ECx00U&EGx00U 系列

QuecOpen CSDK 快速开发指导

LTE Standard 模块系列

版本：1.0

日期：2021-09-17

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

商标

第三方权利

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
-	2021-07-23	Neo KONG	文档创建
1.0	2021-09-17	Neo KONG	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 CSDK 目录结构 8
3 搭建编译环境 10
3.1. 主机系统 10
3.2. 应用程序编译环境 10
4 开发应用程序 11
4.1. 应用程序实例说明 11
4.2. 添加应用程序功能组件 12
4.2.1. 创建应用程序功能组件目录 12
4.2.2. 创建应用线程 12
4.2.3. 修改配置文件 13
4.2.4. 添加编译 13
4.2.5. 链接应用程序功能组件 14
4.2.6. 功能组件编译开关 14
5 编译应用程序 17
5.1. 编译说明 17
5.2. 编译步骤 17
5.3. 编译结果 19
5.4. 清除编译文件 19
6 裁剪功能 20
6.1. 裁剪内核功能 20
6.2. 裁剪 BootLoader 功能 23
7 预置文件 25
7.1. 启用预置文件打包 25
7.2. 配置预置文件 26
7.3. 下载或升级预置文件 26
8 调整 Flash 分区 28
9 固件烧录 29
10 常见问题 30
10.1. Linux 下常见编译报错 30
11 附录 参考文档及术语缩写 32

表格索引

表 1：适用模块 7
表 2：CSDK 目录结构描述 8
表 3：内核中可裁剪库 21
表 4：BootLoader 可裁剪库 23
表 5：参考文档 32
表 6：术语缩写 32

图片索引

图 1：CSDK 目录结构 9
图 2：应用程序入口 11
图 3：应用程序功能组件目录示例 12
图 4：创建功能组件应用线程 12
图 5：修改配置文件 13
图 6：添加编译 13
图 7：链接新应用程序功能组件和增加 Option 开关控制 14
图 8：配置编译开关 15
图 9：增加宏控 16
图 10：CSDK 原始固件 17
图 11：编译应用程序 18
图 12：编译结果 19
图 13：内核中可裁剪库 20
图 14：内核功能裁剪 21
图 15：功能初始化函数 24
图 16：预置打包编译配置 25
图 17：增加需要预置文件的功能开关 25
图 18：预置打包配置文件 26
图 19：FOTA 升级忽略预置文件 27

1 引言

移远通信 ECx00U 系列和 EGx00U-CN 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。

本文档主要介绍在 QuecOpen®方案下，移远通信 ECx00U 系列和 EGx00U-CN 模块的 CSDK 目录结构、编译环境搭建、应用程序开发及编译流程、功能裁剪、分区调整、固件烧录及开发常见问题。

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 CSDK 目录结构

移远通信实际发布的 QuecOpen CSDK 中，不同模块版本的目录可能略有差异。QuecOpen CSDK 目录如***图 1*** 所示，详细信息如下表。

表 2：CSDK 目录结构描述

目录名称	描述
cmake、*prebuilts*、*tools*	包含相关编译工具、配置文件和编译脚本。
components/appstart	包含内核启动程序源文件
components/bootloader	包含引导加载程序源文件
components/hal	包含分区和 flash 配置文件
components/ql-config	包含当前版本的默认 Kernel 固件、原始 App 固件和预置文件等。
components/ql-kernel	包含所有开放的 API 头文件、库文件。
components/ql-application	包含应用程序参考例程，实现了各个功能组件的应用示例。
components/libs *components/net* *components/newlib*	包含应用程序需要用到的各个头文件以及库文件。
build_all.bat、*build_all.sh*	• Windows 操作系统： 在 Windows 命令行窗口中根据所带的参数进行应用程序编译。详情请参考***第 5 章***。 • Linux 操作系统： 在 Linux 命令行窗口中根据所带的参数进行应用程序编译。Linux 需要在管理员模式下进行编译，使用 **bash** 命令执行脚本。详情请参考***第 5 章***。
CmakeLists.txt	顶层编译配置文件。

