ECx00U&EGx00U 系列
QuecOpen SDMMC API 参考手册

LTE Standard 模块系列

版本：1.0

日期：2021-09-14

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
-	2021-03-25	Sum LI	文档创建
1.0	2021-09-14	Sum LI	受控版本

目录

文档历史 3
目录 4
表格索引 5
图片索引 6

1 引言 7
1.1. 适用模块 7

2 SDMMC API 8
2.1. 头文件 8
2.2. 函数概览 8
2.3. SDMMC 相关 API 9
2.3.1. ql_sdmmc_mount 9
2.3.2. ql_sdmmc_umount 11
2.3.3. ql_sdmmc_mkfs 12
2.3.4. ql_sdmmc_is_mount 12
2.3.5. ql_sdmmc_is_format 13
2.3.6. ql_sdmmc_set_clk 13
2.3.7. ql_sdmmc_write 13
2.3.8. ql_sdmmc_read 14
2.3.9. ql_sdmmc_get_block_number 15
2.3.10. ql_sdmmc_open 15
2.3.11. ql_sdmmc_close 15
2.3.12. ql_sdmmc_get_hw_info 16
2.3.13. ql_sdmmc_sdio1_vdd_off 17
2.3.14. ql_sdmmc_fdisk_ex 17
2.3.15. ql_sdmmc_mkfs_ex 19
2.3.16. ql_sdmmc_mount_ex 19
2.3.17. ql_sdmmc_umount_ex 20
2.3.18. ql_sdmmc_is_fdisk_ex 20
2.3.19. ql_sdmmc_is_format_ex 21
2.3.20. ql_sdmmc_is_mount_ex 21

3 SDMMC 开发示例及调试 23
3.1. SDMMC 开发示例 23
3.1.1. SDMMC Demo 说明 23
3.1.2. SDMMC Demo 自启动 25
3.2. SDMMC 相关调试 26
3.2.1. 调试准备 26
3.2.2. SDMMC 功能调试 27

4 附录 28

表格索引

表 1：适用模块 7
表 2：函数概览 8
表 3：参考文档 28
表 4：术语缩写 28

图片索引

图 1: 功能打开/关闭宏定义 23
图 2: 入口函数 ql_sdmmc_app_init() 24
图 3: SDMMC Demo 默认不启动 25
图 4: LTE OPEN EVB 26
图 5: USB 端口显示 27
图 6: AP Log 信息 27

1 引言

移远通信 LTE Standard 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]***。

在 QuecOpen®方案下，ECx00U 系列和 EGx00U 模块支持 SD 卡和 eMMC，本文档主要介绍模块的 SDMMC 应用，包括相关 API、数据结构、开发实例以及调试方法。

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 SDMMC API

2.1. 头文件

ql_sdmmc.h 即为 SDMMC API 的头文件，位于 SDK 包的 *components\ql-kernel\inc* 目录下。

2.2. 函数概览

表 2：函数概览

函数	说明
ql_sdmmc_mount()	初始化 SDMMC 驱动并将其挂载到文件系统上
ql_sdmmc_umount()	反初始化 SDMMC 驱动并将其从文件系统上卸载
ql_sdmmc_mkfs()	格式化 SDMMC 设备
ql_sdmmc_set_clk()	设置 SDMMC 的时钟频率
ql_sdmmc_write()	在 SDMMC 指定扇区写入指定字节大小的数据
ql_sdmmc_read()	从 SDMMC 指定扇区读取指定字节大小的数据
ql_sdmmc_get_block_number()	获取 SDMMC 总扇区数量
ql_sdmmc_open()	初始化 SDMMC 驱动，不挂载文件系统
ql_sdmmc_close()	反初始化 SDMMC 驱动
ql_sdmmc_get_hw_info()	获取 SDMMC 的硬件信息
ql_sdmmc_sdio1_vdd_off()	关闭 SDMMC 通信引脚的电源
ql_sdmmc_fdisk_ex()	对 SDMMC 设备进行分区

ql_sdmmc_mkfs_ex()	格式化 SDMMC 设备的指定分区
ql_sdmmc_mount_ex()	将指定分区初始化并挂载到文件系统上
ql_sdmmc_umount_ex()	将指定分区反初始化并从文件系统上卸载
ql_sdmmc_is_fdisk_ex()	判断 SDMMC 设备是否已进行分区
ql_sdmmc_is_format_ex()	判断分区是否已格式化
ql_sdmmc_is_mount_ex()	判断分区是否已挂载

