ECx00U&EGx00U 系列
QuecOpen 文件系统 API 参考手册

LTE Standard 模块系列

版本：1.0

日期：2021-08-03

状态：受控文件

上海移远通信技术股份有限公司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 51086236　　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，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。因未能遵守有关操作或设计规范而造成的损害，上海移远通信技术股份有限公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

免责声明

上海移远通信技术股份有限公司尽力确保开发中功能的完整性、准确性、及时性或效用，但不排除上述功能错误或遗漏的可能。除非其他有效协议另有规定，否则上海移远通信技术股份有限公司对开发中功能的使用不做任何暗示或明示的保证。在适用法律允许的最大范围内，上海移远通信技术股份有限公司不对任何因使用开发中功能而遭受的损失或损害承担责任，无论此类损失或损害是否可以预见。

保密义务

除非上海移远通信技术股份有限公司特别授权，否则我司所提供文档和信息的接收方须对接收的文档和信息保密，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。未经上海移远通信技术股份有限公司书面同意，不得获取、使用或向第三方泄露我司所提供的文档和信息。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，上海移远通信技术股份有限公司有权追究法律责任。

版权申明

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

文档历史

修订记录

版本	日期	作者	变更描述
-	2020-10-21	Kevin WANG	文档创建
1.0	2021-08-03	Kevin WANG/ Herry GENG	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 模块的文件系统、文件系统 API 及相关示例。

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 文件系统简介

2.1. 文件系统设计原则

ECx00U 系列和 EGx00U QuecOpen 模块文件系统 API 设计尽量靠近 C 标准函数库。模块内部文件系统采用 SFFS 格式，因此不兼容 PC 或 Linux 上的任何文件系统。

2.2. 文件系统分区

文件系统为用户提供三个磁盘分区，盘符分别为 UFS:、EFS:和 SD:。其中，UFS 盘位于内置 Flash，用于存放用户的普通文件（该分区内还设置 SFS 加密文件目录，用于存放用户加密文件）；EFS 盘为外部 SPI NOR Flash 文件系统分区，需要用户外挂 SPI NOR Flash 才可使用；SD 盘为 SD 卡分区。

2.3. 文件系统功能简述

ECx00U 系列和 EGx00U QuecOpen 模块的文件系统可进行以下操作：

- 文件操作：打开、读取、写入、关闭、删除和重命名文件、文件指针跳转、获取当前文件指针位置、设置文件指针回到开始、改变和获取文件大小等。
- 目录操作：创建、打开、读取和关闭目录、删除空目录、目录指针跳转、获取当前目录指针位置和设置目录指针回到开始等。
- 其他功能：获取文件状态和分区剩余空间大小、写入和读取配置文件、配置 SFS 文件密钥、写入和读取一个字符、同步已修改的数据到存储设备等。

3 文件系统相关 API

3.1. 头文件

文件系统 API 头文件为 *ql_fs.h*，位于 SDK 包的 *components\ql-kernel\inc* 目录下。若无特别说明，本文档所涉头文件均位于该目录下。

3.2. 函数概览

表 2：函数概览

函数	说明
ql_fopen()	打开指定文件并返回文件句柄
ql_fclose()	关闭指定文件
ql_remove()	删除指定文件
ql_fread()	从指定文件中读取数据
ql_fwrite()	向指定文件中写入数据
ql_fseek()	设置文件指针的位置
ql_frewind()	设置文件指针于文件开头
ql_ftell()	获取文件指针相对于文件开头的偏移值
ql_fstat()	通过文件句柄获取文件状态
ql_stat()	通过文件名获取文件状态
ql_ftruncate()	从文件开头截取文件为指定长度
ql_fsize()	获取文件大小
ql_file_exist()	判断文件是否存在

ql_mkdir()	创建目录
ql_opendir()	打开指定目录并返回目录句柄
ql_closedir()	关闭指定目录
ql_readdir()	从指定目录中获取文件信息
ql_telldir()	获取目录流位置
ql_seekdir()	设置目录流读取位置
ql_rewinddir()	设置目录流读取位置于目录开头
ql_rmdir()	删除指定空目录
ql_rename()	重命名文件
ql_fs_free_size()	通过文件（带路径）、目录路径或分区名，获取其所在分区的剩余空间大小
ql_nvm_fwrite()	写入简单配置文件
ql_nvm_fread()	读取简单配置文件
ql_sfs_setkey()	配置读写的 SFS 文件密钥
ql_fs_free_size_by_fd()	通过文件句柄获取文件所在分区的剩余空间大小
ql_fputc()	向文件中写入一个字符
ql_fgetc()	从文件中读取一个字符
ql_ferror()	判断当前文件句柄是否有效
ql_fsync()	同步内存中已修改的文件数据到储存设备
ql_cust_nvm_fwrite()	写入用户自定义简单配置文件
ql_cust_nvm_fread()	读取用户自定义简单配置文件