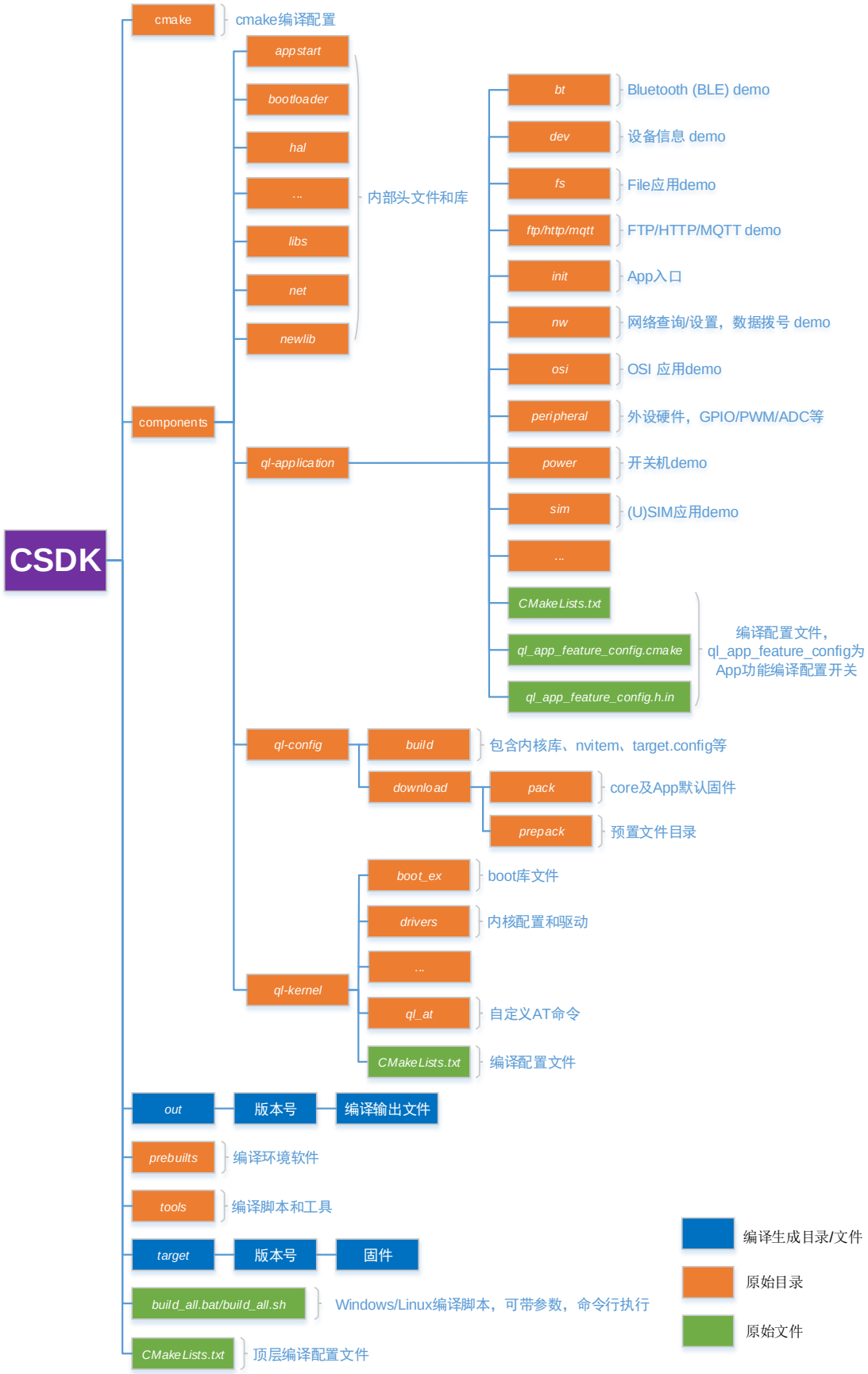

CSDK
cmake
cmake编译配置
components
appstart
bootloader
hal
...
libs
net
newlib
内部头文件和库
ql-application
bt
Bluetooth (BLE) demo
dev
设备信息 demo
fs
File应用demo
ftp/http/mqtt
FTP/HTTP/MQTT demo
init
App入口
nw
网络查询/设置，数据拨号 demo
osi
OSI 应用demo
peripheral
外设硬件，GPIO/PWM/ADC等
power
开关机demo
sim
(U)SIM应用demo
...
CMakeLists.txt
ql_app_feature_config.cmake
ql_app_feature_config.h.in
编译配置文件，
ql_app_feature_config为
App功能编译配置开关
ql-config
build
包含内核库、nvitem、target.config等
download
pack
core及App默认固件
prepack
预置文件目录
ql-kernel
boot_ex
boot库文件
drivers
内核配置和驱动
...
ql_at
自定义AT命令
CMakeLists.txt
编译配置文件
out
版本号
编译输出文件
prebuilts
编译环境软件
tools
编译脚本和工具
target
版本号
固件
build_all.bat/build_all.sh
Windows/Linux编译脚本，可带参数，命令行执行
CMakeLists.txt
顶层编译配置文件
编译生成目录/文件
原始目录
原始文件

图 1：CSDK 目录结构

3 搭建编译环境

3.1. 主机系统

若运行编译环境的主机为 Windows 操作系统，需确保操作系统为 32 位或 64 位的 Windows 7/10，同时需安装如下软件：

- Visual C++ Redistributable for Visual Studio 2015 x86 或以上版本（32 位 Windows 系统）
- Visual C++ Redistributable for Visual Studio 2015 x64 或以上版本（64 位 Windows 系统）

若运行编译环境的主机为 Linux 操作系统，需确保操作系统为 Ubuntu16.04 或 Ubuntu20.04，语言使用 Python 3.5 及以上版本，同时需执行如下命令安装所需组件：

```
sudo apt install build-essential python3 python3-tk qtbase5-dev
sudo apt install libc6:i386 libstdc++6:i386 zlib1g:i386
sudo apt install protobuf-compiler
```

3.2. 应用程序编译环境

移远通信 ECx00U 系列和 EGx00U-CN QuecOpen 模块的应用程序编译工具链采用 gcc-arm-none-eabi，此为 ARM 裸机系统的 GCC 工具链。QuecOpen CSDK 中集成了 gcc-arm-none-eabi 工具链，位于 *prebuilts* 目录下，版本为 7.2.1。

在编译应用程序时，仅支持使用 QuecOpen CSDK 中集成的 gcc-arm-none-eabi 工具链，不支持使用主机系统安装的 gcc-arm-none-eabi 工具链。

4 开发应用程序

4.1. 应用程序实例说明

QuecOpen CSDK 中提供应用程序实例，用户可参考 CSDK 中 *components\ql-application* 目录下的程序进行应用程序的开发。

QuecOpen CSDK 中 *components\ql-application\init* 目录下的 *ql_init.c* 文件中包含应用程序入口，即 *appimg_enter()*。在该函数中启动一个初始化线程，然后在该初始化线程中调用各个功能组件的初始化接口。各个功能组件按照文件夹划分，用户可参照下图添加。

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

#ifdef QL_APP_FEATURE_FILE
    ql_fs_demo_init();
#endif

#ifdef QL_APP_FEATURE_MQTT
    ql_mqtt_app_init();
#endif
    ql_rtos_task_sleep_ms(10);
    ql_rtos_task_delete(NULL);
}

int appimg_enter(void *param)
{
    QlOSStatus err = QL_OSI_SUCCESS;
    ql_task_t ql_init_task = NULL;

    QL_INIT_LOG("init demo enter");
    prvInvokeGlobalCtors();

    err = ql_rtos_task_create(&ql_init_task, 1024, APP_PRIORITY_NORMAL, "ql_init", ql_init_demo_thread, NULL, 1);
    if(err != QL_OSI_SUCCESS)
    {
        QL_INIT_LOG("init failed");
    }
    return err;
}
```

图 2：应用程序入口

4.2. 添加应用程序功能组件

本章节以 QeucOpen CSDK 中提供的音频应用程序为例介绍如何添加应用程序功能组件。

4.2.1. 创建应用程序功能组件目录

在 QuecOpen CSDK 的 *components/ql-application* 目录下添加需要的功能目录，目录中存放源文件、头文件以及编译配置文件 *CMakeLists.txt*。以音频应用程序为例，目录结构如下，其中 *inc* 文件夹用于存放头文件：

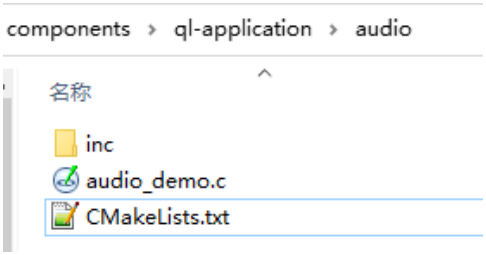


图 3：应用程序功能组件目录示例

4.2.2. 创建应用线程

在功能组件源代码文件的初始化函数 *ql_audio_app_init()*中，使用 *ql_rtos_task_create()*创建功能组件的应用线程。示例如下：

```
static void ql_audio_demo_thread(void *param)
{
    QlOSStatus err = QL_OSI_SUCCESS;
    QL_AUDIO_LOG("audio demo thread enter, param 0x%x", param);

    for (int n = 0; n < 10; n++)
    {
        QL_AUDIO_LOG("hello audio demo %d", n);
        ql_rtos_task_sleep_ms(500);
    }

    err = ql_rtos_task_delete(NULL);
    if(err != QL_OSI_SUCCESS)
    {
        QL_AUDIO_LOG("task deleted failed");
    }
}

void ql_audio_app_init(void)
{
    QlOSStatus err = QL_OSI_SUCCESS;
    ql_task_t ql_audio_task = NULL;

    QL_AUDIO_LOG("audio demo enter");

    err = ql_rtos_task_create(&ql_audio_task, 1024, APP_PRIORITY_NORMAL, "ql_audio", ql_audio_demo_thread, NULL, 5);
    if(err != QL_OSI_SUCCESS)
    {
        QL_AUDIO_LOG("audio task create failed");
    }
}
```

图 4：创建功能组件应用线程

4.2.3. 修改配置文件

按需修改应用程序的配置文件 *components\ql-application\audio\CMakeLists.txt*，主要修改如下三处：

