *)ctx;
                QL_SMS_LOG("QL_SMS_MEM_FULL_IND sim=%d, memory=%d",nSim,msg->mem);
                break;
            }
            default :
                break;
        }
}


void sms_demo_task(void * param)
{
    uint8_t nSim = 0;
    QL_SMS_LOG("enter");
    ql_sms_callback_register(user_sms_event_callback);

    //wait sms ok
    if(ql_rtos_semaphore_wait(sms_init_sem, QL_WAIT_FOREVER)){
        QL_SMS_LOG("Waiting for SMS init timeout");
    }

    //Send English text message
    if(QL_SMS_SUCCESS == ql_sms_send_msg(nSim,"+86189xxxxxxxx","Hello,marvin", GSM)){
        QL_SMS_LOG("send sms OK");
    }else{
        QL_SMS_LOG("send sms FAIL");
    }
    //Send messages in Chinese and English. (Need use UTF8 encoding to open sms_demo.c for chinese.)
    if(QL_SMS_SUCCESS == ql_sms_send_msg(nSim,"+86189xxxxxxxx","hello,你好", UCS2)){
        QL_SMS_LOG("send pdu sms OK");
    }else{
        QL_SMS_LOG("send pdu sms FAIL");
    }

    ql_rtos_task_delete(NULL);
}


QIOSStatus ql_sms_app_init(void)
{
```

```
    QIOSStatus err = QL_OSI_SUCCESS;

    err = ql_rtos_task_create(&sms_task, 4096, APP_PRIORITY_NORMAL, "SMS_TASK",
sms_demo_task, NULL, 2);
    if(err != QL_OSI_SUCCESS)
    {
        QL_SMS_LOG("sms_task created failed, ret = 0x%x", err);
    }

    err = ql_rtos_semaphore_create(&sms_init_sem, 0);
    if(err != QL_OSI_SUCCESS)
    {
        QL_SMS_LOG("sms_init_sem created failed, ret = 0x%x", err);
    }

    return err;
}
```

3.2. Demo 自启动

上述 Demo 已在 *ql_init_demo_thread* 线程中默认不启动，如下图所示，需要测试时请取消注释。另外还需要将 demo 中的 *ql_sms_send_msg()*接口的目的手机号修改成用户自己的手机号。

```
#ifdef QL_APP_FEATURE_SMS
    //ql_sms_app_init();
#endif
#ifdef QL_APP_FEATURE_VOICE_CALL
    //ql_voice_call_app_init();
#endif
#ifdef QL_APP_FEATURE_VOLTE
    //ql_volte_app_init();
#endif
```

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
GSM	Global System for Mobile Communications	全球移动通信系统
IoT	Internet of Things	物联网
LTE	Long-Term Evolution	长期演进
ME	Terminal Equipment	终端设备
PDU	Packet Data Unit	分组数据单元
RTOS	Real-Time Operating System	实时操作系统
SDK	Software Development Kit	软件开发工具包
(U)SIM	(Universal) Subscriber Identity Module	（全球）用户识别模块
SMS	Short Message Service	短消息业务
UCS	Universal Character Set	通用字符集
UTF	Unicode Transformation Format	Unicode 转换格式