

EC200U Series QuecOpen CSDK Quick Start Guide

LTE Standard Module Series

Version: 1.0.0

Date: 2021-12-09

Status: Preliminary



At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.

Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China

Tel: +86 21 5108 6236

Email: info@quectel.com

Or our local offices. For more information, please visit:

<http://www.quectel.com/support/sales.htm>.

For technical support, or to report documentation errors, please visit:

<http://www.quectel.com/support/technical.htm>.

Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

- a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
- b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
- c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
- d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

Copyright © Quectel Wireless Solutions Co., Ltd. 2021. All rights reserved.

About the Document

Revision History

Version	Date	Author	Description
-	2021-12-09	Neo KONG	Creation of the document
1.0.0	2021-12-09	Neo KONG	Preliminary

Contents

About the Document.....	3
Contents	4
Table Index	5
Figure Index	6
1 Introduction	7
2 CSDK Directory Structure	8
3 Compilation Environment Setup	10
3.1. Host System	10
3.2. Application Compilation Environment	10
4 Application Development	11
4.1. Application Demo Description	11
4.2. Add Application Feature Component	12
4.2.1. Create a Directory of Application Feature Component	12
4.2.2. Create an Application Thread	12
4.2.3. Modify Configuration File	13
4.2.4. Add Compilation Task	13
4.2.5. Link Application Feature Component	14
4.2.6. Feature Component Compilation Switch	14
5 Application Compilation	17
5.1. Compilation Description	17
5.2. Compilation Procedure	17
5.3. Compilation Result	19
5.4. Compiled File Clearance	19
6 Feature Removal	20
6.1. Feature Removal in Kernel	20
6.2. Feature Removal in BootLoader	23
7 Preset File	25
7.1. Enable Preset File Package	25
7.2. Preset File Configuration	26
7.3. Preset File Download and Upgrade	26
8 Flash Partition Adjustment	28
9 Firmware Burning	29
10 FAQ	30
10.1. Common Compilation Error in Linux	30
11 Appendix References	32

Table Index

Table 1: CSDK Directory Structure	8
Table 2: The Removable Libraries in Kernel	21
Table 3: The Removable Libraries in BootLoader	23
Table 4: Related Documents	32
Table 5: Terms and Abbreviations	32

Figure Index

Figure 1: CSDK Directory Structure	9
Figure 2: Application Entry	11
Figure 3: Directory Example of Application Feature Component	12
Figure 4: Create Feature Component Application Thread	12
Figure 5: Modify Configuration File	13
Figure 6: Add Compilation Tasks.....	13
Figure 7: Link New Application Feature Component and Add Option Switch Control.....	14
Figure 8: Configure Compilation Switch.....	15
Figure 9: Add Macro Control	16
Figure 10: CSDK Original Firmware	17
Figure 11: Compile Application.....	18
Figure 12: Compilation Results	19
Figure 13: Removable Libraries in Kernel.....	20
Figure 14: Feature Removal in Kernel	21
Figure 15: Feature Initialization Function.....	24
Figure 16: Preset File Package Compilation Configuration.....	25
Figure 17: Add a Feature Switch for Preset File	25
Figure 18: Preset Package Configuration File	26
Figure 19: Ignore the Preset File in FOTA	27

1 Introduction

Quectel EC200U series module supports QuecOpen[®] solution. QuecOpen[®] is an embedded development platform based on RTOS, which is intended to simplify the design and development of IoT applications.

This document introduces CSDK directory structure, compilation environment setup, application development and compilation process, feature removal, partition adjustment, firmware download and FAQ of EC200U series module in QuecOpen[®] solution.

2 CSDK Directory Structure

The directory may differ depending on the actual released versions in the QuecOpen CSDK. The QuecOpen CSDK directory is shown in **Figure 1**, with details in the following table.

Table 1: CSDK Directory Structure

Directory Name	Description
<i>cmake, prebuilts, tools</i>	Includes compilation tools, configuration files and compilation scripts.
<i>components/appstart</i>	Includes source files of Kernel start program.
<i>components/bootloader</i>	Includes source files of boot loader.
<i>components/hal</i>	Includes partitions and Flash configuration files.
<i>components/ql-config</i>	Includes the default Kernel firmware, original application firmware and preset files of the current version.
<i>components/ql-kernel</i>	Includes all open API header files and library files.
<i>components/ql-application</i>	Includes application reference routines and demos that implement various features.
<i>components/libs</i> <i>components/net</i> <i>components/newlib</i>	Includes header files and library files required by the application.
<i>build_all.bat,</i> <i>build_all.sh</i>	<ul style="list-style-type: none"> ● Windows: Compile the application in Windows CLI according to the configured parameters. See Chapter 5 for details. ● Linux: Compile the application in Linux CLI according to the configured parameters. Administrator mode is required during compilation in Linux. And the script should be executed by bash. See Chapter 5 for details.
<i>CmakeLists.txt</i>	Top-level compilation configuration file.