```
set(target ql_app_audio) 库名称

add_library(${target} STATIC)
set_target_properties(${target} PROPERTIES ARCHIVE_OUTPUT_DIRECTORY ${out_app_lib_dir})
target_compile_definitions(${target} PRIVATE OSI_LOG_TAG=LOG_TAG_QUEC)
target_include_directories(${target} PUBLIC inc) 头文件路径
#target_link_libraries(${target} PRIVATE ql_api_common)

target_sources(${target} PRIVATE
    audio_demo.c
)                    源文件列表，支持相对路径

relative_glob(srcs include/*.h src/*.c inc/*.h)
beautify_c_code(${target} ${srcs})
```

图 5：修改配置文件

4.2.4. 添加编译

修改 *components\ql-application\CMakeLists.txt* 配置文件，在文件中添加应用程序功能组件目录，用于在编译固件包时编译该目录。示例如下：

```
# Copyright (C) 2020 QUECTEL Technologies Limited and/or its affiliates("QUECTEL").
# All rights reserved.
#

add_subdirectory_if_exist(init)

add_subdirectory_if_exist(nw)

add_subdirectory_if_exist(peripheral)

add_subdirectory_if_exist(osi)

add_subdirectory_if_exist(dev)

add_subdirectory_if_exist(sim)

add_subdirectory_if_exist(power)

if(QL_APP_FEATURE_FILE)
add_subdirectory_if_exist(fs)
endif()

if(QL_APP_FEATURE_AUDIO)
add_subdirectory_if_exist(audio)
if(QL_APP_FEATURE_AUDIO_TTS)
add_subdirectory_if_exist(tts)
endif()
endif()
```

图 6：添加编译

4.2.5. 链接应用程序功能组件

修改 *components/ql-application/init* 目录下的配置文件 *CMakeLists.txt*，在该文件中添加链接新应用程序功能组件的库名称。若针对该应用程序增加了功能组件编译开关，则还需在该文件中增加 option 开关控制。示例如下：

```
if(CONFIG_APPIMG_LOAD_FLASH)
    set(target ${QL_APP_BUILD_VER})
    add_appimg_flash_ql_example(${target} ql_init.c)

    target_link_libraries(${target} PRIVATE ql_app_nw ql_app_peripheral ql_app_osi ql_app_dev ql_app_sim ql_app_power)
    if(QL_APP_FEATURE_FILE_ZIP)
        target_link_libraries(${target} PRIVATE ql_app_zip)
    endif()
    if(QL_APP_FEATURE_FTP)
        target_link_libraries(${target} PRIVATE ql_app_ftp)
    endif()
    if(QL_APP_FEATURE_HTTP)
        target_link_libraries(${target} PRIVATE ql_app_http)
    endif()
    if(QL_APP_FEATURE_MMS)
        target_link_libraries(${target} PRIVATE ql_app_mms)
    endif()
    if(QL_APP_FEATURE_MQTT)
        target_link_libraries(${target} PRIVATE ql_app_mqtt)
    endif()
    if(QL_APP_FEATURE_SSL)
        target_link_libraries(${target} PRIVATE ql_app_ssl)
    endif()
    if(QL_APP_FEATURE_PING)
        target_link_libraries(${target} PRIVATE ql_app_ping)
    endif()
    if(QL_APP_FEATURE_NTP)
        target_link_libraries(${target} PRIVATE ql_app_ntp)
    endif()
    if(QL_APP_FEATURE_LBS)
        target_link_libraries(${target} PRIVATE ql_app_lbs)
    endif()

    if(QL_APP_FEATURE_CTSREG)
        target_link_libraries(${target} PRIVATE ql_app_ctsreg)
    endif()

    if(QL_APP_FEATURE_SOCKET)
        target_link_libraries(${target} PRIVATE ql_app_socket)
    endif()

    if(QL_APP_FEATURE_AUDIO)
        target_link_libraries(${target} PRIVATE ql_app_audio)
        if(QL_APP_FEATURE_TTS)
            add_library(ql_tts_api STATIC IMPORTED)
            set_target_properties(ql_tts_api PROPERTIES IMPORTED_LOCATION ${SOURCE_TOP_DIR}/components/newlib/armca5/libql_api_tts.a)
            target_link_libraries(${target} PRIVATE ql_app_tts ql_tts_api ${libm_file_name})
        endif()
    endif()
```

图 7：链接新应用程序功能组件和增加 Option 开关控制

应用程序功能组件添加完成，且相关配置文件配置完成后，按照***第 5 章***编译固件。

4.2.6. 功能组件编译开关

为便于应用侧对所需功能进行配置或裁剪，可在 QuecOpen CSDK 的 *components/ql-application* 目录下修改 *ql_app_feature_config.cmake* 和 *ql_app_feature_config.h.in* 文件。*ql_app_feature_config.cmake* 用于配置编译脚本可编译的功能组件；*ql_app_feature_config.h.in* 文件用于生成组件使能宏控，生成的宏控将在代码中使用。对所需功能进行配置或裁减在两个文件中都需同步修改（有关裁剪功能的详情，请参考***第 6 章***）。

ql_app_feature_config.cmake 文件内容如下图，各功能组件若设置为“ON”即表示执行编译脚本时将编译该功能组件；若设置为“OFF”即表示不编译该功能组件。

```
message("\n")

option(QL_APP_FEATURE_FTP  "Enable FTP" ON)
message(STATUS "QL_APP_FEATURE_FTP ${QL_APP_FEATURE_FTP}")

option(QL_APP_FEATURE_HTTP  "Enable HTTP" ON)
message(STATUS "QL_APP_FEATURE_HTTP ${QL_APP_FEATURE_HTTP}")

option(QL_APP_FEATURE_MQTT  "Enable MQTT" ON)
message(STATUS "QL_APP_FEATURE_MQTT ${QL_APP_FEATURE_MQTT}")

option(QL_APP_FEATURE_SSL  "Enable SSL" ON)
message(STATUS "QL_APP_FEATURE_SSL ${QL_APP_FEATURE_SSL}")

option(QL_APP_FEATURE_FILE  "Enable FILE" ON)
message(STATUS "QL_APP_FEATURE_FILE ${QL_APP_FEATURE_FILE}")

option(QL_APP_FEATURE_AUDIO  "Enable AUDIO" ON)
message(STATUS "QL_APP_FEATURE_AUDIO ${QL_APP_FEATURE_AUDIO}")
if(QL_APP_FEATURE_AUDIO)
option(QL_APP_FEATURE_AUDIO_TTS  "Enable TTS" ON)
else()
option(QL_APP_FEATURE_AUDIO_TTS  "Enable TTS" OFF)
endif()
message(STATUS "QL_APP_FEATURE_AUDIO_TTS ${QL_APP_FEATURE_AUDIO_TTS}")