3.3. 函数详解

3.3.1. ql_fopen

该函数用于打开指定文件并返回文件句柄。文件句柄为 32 位有符号型数。

- **函数原型**

```
QFILE ql_fopen(const char * file_name, const char *mode)
```

- **参数**

file_name：

[In] 文件名（带路径）。长度应不大于 132 字节，标准格式为“磁盘名:/路径名/文件名”。

暂支持如下分区：

UFS	主分区，用于存放用户的普通文件。
SFS	加密文件目录，与 UFS 共享一个分区，用于存放用户加密文件。
EFS	外部 SPI NOR Flash 文件系统分区。
SD	SD 卡分区。

mode：

[In] 文件打开方式。与 C 标准函数库一致，常用如“wb+”。

- **返回值**

文件句柄	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.2. ql_close

该函数用于关闭指定文件。

- **函数原型**

```
int ql_fclose(QFILE fd)
```

- **参数**

fd：

[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.3. ql_remove

该函数用于删除指定文件。

- **函数原型**

```
int ql_remove(const char *file_name)
```

- **参数**

file_name：
[In] 文件名称。格式与 *ql_fopen()*中的 *file_name* 一致。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.4. ql_fread

该函数用于从指定文件中读取数据。

- **函数原型**

```
int ql_fread(void * buffer, size_t size, size_t num, QFILE fd)
```

- **参数**

buffer：
[In] 存储读取数据的内存地址。不可为空指针。

size：
[In] 需读取的每个元素大小。单位：字节。

num：
[In] 需读取的元素个数。最终读取数据大小为 *size*num*。

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

读取数据大小	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.5. ql_fwrite

该函数用于向指定文件中写入数据。

- **函数原型**

```
int ql_fwrite(void * buffer, size_t size, size_t num, QFILE fd)
```

- **参数**

buffer：
[In] 存储写入数据的内存地址。不可为空指针。

size：
[In] 需写入的每个元素大小。单位：字节。

num：
[In] 需写入的元素个数。最终写入数据大小为 *size*num*。

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

写入数据大小	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.6. ql_fseek

该函数用于设置文件指针的位置。

- **函数原型**

```
int ql_fseek(QFILE fd, long offset, int origin)
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

offset：
[In] 文件指针位置偏移值。单位：字节。

origin：
[In] 开始添加 *offset* 的位置。包含如下值：
QL_SEEK_SET 文件开头
QL_SEEK_CUR 文件指针的当前位置
QL_SEEK_END 文件末尾

- **返回值**

文件指针相对于文件开头的偏移值 函数执行成功
其他值 函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.7. ql_frewind

该函数用于设置文件指针于文件开头。

- **函数原型**

```
int ql_frewind(QFILE fd)
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.8. ql_ftell

该函数用于获取文件指针相对于文件开头的偏移值。

- **函数原型**

```
int ql_ftell(QFILE fd);
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

文件指针偏移值 函数执行成功
其他值 函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.9. ql_fstat

该函数用于通过文件句柄获取文件状态。

- **函数原型**

```
int ql_fstat(QFILE fd, struct stat *st)
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

st：
[Out] 文件状态结构体。不可为空指针。可参考 C 标准函数库中 *stat* 结构体。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.10. ql_stat

该函数用于通过文件名获取文件状态。

- **函数原型**

```
int ql_stat(const char *file_name, struct stat *st)
```

- **参数**

file_name：
[In] 文件名（带路径）。格式与 *ql_fopen()*中的 *file_name* 一致。

st：
[Out] 文件状态结构体。不可为空指针。可参考 C 标准函数库中 stat 结构体。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.11. ql_ftruncate

该函数用于从文件开头截取文件为指定长度。

- **函数原型**

```
int ql_ftruncate(QFILE fd, uint length)
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

length：
[In] 指定截取长度。单位：字节。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.12. ql_fsize

该函数用于获取文件大小。单位：字节。

- **函数原型**