2.3. SDMMC 相关 API

2.3.1. ql_sdmmc_mount

该函数用于初始化 SDMMC 驱动并将其挂载到文件系统上，仅适用于通过文件系统操作 SDMMC 的场景。

当 SDMMC 设备未进行分区时，则调用该函数来进行初始化并挂载到文件系统。当 SDMMC 设备已进行分区时，则建议调用 *ql_sdmmc_mount_ex()*；此时若调用 *ql_sdmmc_mount()*，则默认是将第一分区进行初始化并挂载到文件系统上。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_mount(void)
```

- **参数**

无

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.1.1. ql_errcode_sdmmc_e

SDMMC 的 API 错误码，枚举信息定义如下：

```
typedef enum
{
    QL_SDMMC_SUCCESS                    = QL_SUCCESS,
    QL_SDMMC_INVALID_PARAM_ERR          = 1|QL_SDMMC_ERRCODE_BASE,
```

```
    QL_SDMMC_ADDR_NULL_ERR              = 3|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_INIT_ERR                   = 4|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MOUNT_ERR                  = 5|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MKFS_ERR                   = 10|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_SET_CLK_ERR                = 11|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_WRITE_ERR                  = 12|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_READ_ERR                   = 13|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_DET_ERR                    = 14|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MEM_ILLEGAL_ERR            = 20|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_ADDR_ALIGNED_ERR           = 21|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_SIZE_ALIGNED_ERR           = 22|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_BLOCK_NUM_OVER_ERR         = 23|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_CREATE_ERR                 = 30|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_OPEN_ERR                   = 31|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_CLOSE_ERR                  = 32|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_OPERATION_NOT_ALLOWED      = 33|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MULTI_PARTITON_NUM_ERR     = 40|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MULTI_PARTITON_FDISK_ERR   = 41|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_UMOUNT_ERR                 = 42|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MOUNT_ALL_ERR              = 43|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_UMOUNT_ALL_ERR             = 44|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MKFS_ALL_ERR               = 45|QL_SDMMC_ERRCODE_BASE,
    QL_SDMMC_MULTI_PARTITON_OTHER_ERR   = 46|QL_SDMMC_ERRCODE_BASE,
} ql_errcode_sdmmc_e
```

- **参数**

参数	描述
QL_SDMMC_SUCCESS	SDMMC 操作成功
QL_SDMMC_INVALID_PARAM_ERR	API 参数错误
QL_SDMMC_ADDR_NULL_ERR	指针地址为空
QL_SDMMC_INIT_ERR	初始化 SDMMC 驱动以及挂载文件系统失败
QL_SDMMC_MOUNT_ERR	SDMMC 挂载失败
QL_SDMMC_MKFS_ERR	SDMMC 格式化失败
QL_SDMMC_SET_CLK_ERR	SDMMC 时钟频率设置失败
QL_SDMMC_WRITE_ERR	SDMMC 写操作失败
QL_SDMMC_READ_ERR	SDMMC 读操作失败

QL_SDMMC_DET_ERR	热插拔中断注册失败
QL_SDMMC_MEM_ILLEGAL_ERR	非法内存地址
QL_SDMMC_ADDR_ALIGNED_ERR	内存地址 32 字节对齐错误
QL_SDMMC_SIZE_ALIGNED_ERR	读写数据 512 字节对齐错误
QL_SDMMC_BLOCK_NUM_OVER_ERR	读写的扇区号超出范围
QL_SDMMC_CREATE_ERR	SDMMC 句柄创建失败
QL_SDMMC_OPEN_ERR	初始化 SDMMC 驱动失败
QL_SDMMC_CLOSE_ERR	反初始化 SDMMC 驱动失败
QL_SDMMC_OPERATION_NOT_ALLOWED	非法操作
QL_SDMMC_MULTI_PARTITON_NUM_ERR	多分区的分区号参数错误
QL_SDMMC_MULTI_PARTITON_FDISK_ERR	分区操作失败
QL_SDMMC_UMOUNT_ERR	卸载操作失败
QL_SDMMC_MOUNT_ALL_ERR	挂载所有分区操作失败
QL_SDMMC_UMOUNT_ALL_ERR	卸载所有分区操作失败
QL_SDMMC_MKFS_ALL_ERR	格式化所有分区操作失败
QL_SDMMC_MULTI_PARTITON_OTHER_ERR	多分区其他操作失败

2.3.2. ql_sdmmc_umount

该函数用于反初始化 SDMMC 驱动并将其从文件系统上卸载，与 *ql_sdmmc_mount()*配套使用；仅适用于通过文件系统操作 SDMMC 的场景。

当 SDMMC 设备未进行分区时，调用该函数则反初始化 SDMMC 设备并从文件系统上卸载。当 SDMMC 设备已进行分区时，则建议调用 *ql_sdmmc_umount_ex()*；此时若调用 *ql_sdmmc_umount()*，则默认是将第一分区进行反初始化并从文件系统上卸载。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_umount(void)
```