option(QL_APP_FEATURE_WIFISCAN  "Enable WIFI-Scan" ON)
message(STATUS "QL_APP_FEATURE_WIFISCAN ${QL_APP_FEATURE_WIFISCAN}")

option(QL_APP_FEATURE_BT  "Enable BT" ON)
message(STATUS "QL_APP_FEATURE_BT ${QL_APP_FEATURE_BT}")

option(QL_APP_FEATURE_GPS  "Enable GPS" ON)
message(STATUS "QL_APP_FEATURE_GPS ${QL_APP_FEATURE_GPS}")


option(QL_APP_FEATURE_LCD  "Enable LCD" ON)
message(STATUS "QL_APP_FEATURE_LCD ${QL_APP_FEATURE_LCD}")

option(QL_APP_FEATURE_CAMERA  "Enable CAMERA" ON)
message(STATUS "QL_APP_FEATURE_CAMERA ${QL_APP_FEATURE_CAMERA}")
```

图 8：配置编译开关

ql_app_feature_config.h.in 文件内容如下图，在 *ql_app_feature_config.cmake* 文件中添加了功能组件编译开关后，参考下图增加相应功能组件的宏控：

```
#ifndef _QL_APP_FEATURE_CONFIG_H_
#define _QL_APP_FEATURE_CONFIG_H_

// @AUTO_GENERATION_NOTICE@

/**
 * whether to eanble APP feature FTP
 */
#cmakedefine QL_APP_FEATURE_FTP

/**
 * whether to eanble APP feature HTTP
 */
#cmakedefine QL_APP_FEATURE_HTTP

/**
 * whether to eanble APP feature MQTT
 */
#cmakedefine QL_APP_FEATURE_MQTT

/**
 * whether to eanble APP feature SSL
 */
#cmakedefine QL_APP_FEATURE_SSL

/**
 * whether to eanble APP feature FILE
 */
#cmakedefine QL_APP_FEATURE_FILE

/**
 * whether to eanble APP feature AUDIO
 */
#cmakedefine QL_APP_FEATURE_AUDIO

/**
 * whether to eanble APP feature TTS
 * if TTS is eanbled, you need enable AUDIO too
 */
#cmakedefine QL_APP_FEATURE_AUDIO_TTS
```

图 9：增加宏控

5 编译应用程序

5.1. 编译说明

在 QuecOpen CSDK 的 *components\ql-config\download\pack* 目录下包含一个使用模块型号命名的文件夹，该文件夹下保存了当前模块版本的默认 Kernel 固件、移远通信存放的原始应用程序固件，及相关*.map* 和*.elf* 文件，同时提供了含 Kernel 和应用程序的合并固件。

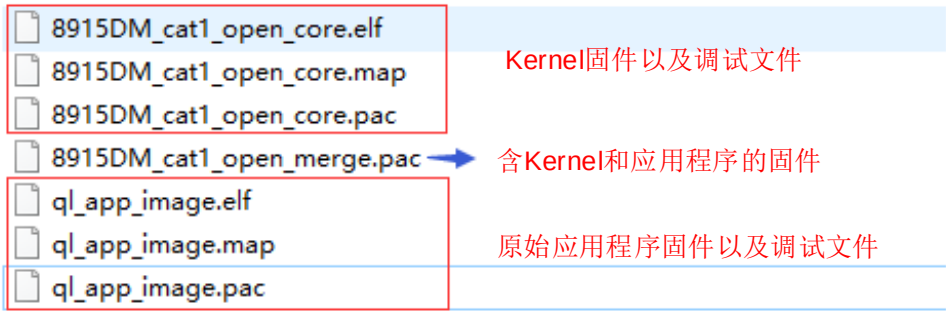


图 10：CSDK 原始固件