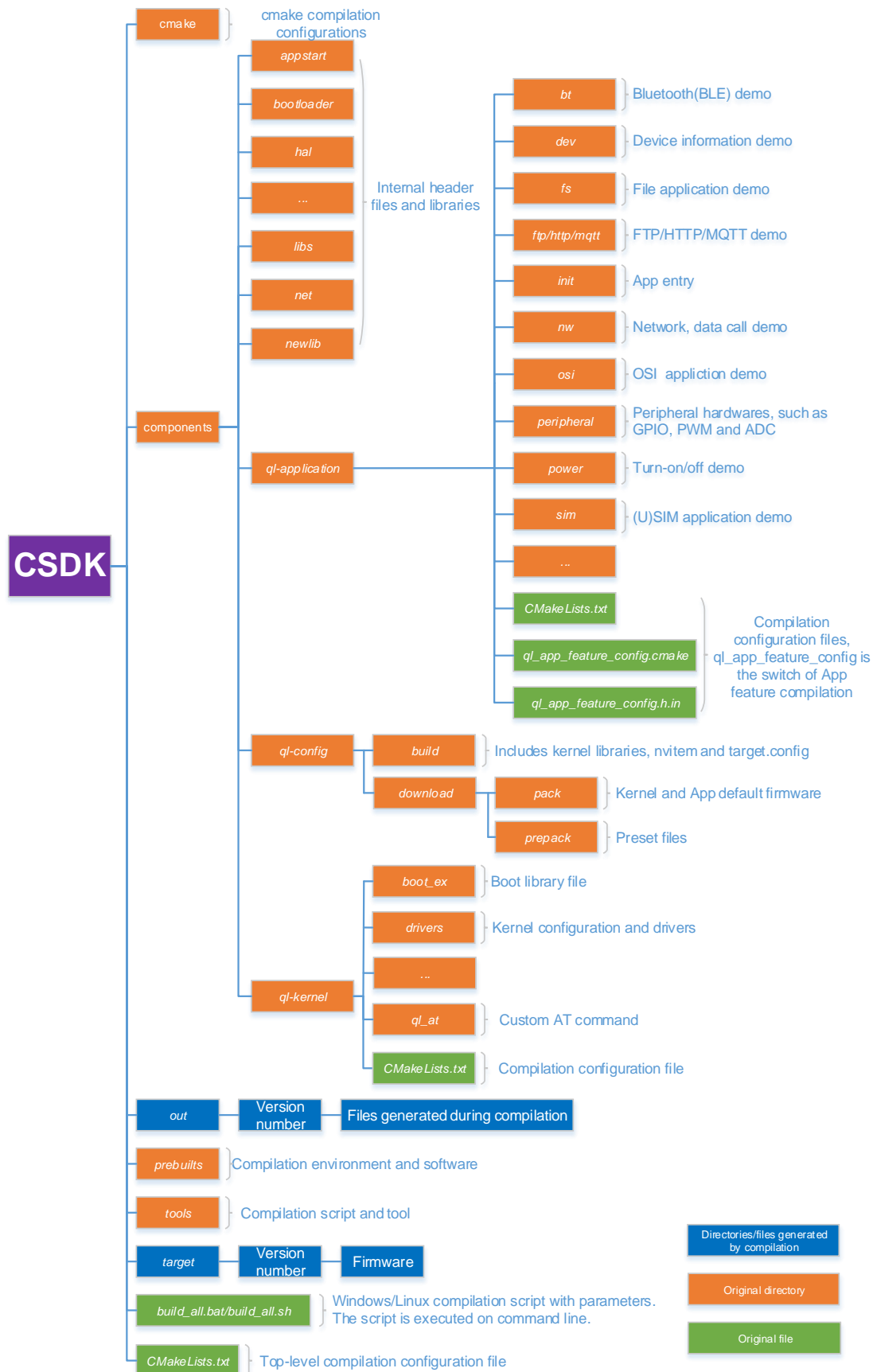


Figure 1: CSDK Directory Structure

3 Compilation Environment Setup

3.1. Host System

If the host system running the compilation environment is Windows, ensure that the host system is 32-bit or 64-bit Windows 7/10 and install the following software:

- Visual C++ Redistributable for Visual Studio 2015 x86 or above (32-bit Windows)
- Visual C++ Redistributable for Visual Studio 2015 x64 or above (64-bit Windows)