- **参数**

无

- **返回值**

请参考*第 2.3.1.1 章*。

2.3.3. ql_sdmmc_mkfs

该函数用于格式化 SDMMC 设备，格式化成功后用文件系统再读 SDMMC 设备时将显示为空；仅适用于通过文件系统操作 SDMMC 的场景。

若 SDMMC 设备已进行分区，调用该函数也可实现格式化，但会使所有分区合并，即 SDMMC 设备恢复到未分区的状态，因此建议调用 *ql_sdmmc_mkfs_ex()*来进行指定分区的格式化。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_mkfs(uint8_t opt)
```

- **参数**

opt：
[In] SDMMC 格式化的格式；目前只支持 FM_FAT32 格式。

- **返回值**

请参考*第 2.3.1.1 章*。

2.3.4. ql_sdmmc_is_mount

该函数用于判断 SDMMC 设备是否已挂载到文件系统。

- **函数原型**

```
bool ql_sdmmc_is_mount(void)
```

- **参数**

无

- **返回值**

TRUE　已挂载到文件系统
FALSE　未挂载到文件系统

2.3.5. ql_sdmmc_is_format

该函数用于判断 SDMMC 设备是否已格式化。

- **函数原型**

```
bool ql_sdmmc_is_format(void)
```

- **参数**

无

- **返回值**

TRUE 已格式化
FALSE 未格式化

2.3.6. ql_sdmmc_set_clk

该函数用于设置 SDMMC 的时钟频率；SDSC 设备默认时钟频率为 25 MHz，SDHC/SDXC/EMMC 设备默认时钟频率为 50 MHz。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_set_clk(uint32_t freq)
```

- **参数**

freq：
[In] 要设置的时钟频率。范围：400000~50000000；单位：Hz。

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.7. ql_sdmmc_write

该函数用于在 SDMMC 指定扇区写入指定字节大小的数据。

仅适用于不通过文件系统而直接通过 SDMMC 设备的驱动层操作 SDMMC 的场景，否则有可能会破坏 SDMMC 设备中记录的分区信息；分区被破坏后，必须先使用 *ql_sdmmc_mkfs()*格式化设备才能实现文件系统的正常挂载。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_write(uint32_t block_number, const void *buffer, uint32_t size)
```

- **参数**

block_number：
[In] 待写入数据的扇区号。

buffer：
[In] 存储待写入数据的内存地址，不可为空指针且必须 32 字节对齐。

size：
[In] 待写入的数据大小，必须一个扇区大小（512 字节）对齐。

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.8. ql_sdmmc_read

该函数用于从 SDMMC 指定扇区读取指定字节大小的数据；仅适用于不通过文件系统而直接通过 SDMMC 设备的驱动层操作 SDMMC 的场景。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_read(uint32_t block_number, void *buffer, uint32_t size)
```

- **参数**

block_number：
[In] 待读取数据的扇区号。

buffer：
[In] 存储待读取数据的内存地址，不可为空指针且必须 32 字节对齐。

size：
[In] 待读取的数据大小，必须一个扇区大小（512 字节）对齐。

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.9. ql_sdmmc_get_block_number