应用侧源代码位于 *components\ql-application* 目录下，使用 QuecOpen CSDK 根目录下的编译脚本文件 *build_all.bat*（主机系统为 Windows 系统）或 *build_all.sh*（主机系统为 Linux 系统）进行编译。两个脚本文件使用的命令参数一致，但 *build_all.sh* 脚本文件需使用 **bash** 命令，即 **bash build_all.sh**。以下章节将以主机系统为 Windows 系统为例介绍应用程序的编译过程。

5.2. 编译步骤

1. 打开命令行窗口或 PowerShell（仅 Windows 10 支持 PowerShell）执行 **build_all.bat -h** 查询使用方法。常用命令如下：

   - 编译命令 **build_all <r/new> <Project> <Version>[VOLTE] [DSIM] [debug/release]**

     <r/new> 编译类型。r 表示增量编译，w 表示全新编译。

     <Project > 当前移远通信模块型号。该参数需和 *components\ql-config\download\pack* 目录下的文件夹名称一致。

     <Version > 编译生成的应用程序固件名称，可自定义。

[VOLTE]	VoLTE 功能选项（可选）。VOLTE 或 NOVOLTE；默认为 NOVOLTE。
[DSIM]	单双卡选项（可选）。SINGLESIM：单卡，DOUBLESIM：双卡；默认为 SINGLESIM。
[debug/release]	版本选项（可选）。debug 将编译生成调试版本，release 将编译生成正式版本；默认为 release。

- 清除上一次编译过程中生成的文件：**build_all clean**。

2. 在命令行或 PowerShell（仅 Windows 10 支持 PowerShell），进入 QuecOpen CSDK 根目录，执行 **build_all <r/new> <Project> <Version>[VOLTE] [DSIM] [debug/release]**开始编译。

   命令执行后，编译脚本首先检查 QuecOpen CSDK 中 *components\ql-config\download\pack* 目录下是否包含名为**<Project>**的目录，若包含则继续编译；若无该目录，则报错，编译失败。执行结果如下图所示：

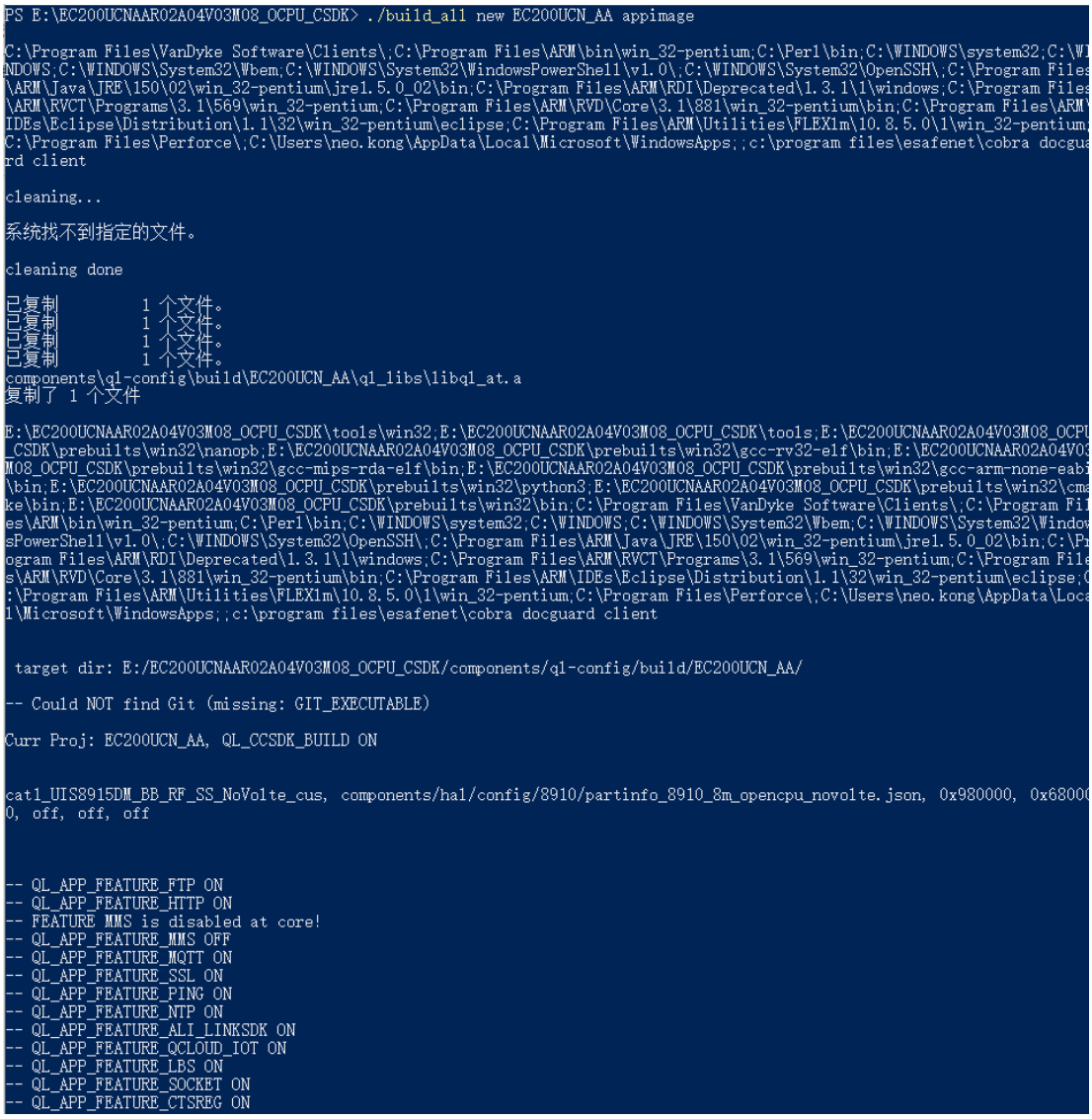


图 11：编译应用程序

5.3. 编译结果

编译过程中将在 QuecOpen CSDK 根目录下生成 *out* 目录，用于存放生成的编译文件。

编译成功后，将在 QuecOpen CSDK 根目录下生成 *target* 目录，用于存放编译后的 Kernel 固件、目标应用程序固件以及含 Kernel 和应用程序的固件。若仅更新应用程序固件，烧录新生成的应用程序固件即可；若需同时更新应用程序固件和 Kernel 固件，则烧录新生成的含 Kernel 和应用程序的固件包即可。

图 12：编译结果

5.4. 清除编译文件

打开命令行窗口或 PowerShell（仅 Windows 10 支持 PowerShell），执行 **build_all clean** 可清除 QuecOpen CSDK 中 *out* 目录下存放的编译文件，但并不会删除 *target* 目录下存放的目标固件。当编译命令参数相同时，*target* 目录下存放的上一次编译生成的版本固件会在当前编译操作成功前自动清除，若编译命令参数不同，该目录下存放的上一次编译生成的版本固件将不会清除。

6 裁剪功能

6.1. 裁剪内核功能

QuecOpen CSDK 默认开放所有功能组件，根目录下的 *CMakeLists.txt* 中列出了可裁剪的库。各个库根据内核功能组件宏控进行裁剪控制，基础库不允许裁剪。可在 *components\ql-config\build\{项目型号}\8915DM_cat1_open\target.config* 中完成所需的功能裁剪。裁剪后需进行全新编译，Kernel 固件使用编译后生成的文件。可裁剪库参考下表。