```
int ql_fsize(int fd)
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

文件大小	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.13. ql_file_exist

该函数用于判断文件是否存在。

- **函数原型**

```
int ql_file_exist(const char *file_path)
```

- **参数**

file_path：
[In] 文件名（带路径）。格式与 *ql_fopen()*中的 *file_name* 一致。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.14. ql_mkdir

该函数用于创建目录。

- **函数原型**

```
int ql_mkdir(const char *path, uint mode)
```

- **参数**

path：
[In] 目录名（带路径）。路径格式与 *ql_fopen()*中的 *file_name* 一致。

mode：
[In] 该参数为无效参数。此处为靠近 C 标准函数库而设计，输入 0 或正数即可。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.15. ql_opendir

该函数用于打开指定目录并返回目录句柄。

- **函数原型**

```
QDIR *ql_opendir(const char *path)
```

- **参数**

path：
[In] 目录名（带路径）。路径格式与 *ql_fopen()*中的 *file_name* 一致。

- **返回值**

目录句柄	函数执行成功
NULL	函数执行失败

3.3.16. ql_closedir

该函数用于关闭指定目录。

- **函数原型**

```
int ql_closedir(QDIR *pdir)
```

- **参数**

pdir：
[In] 目录句柄。即 *ql_opendir()*执行成功后的返回值。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.17. ql_readdir

该函数用于从指定目录中获取文件信息。文件信息结构体 *qdirent* 详见***第 3.3.17.1 章***。

- **函数原型**

```
qdirent * ql_readdir(QDIR *pdir)
```

- **参数**

pdir：
[In] 目录句柄。即 *ql_opendir()*执行成功后的返回值。

- **返回值**

指向 *qdirent* 结构体的非空指针	函数执行成功
NULL	函数执行失败

3.3.17.1. qdirent

文件信息 *qdirent* 结构体定义如下：

```
    typedef struct
    {
        int d_ino;          ///< inode number, file system implementation can use it for any purpose
        uint8 d_type;       ///< type of file
        char d_name[256];   ///< file name
}qdirent;
```

- **参数**

类型	参数	描述
int	*d_ino*	索引节点号
uint8	*d_type*	文件类型
char	*d_name*	文件名，最长 255 字符。

3.3.18. ql_telldir

该函数用于获取目录流位置。

- **函数原型**

```
int ql_telldir(QDIR *pdir);
```

- **参数**

pdir：
[In] 目录句柄。即 *ql_opendir()*执行成功后的返回值。

- **返回值**

目录流位置　函数执行成功
其他值　　函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.19. ql_seekdir

该函数用于设置目录流读取位置。

- **函数原型**

```
int ql_seekdir(QDIR *pdir, int offset)
```

- **参数**

pdir：
[In] 目录句柄。即 *ql_opendir()*执行成功后的返回值。

offset：
[In] 目录距离开头的偏移值。单位：字节。

- **返回值**

实际设置目录流读取位置	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.20. ql_rewinddir

该函数用于设置目录流读取位置于目录开头。

- **函数原型**

```
int ql_rewinddir(QDIR *pdir)
```

- **参数**

pdir：
[In] 目录句柄。即 *ql_opendir()*执行成功后的返回值。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.21. ql_rmdir

该函数用于删除指定空目录。

- **函数原型**

```
int ql_rmdir(const char *path)
```

- **参数**

path：
[In] 目录名（带路径）。路径格式与 *ql_fopen()*中的 *file_name* 一致。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.22. ql_rename

该函数用于重命名文件。

- **函数原型**

```
int ql_rename(const char *oldpath, const char *newpath);
```

- **参数**

oldpath：
[In] 原文件名（带路径）。格式与 *ql_fopen()*中 *file_name* 一致。

newpath：
[In] 新文件名（带路径）。格式与 *ql_fopen()*中 *file_name* 一致。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.23. ql_fs_free_size

该函数用于通过文件（带路径）、目录路径或分区名，获取其所在分区的剩余空间大小。单位：字节。

- **函数原型**

```
int64 ql_fs_free_size(const char *path)
```

- **参数**

path：
[In] 文件（带路径）、目录路径或分区名。如 UFS:a.txt、UFS 或 SD。

- **返回值**

分区剩余空间大小	函数执行成功
其他值	函数执行失败

3.3.24. ql_nvm_fwrite

该函数用于写入简单配置文件。

- **函数原型**

```
int ql_nvm_fwrite(const char *config_file_name, void *buffer, size_t size, size_t num)
```