该函数用于获取 SDMMC 总扇区数量。

- **函数原型**

```
uint32_t ql_sdmmc_get_block_number(void)
```

- **参数**

无

- **返回值**

SDMMC 总扇区数量。

2.3.10. ql_sdmmc_open

该函数仅用于初始化 SDMMC 驱动，不挂载文件系统；如果文件系统已挂载，调用此函数后，会反挂载文件系统。

仅适用于不通过文件系统而直接通过 SDMMC 设备的驱动层操作 SDMMC 的场景。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_open(void)
```

- **参数**

无

- **返回值**

请参考*第 2.3.1.1 章*。

2.3.11. ql_sdmmc_close

该函数用于反初始化 SDMMC 驱动，与 *ql_sdmmc_open()*配套使用。如果文件系统已挂载，调用此函数后，会反挂载文件系统。

仅适用于不通过文件系统而直接通过 SDMMC 设备的驱动层操作 SDMMC 的场景。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_close(void)
```

- **参数**

无

- **返回值**

请参考*第 2.3.1.1 章*。

2.3.12. ql_sdmmc_get_hw_info

该函数用于获取 SDMMC 的硬件信息。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_get_hw_info(ql_sdmmc_hw_info_t *info)
```

- **参数**

info：
[Out] 需要获取的数据；详情请参考*第 2.3.12.1 章*。

- **返回值**

请参考*第 2.3.1.1 章*。

2.3.12.1. ql_sdmmc_hw_info_t

SDMMC 的硬件信息，结构体定义如下：

```
typedef struct
{
uint8_t    mid;
uint8_t    pnm[6];
uint8_t    psn[4];
uint32_t blknum;
uint32_t blksize;
} ql_sdmmc_hw_info_t
```

- **参数**

类型	参数	描述
uint8_t	*mid*	厂商 ID
uint8_t	*pnm*	产品名；最大长度 6 字节
uint8_t	*psn*	产品序列号；最大长度 4 字节
uint32_t	*blknum*	设备的扇区数量
uint32_t	*blksize*	设备的一个扇区大小

2.3.13. ql_sdmmc_sdio1_vdd_off

该函数用于关闭 SDMMC 设备的通信引脚的电源。请慎用此函数；如必须使用，需先了解 *ql_sdmmc.h* 中关于使用该函数的注意事项。

- **函数原型**

```
void ql_sdmmc_sdio1_vdd_off(void)
```

- **参数**

无

- **返回值**

无

2.3.14. ql_sdmmc_fdisk_ex

该函数用于对 SDMMC 设备进行分区，目前最多只支持 2 个分区。

当设置第一分区的大小后，SDMMC 设备的剩余容量会自动调整为第二分区的大小。由于文件系统分区至少需包含 1024 个扇区，所以设置的每个分区大小不能低于 1024 个扇区对应的大小（1024 × 512 字节）。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_fdisk_ex(ql_sdmmc_part_info_t part_info[])
```

- **参数**

part_info：
[In] 分区信息；详情请参考***第 2.3.14.1 章***。

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.14.1. ql_sdmmc_part_info_t

SDMMC 的分区信息，结构体定义如下：

```
typedef struct
{
ql_sdmmc_partition_num_e partnum;
int64 partsize;
} ql_sdmmc_part_info_t
```

- **参数**

类型	参数	描述
ql_sdmmc_partition_num_e	*partnum*	分区号；详情请参考***第 2.3.14.2 章***
int64	*partsize*	分区大小；单位：兆字节

2.3.14.2. ql_sdmmc_partition_num_e

SDMMC 的分区号，枚举定义如下：

```
typedef enum
{
    QL_SDMMC_PARTITION_NUM_1     = 1,
    QL_SDMMC_PARTITION_NUM_2     = 2,
    QL_SDMMC_PARTITION_NUM_MAX,
    QL_SDMMC_PARTITION_NUM_ALL = 0xff
} ql_sdmmc_partition_num_e
```

- **参数**

参数	描述
QL_SDMMC_PARTITION_NUM_1	第一个分区
QL_SDMMC_PARTITION_NUM_2	第二个分区
QL_SDMMC_PARTITION_NUM_MAX	最大的分区号标志
QL_SDMMC_PARTITION_NUM_ALL	所有可用分区号