```
# "all_libs" is a target to link all app libraries
set(target all_libs)
set(all_libs_out ${out_lib_dir}/all_libs.a)
get_property(app_libraries GLOBAL PROPERTY app_libraries)

list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_dev.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_adc.a)
if(EXISTS ${SOURCE_TOP_DIR}/components/ql-kernel/libs/liboql_pin_init.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/liboql_pin_init.a)
endif()
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_gpio.a)
if(EXISTS ${SOURCE_TOP_DIR}/components/ql-kernel/libs/liboql_api_power.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/liboql_api_power.a)
endif()
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_power.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_bsp.a)

if(CONFIG_QUEC_PROJECT_FEATURE_FILE_AT)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_file_at.a)
endif()
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_at.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_urc.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libquectel.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_common.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_utils.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_rtos.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_osi.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libq_modem.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_sim.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_nw.a)
if(CONFIG_QUEC_PROJECT_FEATURE_USBNET)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libq_usbnet.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_usbnet.a)
endif()
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_net.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_datacall.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_fs.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_file.a)

if(CONFIG_QUEC_PROJECT_FEATURE_FILE_ZIP)
# FILE_ZIP depends on FILE
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_zip.a)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_zip.a)
endif()

if(CONFIG_QUEC_PROJECT_FEATURE_RTC)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_rtc.a)
endif()
if(CONFIG_QUEC_PROJECT_FEATURE_VIRT_AT)
list(APPEND app_libraries ${SOURCE_TOP_DIR}/components/ql-kernel/libs/libql_api_virt_at.a)
endif()
```

基础库，不可裁剪

可裁剪库

图 13：内核中可裁剪库

可参考如下修改配置文件 *target.config* 进行功能裁剪：

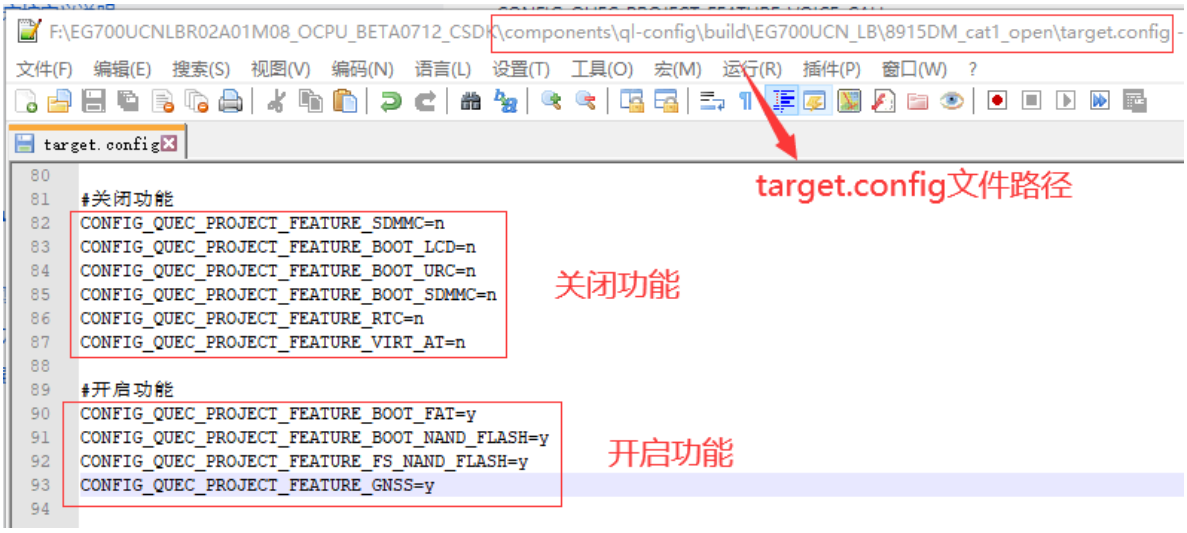


图 14：内核功能裁剪

内核中可裁剪库列表如下所示：

表 3：内核中可裁剪库

功能宏	功能说明	依赖项
CONFIG_QUEC_PROJECT_FEATURE_USBNET	USB 网卡	-
CONFIG_QUEC_PROJECT_FEATURE_GNSS	GNSS	CONFIG_QUEC_PROJECT_FEATURE_UART
CONFIG_QUEC_PROJECT_FEATURE_RTC	实时时钟	-
CONFIG_QUEC_PROJECT_FEATURE_VIRT_AT	虚拟 AT	-
CONFIG_QUEC_PROJECT_FEATURE_FOTA	FOTA 升级	-
CONFIG_QUEC_PROJECT_FEATURE_SMS	短信	CONFIG_QUEC_PROJECT_FEATURE_HTTP
CONFIG_QUEC_PROJECT_FEATURE_VOICE_CALL	语音通话	-
CONFIG_QUEC_PROJECT_FEATURE_FTP	文件传输协议	-
CONFIG_QUEC_PROJECT_FEATURE_HTTP	超文本传输协议	-
CONFIG_QUEC_PROJECT_FEATURE_SSH2	安全外壳协议	-

CONFIG_QUEC_PROJECT_FEATURE_MQTT	消息传输协议	-
CONFIG_QUEC_PROJECT_FEATURE_PING	因特网包探索器	-
CONFIG_QUEC_PROJECT_FEATURE_NTP	网络时间协议	-
CONFIG_QUEC_PROJECT_FEATURE_MMS	多媒体短信服务	-
CONFIG_QUEC_PROJECT_FEATURE_LBS	位置服务	CONFIG_QUEC_PROJECT_FEATURE_HTTP
CONFIG_QUEC_PROJECT_FEATURE_UART	串口	-
CONFIG_QUEC_PROJECT_FEATURE_I2C	I2C	-
CONFIG_QUEC_PROJECT_FEATURE_KEYPAD	矩阵键盘	-
CONFIG_QUEC_PROJECT_FEATURE_LCD	LCD	-
CONFIG_QUEC_PROJECT_FEATURE_BT	蓝牙	-
CONFIG_QUEC_PROJECT_FEATURE_BT_HFP	蓝牙 HFP	CONFIG_QUEC_PROJECT_FEATURE_BT
CONFIG_QUEC_PROJECT_FEATURE_BT_A2DP_AVRCP	蓝牙音频	CONFIG_QUEC_PROJECT_FEATURE_BT
CONFIG_QUEC_PROJECT_FEATURE_BLE_GATT	低功耗蓝牙	CONFIG_QUEC_PROJECT_FEATURE_BT
CONFIG_QUEC_PROJECT_FEATURE_CAMERA	相机	-
CONFIG_QUEC_PROJECT_FEATURE_SPI	SPI	-
CONFIG_QUEC_PROJECT_FEATURE_SPI_FLASH	4 线 SPI Flash	CONFIG_QUEC_PROJECT_FEATURE_SPI
CONFIG_QUEC_PROJECT_FEATURE_SPI_NOR_FLASH	4 线 SPI NOR Flash	CONFIG_QUEC_PROJECT_FEATURE_SPI CONFIG_QUEC_PROJECT_FEATURE_SPI_FLASH
CONFIG_QUEC_PROJECT_FEATURE_SPI_NAND_FLASH	4 线 SPI NAND Flash	CONFIG_QUEC_PROJECT_FEATURE_SPI CONFIG_QUEC_PROJECT_FEATURE_SPI_FLASH
CONFIG_QUEC_PROJECT_FEATURE_SDMMC	SDMMC	-
CONFIG_QUEC_PROJECT_FEATURE_PBK	电话本	-

CONFIG_QUEC_PROJECT_FEATURE_FILE_ZIP	文件压缩	-
CONFIG_QUEC_PROJECT_FEATURE_FS_NAND_FLASH	4 线 SPI NAND Flash（文件系统）	CONFIG_QUEC_PROJECT_FEATURE_SPI CONFIG_QUEC_PROJECT_FEATURE_SPI_FLASH CONFIG_QUEC_PROJECT_FEATURE_SPI_NAND_FLASH
CONFIG_QUEC_PROJECT_FEATURE_SPI6_EXT_NOR	6 线 SPI NOR FLASH（文件系统）	-
CONFIG_QUEC_PROJECT_FEATURE_FILE_AT	文件系统 AT 命令	-
CONFIG_QUEC_PROJECT_FEATURE_CLOUDOTA	Cloud OTA	CONFIG_QUEC_PROJECT_FEATURE_HTTP
CONFIG_QUEC_PROJECT_FEATURE_VOLTE	VoLTE	-
CONFIG_QUEC_PROJECT_FEATURE_AUDIO	音频	-
CONFIG_QUEC_PROJECT_FEATURE_USB	USB	-
CONFIG_QUEC_PROJECT_FEATURE_LEDCFG	LED&PWM 功能	-
CONFIG_QUEC_PROJECT_FEATURE_WIFISCAN	Wi-Fi Scan	-

6.2. 裁剪 BootLoader 功能

模块 BootLoader 中含以下功能可供裁剪。如无需该功能可在 *target.config* 中关闭；如需要该功能可在 *target.config* 中打开，功能打开后还需在 *boot2_start_8910.c* 文件中调用功能初始化函数。裁剪后需进行全新编译。

表 4：BootLoader 可裁剪库

功能宏	功能说明	依赖项
CONFIG_QUEC_PROJECT_FEATURE_BOOT_LCD	启动 LCD	CONFIG_QUEC_PROJECT_FEATURE_LCD
CONFIG_QUEC_PROJECT_FEATURE_BOOT_FAT	启动 FAT 文件系统	-
CONFIG_QUEC_PROJECT_FEATURE_BOOT_SDMMC	启动 SDMMC	CONFIG_QUEC_PROJECT_FEATURE_BOOT_FAT

CONFIG_QUEC_PROJECT_FEATURE_BOOT_NAND_FLASH	启动 NAND Flash	CONFIG_QUEC_PROJECT_FEATURE_BOOT_FAT