- **参数**

config_file_name：
[In] 简单配置文件名。不可带路径。

buffer：
[In] 存储写入数据的内存地址。不可为空指针。

size：
[In] 需写入的每个元素大小。单位：字节。

num：
[In] 需写入的元素个数。最终写入数据大小为 *size*num*。

- **返回值**

写入数据大小	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.25. ql_nvm_fread

该函数用于读取简单配置文件。

- **函数原型**

```
int ql_nvm_fread(const char *config_file_name, void *buffer, size_t size, size_t num)
```

- **参数**

config_file_name：
[In] 简单配置文件名。不可带路径。

buffer：
[In] 存储读取数据的内存地址。不可为空指针。

size：
[In] 需读取的每个元素大小。单位：字节。

num：
[In] 需读取的元素个数。最终读取数据大小为 *size*num*。

- **返回值**

读取数据大小	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.26. ql_sfs_setkey

该函数用于配置读写的 SFS 文件密钥。

- **函数原型**

```
int ql_sfs_setkey(void *key, uint8_t len)
```

- **参数**

key：
[In] 存储密钥的内存地址。不可为空指针。

len：
[In] 需设置的密钥的长度。范围：1~16。单位：字节。

- **返回值**

详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.27. ql_fs_free_size_by_fd

该函数用于通过文件句柄获取文件所在分区的剩余空间大小。单位：字节。

- **函数原型**

```
int64 ql_fs_free_size_by_fd(int fd);
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

分区剩余空间大小	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.28. ql_fputc

该函数用于向文件中写入一个字符。

- **函数原型**

```
int ql_fputc(int chr, int fd);
```

- **参数**

chr：
[In] 写入的字符的 ASCII 值。

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

写入字符的 ASCII 值	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.29. ql_fgetc

该函数用于从文件中读取一个字符。

- **函数原型**

```
int ql_fgetc(int fd);
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

读取字符的 ASCII 值	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.30. ql_ferror

该函数用于判断当前文件句柄是否有效。

- **函数原型**

```
int ql_ferror(int fd);
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

0	当前文件句柄有效
其他值	无效输入。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.31. ql_fsync

该函数用于同步内存中已修改的文件数据到储存设备。

- **函数原型**

```
int ql_fsync(int fd);
```

- **参数**

fd：
[In] 文件句柄。即 *ql_fopen()*执行成功后的返回值。

- **返回值**

0 函数执行成功
其他值 函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.32. ql_cust_nvm_fwrite

该函数用于写入用户自定义简单配置文件。

- **函数原型**

```
int ql_cust_nvm_fwrite(void *buffer, size_t size, size_t num);
```

- **参数**

buffer：
[In] 存储写入数据的内存地址。不可为空指针。

size：
[In] 需写入的每个元素大小。单位：字节。

num：
[In] 需写入的元素个数。最终写入数据大小为 *size*num*。

- **返回值**

写入数据大小 函数执行成功
其他值 函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

3.3.33. ql_cust_nvm_fread

该函数用于读取用户自定义简单配置文件。

- **函数原型**

```
int ql_cust_nvm_fread(void *buffer, size_t size, size_t num);
```

- **参数**

buffer：
[In] 存储读取数据的内存地址。不可为空指针。

size：
[In] 需读取的每个元素大小。单位：字节。

num：
[In] 需读取的元素个数。最终读取数据大小为 *size*num*。

- **返回值**

读取数据大小	函数执行成功
其他值	函数执行失败。详见头文件中 *ql_file_errcode_e* 枚举相关信息。

4 示例

SDK 中提供文件系统代码示例文件 *ql_fs_demo.c*，路径为 *components\ql-application\fs*。用户可参考此文件实现相关功能。

5 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_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
ASCII	American Standard Code for Information Interchange	美国信息交换标准码
RTOS	Real Time Operating System	实时操作系统
SDK	Software Development Kit	软件开发工具包
UFS	User File System	用户文件系统
EFS	External File System	外挂 Flash 文件系统
SD	Secure Digital Memory Card/SD card	安全数码卡
SFFS	Small-File-Oriented File System	面向小文件的文件系统
SFS	Secure File System	安全文件系统
SPI	Serial Peripheral Interface	串行外设接口