2.3.15. ql_sdmmc_mkfs_ex

该函数用于对指定分区进行格式化，目前最多只支持 2 个分区。

此函数可根据分区的扇区数自动调整格式化类型；若总扇区数小于 66130，格式化为 FM_FAT；若总扇区数大于等于 66130，格式化为 FM_FAT32。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_mkfs_ex(ql_sdmmc_partition_num_e part,uint8_t opt)
```

- **参数**

part：
[In] 分区号；
QL_SDMMC_PARTITION_NUM_1：第一分区；
QL_SDMMC_PARTITION_NUM_2：第二分区；
QL_SDMMC_PARTITION_NUM_AL：所有分区。

opt：
[In] 分区格式化的格式；目前只支持 FM_FAT32 格式和 FM_FAT 格式。

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.16. ql_sdmmc_mount_ex

该函数用于将指定分区初始化并挂载到文件系统上，目前最多只支持 2 个分区。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_mount_ex(ql_sdmmc_partition_num_e part)
```

- **参数**

part：
[In] 分区号；
QL_SDMMC_PARTITION_NUM_1：第一分区；
QL_SDMMC_PARTITION_NUM_2：第二分区；
QL_SDMMC_PARTITION_NUM_AL：所有分区。

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.17. ql_sdmmc_umount_ex

该函数用于将指定分区反初始化并从文件系统上卸载，目前最多只支持 2 个分区。

- **函数原型**

```
ql_errcode_sdmmc_e ql_sdmmc_umount_ex(ql_sdmmc_partition_num_e part)
```

- **参数**

part：
[In] 分区号；
QL_SDMMC_PARTITION_NUM_1：第一分区；
QL_SDMMC_PARTITION_NUM_2：第二分区；
QL_SDMMC_PARTITION_NUM_AL：所有分区。

- **返回值**

请参考***第 2.3.1.1 章***。

2.3.18. ql_sdmmc_is_fdisk_ex

该函数用于判断 SDMMC 设备是否已进行分区。

- **函数原型**

```
bool ql_sdmmc_is_fdisk_ex(void)
```

- **参数**

无

- **返回值**

TRUE 已进行分区
FALSE 未进行分区

2.3.19. ql_sdmmc_is_format_ex

该函数用于判断 SDMMC 设备分区是否已格式化。

- **函数原型**

```
bool ql_sdmmc_is_format_ex(ql_sdmmc_partition_num_e part)
```

- **参数**

part：
[In] 分区号；
QL_SDMMC_PARTITION_NUM_1：第一分区；
QL_SDMMC_PARTITION_NUM_2：第二分区；
QL_SDMMC_PARTITION_NUM_AL：所有分区。

- **返回值**

TRUE 已格式化
FALSE 未格式化

2.3.20. ql_sdmmc_is_mount_ex

该函数用于判读 SDMMC 设备分区是否已挂载。

- **函数原型**

```
bool ql_sdmmc_is_mount_ex(ql_sdmmc_partition_num_e part)
```

- **参数**

part：
[In] 分区号；
QL_SDMMC_PARTITION_NUM_1：第一分区；
QL_SDMMC_PARTITION_NUM_2：第二分区；
QL_SDMMC_PARTITION_NUM_AL：所有分区。

- **返回值**

TRUE 已挂载
FALSE 未挂载

3 SDMMC 开发示例及调试

本章节主要介绍如何使用上述 API 进行 SDMMC 开发以及简单的调试。

3.1. SDMMC 开发示例

3.1.1. SDMMC Demo 说明

ECx00U 系列、EGx00U QuecOpen SDK 代码中提供了 SDMMC 的示例，即 *ql_sdmmc_demo.c* 文件，位于 SDK 包的*\components\ql-application\sdmmc* 目录下。

ql_sdmmc_demo.c 文件中主要包含了三个功能块，具体如下：
(1) 通过文件系统对 SDMMC 进行挂载、读写操作等；
(2) 不通过文件系统对 SDMMC 进行打开、关闭、读写等操作；
(3) 热插拔例程。