```
84:
85: #ifdef CONFIG_QUEC_PROJECT_FEATURE_FOTA
86: extern quec_boot_fs_type_e quec_boot_fs_type;
87: static void quec_boot_ext_flash_init()
88: {
89:     switch(quec_boot_fs_type)
90:     {
91:         case QUEC_BOOT_SFFS_EXT:
92: #ifdef CONFIG_QUEC_PROJECT_FEATURE_SPI6_EXT_NOR
93:          quec_boot_spi6_ext_norflash_init();
94: #endif
95:          break;
96:
97:         case QUEC_BOOT_FAT_SDMMC:
98: #ifdef CONFIG_QUEC_PROJECT_FEATURE_BOOT_SDMMC
99:          quec_boot_sdmmc_init();
100: #endif
101:          break;
102:
103:         case QUEC_BOOT_FAT_EXNAND_FLASH:
104: #ifdef CONFIG_QUEC_PROJECT_FEATURE_BOOT_NAND_FLASH
105:          //quec_boot_nand_init(QL_BOOT_SPI_PORT_1);
106: #endif
107:          break;
108:
109:         default:
110:         break;
111:     } « end switch quec_boot_fs_type »
112: } « end quec_boot_ext_flash_init »
113: #endif
114:
```

功能初始化函数

图 15：功能初始化函数

7 预置文件

预置文件指在固件包中打包的指定的初始文件，用于在固件烧录时直接写入模块文件系统中。预置文件会占用文件系统空间，目前单个预置文件最大大小限制为 512 KB。若文件大小超过 512 KB，下载文件时会跳过该文件。

7.1. 启用预置文件打包

在 *ql_app_feature_config.cmake* 文件中，用户可通过 QL_APP_PACK_FILE 配置是否进行预置文件的打包。具体配置如下图：

```
##########################################################################################
# Quectel open sdk package config
##########################################################################################
if (QL_APP_FEATURE_GNSS)
option(QL_APP_PACK_FILE "Enable pack file to firmware package" ON)  预置文件开关
endif()
if (QL_APP_PACK_FILE)
#set(QL_APP_PACK_FILE_JSON_PATH components/ql-config/download/prepack/example/prepack.json)
set(QL_APP_PACK_FILE_JSON_PATH components/ql-config/download/prepack/ql_prepack.json)
endif()              需被打包的预置文件路径
message(STATUS "QL_APP_PACK_FILE ${QL_APP_PACK_FILE} @ ${QL_APP_PACK_FILE_JSON_PATH}")
```

图 16：预置打包编译配置

备注

SDK 中该配置受 GNSS 功能影响，若 GNSS 功能开启，模块默认将 GNSS 芯片固件（目前约 238 KB）作为预置文件打包至应用程序版本固件中。ECx00U 系列和 EGx00U-CN QuecOpen 模块 GNSS 功能的支持情况，可查看各模块对应的硬件设计手册文档。

如需预置其他功能文件，则增加对应的功能开关。例如，除了例如音频功能预置音频文件，可增加音频的功能开关，如下所示：

```
##########################################################################
# Quectel open sdk package config
##########################################################################
if (QL_APP_FEATURE_GNSS OR QL_APP_FEATURE_AUDIO)
option(QL_APP_PACK_FILE "Enable pack file to firmware package" ON)
endif()
```

图 17：增加需要预置文件的功能开关

7.2. 配置预置文件

ql_app_feature_config.cmake 文件中 QL_APP_PACK_FILE_JSON_PATH 用来指定预置文件的具体配置所在路径。编译脚本根据指定的 *json* 文件打包预置文件。CSDK 的 *json* 文件指定了 GNSS 芯片固件的所在位置，采用相对路径表示，即与 json 文件在相同路径下。用户可以参考此格式在下图中*"files"*节点内进行添加修改，注意不要预置太多文件以避免占用过多文件系统空间。

```
{
    "_comment": [
        "This is the configuration file to pre-pack files to pac.",
        "It contains a list of _files_, and for each file,",
        "_file_ is the absolute path in target, and _local_file_",
        "is the path in host. When _local_file_ is relative path,",
        "it is related to the directory of this configuration file.",
        "Also @SOURCE_TOP_DIR@ and @BINARY_TOP_DIR@ will be substitued",
        "to the absolte of current build.",
        "",
        "When it is empty, PREPACK won't be inserted into pac."
    ],
    "files": [
        {
            "file": "/user/boot",
            "local_file": "bootloader_r3.0.0_build2817_uartboot_4468750.pkg"
        },
        {
            "file": "/user/firm",
            "local_file": "UC6226CI-R3.2.0.12Build6815_mfg.pkg"
        }
    ]
}
```

指定文件所在路径和文件名

文件所在路径

图 18：预置打包配置文件

预置文件烧录到模块中的位置一般为 *user* 目录，便于通过模块 FILE 相关 API 进行查询。写入的文件路径和文件名的总长度不可超过 192 字节。

7.3. 下载或升级预置文件

预置文件可通过 FOTA 进行下载、升级或者删除：

- 将旧版本的预置文件升级到新版本的预置文件；
- 若旧版本中无预置文件，可通过 FOTA 升级增加预置文件；
- 若旧版本中有预置文件，新版本中无预置文件，FOTA 升级后将删除预置文件；
- 若进行 FOTA 固件升级时需忽略预置文件的升级，则需修改制作 FOTA 包的 xml 配置文件。在 xml 配置文件中将<paccpio id="PREPACK" method="diff"> </paccpio>中 *method* 的值"diff"修改为"ignore"，如下图所示。请参考***文档 [2]***。

```
</pacnv>
<paccpio id="PREPACK" method="diff">
    <!--
        <file name="some_file_name" method="ignore"/>
    -->
</paccpio>
```

图 19：FOTA 升级忽略预置文件

8 调整 Flash 分区

移远通信 ECx00U 系列和 EGx00U-CN QuecOpen 支持调整内置 flash 分区，可调整 Kernel、应用程序、文件系统分区大小。具体实现方式请参考*文档 [3]*。

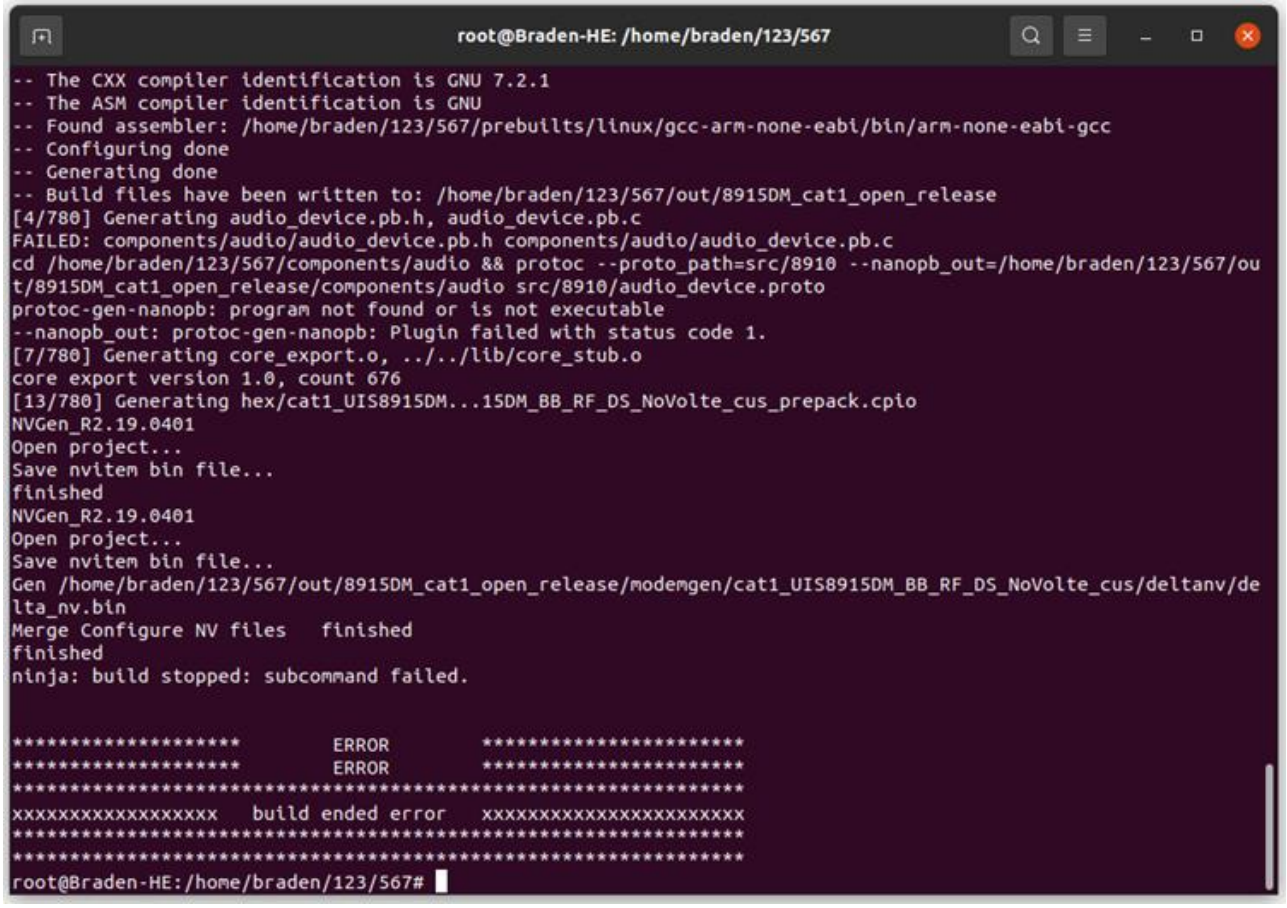
9 固件烧录

移远通信 ECx00U 系列和 EGx00U-CN QuecOpen 模块支持通过 QFlash 工具进行固件烧录。有关工具使用方法，请参考***文档 [1]***。

10 常见问题

10.1. Linux 下常见编译报错

1. 若出现下图所示的错误，表示主机系统未安装需求软件。详情请参考***第 3.1 章***。重新安装即可。



2. 若出现错误提示“dtools：error while loading shared libraries: libicui18n.so.55:canot openShared object file:No such file or directory. Ninja:build stopped:subcommand failed”，表示 dtools 缺少组件，依次执行如下命令安装组件即可：

```
wget http://security.ubuntu.com/ubuntu/pool/main/i/icu/libicu55_55.1-7_amd64.deb
sudo dpkg -i libicu55_55.1-7_amd64.deb
```

```
910/audio_device.proto
/bin/sh: 1: protoc: not found
[7/862] Generating core_export.o, ../../lib/core_stub.o
FAILED: components/apploader/core_export.o lib/core_stub.o
cd /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_open_release/components/apploader && dt
ools expgen -p 8910 --export /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_open_release/
components/apploader/core_export.o --stub /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_
open_release/lib/core_stub.o /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_open_release/
components/apploader/CMakeFiles/export_list_gen.dir/src/core_export.list.obj
dtools: error while loading shared libraries: libicui18n.so.55: cannot open shared object file: No such file or d
irectory
[13/862] Generating hex/cat1_UIS8915DM...15DM_BB_RF_SS_NoVolte_cus_prepack.cpio
NVGen_R2.19.0401
Open project...
Save nvitem bin file...
finished
NVGen_R2.19.0401
Open project...
Save nvitem bin file...
[Warning] File: /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_open_release/modemgen/cat1
_UIS8915DM_BB_RF_SS_NoVolte_cus/deltanv/delta.nv, Not find nv_ver_flag
Gen /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_open_release/modemgen/cat1_UIS8915DM_B
B_RF_SS_NoVolte_cus/deltanv/delta_nv.bin
Merge Configure NV files   finished
finished
dtools: error while loading shared libraries: libicui18n.so.55: cannot open shared object file: No such file or d
irectory
ninja: build stopped: subcommand failed.


********************        ERROR        **********************
********************        ERROR        **********************
************************************************************
xxxxxxxxxxxxxxxxx   build ended error   xxxxxxxxxxxxxxxxxxxxxx
************************************************************
```

11 附录 参考文档及术语缩写

表 5：参考文档

参考文档
[1] Quectel_QFlash_用户指导
[2] Quectel_ECx00U&EGx00U 系列_QuecOpen_FOTA_API_参考手册
[3] Quectel_ECx00U&EGx00U 系列_QuecOpen_内置 Flash 分区调整指导

表 6：术语缩写

缩写	英文全称	中文全称
ADC	Analog-to-Digital Converter	模数转换器
API	Application Programming Interface	应用程序接口
App	Application	应用程序
BT	Bluetooth	蓝牙
BLE	Bluetooth Low Energy	蓝牙低功耗
FOTA	Firmware Over-The-Air	固件空中升级
FTP	File Transfer Protocol	文件传输协议
GCC	GNU Compiler Collection	GNU 编译器套件
GNSS	Global Navigation Satellite System	全球导航卫星系统
GPIO	General-Purpose Input/Output	通用型输入/输出
HFP	Hands-free Profile	免提规格
HTTP	Hypertext Transfer Protocol	超文本传输协议
I2C	Inter-Integrated Circuit	集成电路总线

IoT	Internet of Things	物联网
LCD	Liquid Crystal Display	液晶显示屏
LED	Light Emitting Diode	发光二极管
MQTT	Message Queuing Telemetry Transport	消息队列遥测传输
OSI	Open System Interconnection Reference Model	开放式系统互联通信参考模型
PWM	Pulse Width Modulation	脉冲宽度调制
RTOS	Real-Time Operating System	实时操作系统
SDMMC	Secure Digital/MultiMediaCard	数字安全记忆卡/多媒体卡
SPI	Serial Peripheral Interface	串行外设接口
USB	Universal Serial Bus	通用串行总线
VoLTE	Voice (voice calls) over LTE	长期演进语音承载
(U)SIM	(Universal) Subscriber Identity Module	（通用）用户身份识别模块
Wi-Fi	Wireless Fidelity	无线保真（技术）