通过宏定义控制每个功能的打开或关闭，其中(1)和(2)不能同时打开，如下图所示：

```
#define QL_SDMMC_FS_TEST                0  //开启文件系统api测试
#define QL_SDMMC_MUTIL_PARTITION_TEST   0  //多分区功能

#define QL_SDMMC_EVENT_PLUGOUT          0
#define QL_SDMMC_EVENT_INSERT           1
#define QL_SDMMC_DET_TEST               1

/*****************************************************************************/

#define QL_SDMMC_ONLY_USE_DRIVER        0  //0--使用文件系统 1--仅使用sdmmc驱动层

/*****************************************************************************/
```

图 1: 功能打开/关闭宏定义

入口函数为 *ql_sdmmc_app_init()*，如下图所示。

```
void ql_sdmmc_app_init(void)
{
    QlOSStatus err = QL_SDMMC_SUCCESS;

    /*sd pin init*/
    ql_sdmmc_pin_init();

#if QL_SDMMC_ONLY_USE_DRIVER == 0
    err = ql_sdmmc_mount_demo();
    if(err!= QL_SDMMC_SUCCESS)
    {
        QL_SDMMC_DEMO_LOG("sdmmc mount demo fail err = %d", err);
    }
#endif

#ifdef QL_SDMMC_DET_TEST
    err = ql_rtos_task_create(&ql_sdmmc_det_task, QL_SDMMC_TASK_STACK_SIZE, QL_SDMMC_TASK_PRIO, "sdDEMO_det", ql_sdmmc_demo_det_thread, NU
    if (err != QL_OSI_SUCCESS)
    {
        QL_SDMMC_DEMO_LOG("creat sd task fail err = %d", err);
    }

    err = ql_rtos_timer_create(&ql_sdmmc_det_debounce_timer, ql_sdmmc_det_task, ql_sdmmc_det_callback, NULL);
    if(err != QL_OSI_SUCCESS)
    {
        QL_SDMMC_DEMO_LOG("creat timer task fail err = %d", err);
    }
#endif

#if QL_SDMMC_ONLY_USE_DRIVER
    ql_task_t sdmmc_task = NULL;
    err = ql_rtos_task_create(&sdmmc_task, QL_SDMMC_TASK_STACK_SIZE, QL_SDMMC_TASK_PRIO, "sdDEMO", ql_sdmmc_demo_thread, NULL, QL_SDMMC_TA
    if (err != QL_OSI_SUCCESS)
    {
        QL_SDMMC_DEMO_LOG("creat sd task failed err = %d", err);
    }
#else
#if QL_SDMMC_FS_TEST
#ifdef CONFIG_QUEC_PROJECT_FEATURE_FILE
    ql_task_t sdmmc_task_fs = NULL;
    err = ql_rtos_task_create(&sdmmc_task_fs, QL_SDMMC_TASK_STACK_SIZE, QL_SDMMC_TASK_PRIO, "sdDEMO_fs", ql_sdmmc_demo_fs_thread, NULL, QL
    if(err != QL_OSI_SUCCESS)
    {
        QL_SDMMC_DEMO_LOG("creat sd task fs failed err = %d", err);
    }
#endif
#endif
#endif

} « end ql_sdmmc_app_init »
```

图 2: 入口函数 ql_sdmmc_app_init()

该 Demo 先对 SDMMC 功能占用的引脚进行初始化，然后根据所打开的宏，挂载 SDMMC 设备，对 SDMMC 设备进行初始化，并将其挂载到文件系统上，然后读取 SDMMC 设备总扇区数，读、写指定扇区指定字节大小的数据。

需要注意的是，对 SDMMC 功能占用引脚进行初始化时，不能将其所占用的 GPIO 引脚作为普通 I/O 使用，否则会导致 SDMMC 设备初始化失败。SDMMC 功能占用的 GPIO 引脚可在 *ql_sdmmc_pin_init()*函数中查看到对应的 GPIO 号，如下图所示：

```
void ql_sdmmc_pin_init(void)
{
#ifdef QL_SDMMC_DET_TEST
    ql_pin_set_func(QL_SDMMC_PIN_DET    , QL_PIN_SDMMC_MODE_FUNC_GPIO);     //Pin reuse
    ql_gpio_deinit(GPIO_15);
#endif
    ql_pin_set_func(QL_PIN_SDMMC_CMD    , QL_PIN_SDMMC_MODE_FUNC);      //Pin reuse
    ql_pin_set_func(QL_PIN_SDMMC_DATA_0 , QL_PIN_SDMMC_MODE_FUNC);      //Pin reuse
    ql_pin_set_func(QL_PIN_SDMMC_DATA_1 , QL_PIN_SDMMC_MODE_FUNC);      //Pin reuse
    ql_pin_set_func(QL_PIN_SDMMC_DATA_2 , QL_PIN_SDMMC_MODE_FUNC);      //Pin reuse
    ql_pin_set_func(QL_PIN_SDMMC_DATA_3 , QL_PIN_SDMMC_MODE_FUNC);      //Pin reuse
    ql_pin_set_func(QL_PIN_SDMMC_CLK    , QL_PIN_SDMMC_MODE_FUNC);      //Pin reuse
}
```

备注

通过文件系统进行 SDMMC 设备的读写操作，可参考*文档 [3]*。

3.1.2. SDMMC Demo 自启动

SDMMC Demo 在 *ql_init_demo_thread()*线程中默认不启动，如下图所示，测试时取消注释即可。

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

/*Caution:If the macro of secure boot and the function are opene
          the secret key cannot be changed forever*/
#ifdef QL_APP_FEATURE_SECURE_BOOT
    //ql_dev_enable_secure_boot();
#endif

    /*Caution: GPIO pin must be initialized here, otherwise the
    ql_pin_cfg_init();

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

#ifdef QL_APP_FEATURE_LCD
    //ql_lcd_app_init();
#ifdef QL_APP_FEATURE_LVGL
    //ql_lvgl_app_init();
#endif
#endif
    //ql_nw_app_init();
    //ql_datacall_app_init();
    //ql_osi_demo_init();

#ifdef QL_APP_FEATURE_SDMMC
    //ql_sdmmc_app_init();
#endif

#ifdef QL_APP_FEATURE_FILE
    //ql_fs_demo_init();
#endif
```

图 3: SDMMC Demo 默认不启动

3.2. SDMMC 相关调试

3.2.1. 调试准备

若需进行 SDMMC 功能调试，可在移远通信的 LTE OPEN EVB 板上外接 SD 卡，或者直接使用 LTE OPEN EVB 板上自带的 eMMC 芯片。

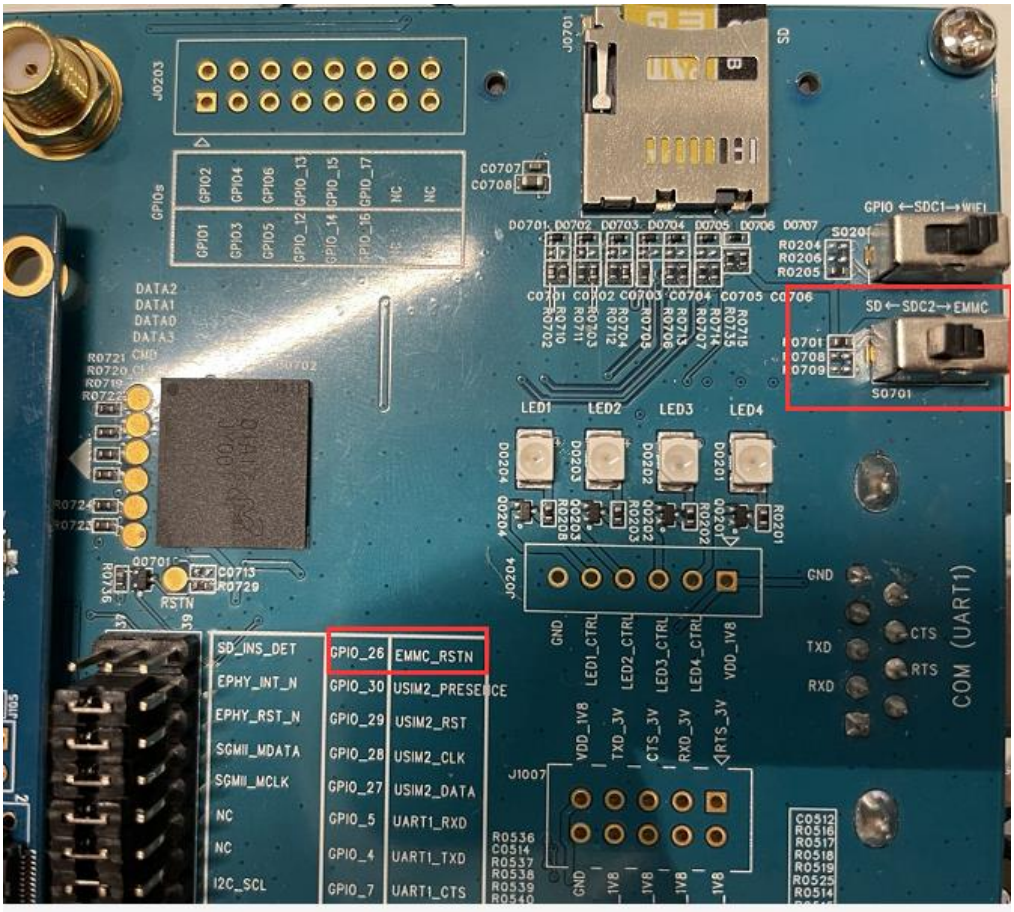

图 4: LTE OPEN EVB

使用 SD 卡调试时，右上角的拨码开关要拨到 SD 侧且去掉 GPIO_26 处的跳线帽；使用 eMMC 调试时，右上角的拨码开关要拨到 eMMC 侧且 GPIO_26 处的跳线帽要短接到 EMMC_RSTN 侧。

编译并烧录固件后，使用 USB 线连接 LTE OPEN EVB 板的 USB 端口和 PC。模块开机后，将在 PC 中的“设备管理器”显示如下所示端口；其中 USB AP Log Port 可用于显示系统调试 Log 信息，通过 *coolwatcher_usb.exe* 的 *Trace Tool* 打印的 Log 信息可以看到 SDMMC Demo 的相关信息。Log 的抓取方法请参考***文档 [2]***。

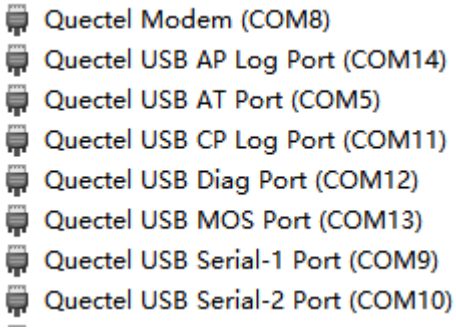

图 5: USB 端口显示

3.2.2. SDMMC 功能调试

模块开机后，取消注释 *ql_sdmmc_app_init()*实现自启动，通过 Log 信息可以看到 SDMMC 的相关信息。

AP Log 端口调试信息如下图所示：

Index	Received	Tick	Level	Description
688	20:09:34.694	65379	QOPN/I	[ql_SDMMC][ql_sdmmc_mount_demo, 62] Mount succeed
770	20:09:37.679	48893	QOPN/I	[ql_SDMMC][ql_sdmmc_demo_thread, 93] block num:15499264
771	20:09:37.679	48966	QOPN/I	[ql_SDMMC][ql_sdmmc_demo_thread, 106] sdmmc read :1234567890abcdefg
773	20:09:37.679	48984	QOPN/I	[ql_SDMMC][ql_sdmmc_demo_thread, 114] sdmmc set clk :25000000
774	20:09:37.679	48984	QOPN/I	[ql_SDMMC][ql_sdmmc_demo_thread, 117] exit ql_sdmmc_demo_thread

图 6: AP Log 信息

4 附录

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_ECx00U&EGx00U 系列_QuecOpen_Log_抓取指导
[3] Quectel_ECx00U&EGx00U 系列_QuecOpen_文件系统_API_参考手册

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	应用程序
eMMC	Embedded Multimedia Card	嵌入式多媒体卡
PC	Personal Computer	个人电脑
SD	Secure Digital Card	安全数码卡
SDK	Software Development Kit	软件开发工具包
SDHC	Secure Digital High Capacity	高容量 SD 卡
SDSC	Secure Digital Standard Capacity	标准容量 SD 卡
SDXC	Secure Digital Extended Capacity	容量扩大化 SD 卡
SDMMC	Secure Digital/MultiMedia Card	数字安全数码卡/多媒体卡
USB	Universal Serial Bus	通用串行总线