If the host system running the compilation environment is Linux, ensure that the operating system is Ubuntu 16.04 or Ubuntu 20.04 and the language is Python 3.5 or above, and execute the following commands to install the required components:

```
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. Application Compilation Environment

The application compilation tool chain of EC200U series QuecOpen module adopts gcc-arm-none-eabi, which is the GCC tool chain of ARM bare metal system. The gcc-arm-none-eabi with version 7.2.1 is integrated in the QuecOpen CSDK in the directory of *prebuilts*.

When compiling an application, only the gcc-arm-none-eabi tool chain integrated in the QuecOpen CSDK is supported and the gcc-arm-none-eabi tool chain installed on the host system is not supported.

4 Application Development

4.1. Application Demo Description

QuecOpen CSDK provides application demos to which users can refer in the directory of *components\ql-application* in CSDK for application development.

The application entry, *appimg_enter()*, is contained in the *ql_init.c* file in the directory of *components\ql-application\init* in the QuecOpen CSDK. Start an initialization thread in this function, and then call the initialization interface of each feature component in this initialization thread. Each feature component is divided according to folders, and you can refer to the following figure to add feature component.

```
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;
}
```

Figure 2: Application Entry

4.2. Add Application Feature Component

This chapter takes the audio Application provided in QuecOpen CSDK as an example to introduce how to add an application feature component.

4.2.1. Create a Directory of Application Feature Component

First, create a feature directory under the directory of *components/ql-application* in the QuecOpen CSDK. Then store the source files, header files, and compilation configuration file *CMakeLists.txt* in the feature directory. Taking an audio application as an example, the directory structure is as follows and the *inc* folder is used to store header files:

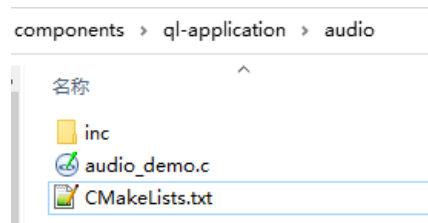


Figure 3: Directory Example of Application Feature Component

4.2.2. Create an Application Thread

Use *ql_rtos_task_create()* to create the application thread of the feature component in *ql_audio_app_init()*, the initialization function of the feature component source code file, as shown in the following example:

```
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");
    }
}
```

Figure 4: Create Feature Component Application Thread

4.2.3. Modify Configuration File

Modify the application configuration file, *components\ql-application\audio\CMakeLists.txt*, as needed, with the following three modifications:

```
set(target ql_app_audio) Library name

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(${target} PUBLIC inc) Header file path
#target_link_libraries(${target} PRIVATE ql_api_common)

target_sources(${target} PRIVATE
    audio_demo.c Source file list, supporting relative path
)

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

Figure 5: Modify Configuration File

4.2.4. Add Compilation Task

Modify the configuration file, *components\ql-application\CMakeLists.txt* to add the application feature component directory to the file so as to compile the directory when compiling the firmware package, as shown in the following example:

```
# 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()
```

Figure 6: Add Compilation Tasks

4.2.5. Link Application Feature Component

Modify the configuration file *CMakeLists.txt* in the directory of *components/ql-application/init* to add the name of the library that links the new application feature components. If a feature component compilation switch is added for the application, the option switch control needs to be added to the file *CMakeLists.txt* as well, as shown in the following example:

```

19 if(CONFIG_APPING_LOAD_FLASH)
20     set(target ${QL_APP_BUILD_VER})
21     add_appimg_flash_ql_example(${target} ql_init.c)
22
23     target_link_libraries(${target} PRIVATE ql_app_nw ql_app_peripheral ql_app_osi ql_app_dev ql_app_sim ql_app_power)
24     if(QL_APP_FEATURE_FILE_ZIP)
25         target_link_libraries(${target} PRIVATE ql_app_zip)
26     endif()
27     if(QL_APP_FEATURE_FTP)
28         target_link_libraries(${target} PRIVATE ql_app_ftp)
29     endif()
30     if(QL_APP_FEATURE_HTTP)
31         target_link_libraries(${target} PRIVATE ql_app_http)
32     endif()
33     if(QL_APP_FEATURE_MMS)
34         target_link_libraries(${target} PRIVATE ql_app_mms)
35     endif()
36     if(QL_APP_FEATURE_MQTT)
37         target_link_libraries(${target} PRIVATE ql_app_mqtt)
38     endif()
39     if(QL_APP_FEATURE_SSL)
40         target_link_libraries(${target} PRIVATE ql_app_ssl)
41     endif()
42     if(QL_APP_FEATURE_PING)
43         target_link_libraries(${target} PRIVATE ql_app_ping)
44     endif()
45     if(QL_APP_FEATURE_NTP)
46         target_link_libraries(${target} PRIVATE ql_app_ntp)
47     endif()
48     if(QL_APP_FEATURE_LBS)
49         target_link_libraries(${target} PRIVATE ql_app_lbs)
50     endif()
51
52     if(QL_APP_FEATURE_CTSREG)
53         target_link_libraries(${target} PRIVATE ql_app_ctsreg)
54     endif()
55
56     if(QL_APP_FEATURE_SOCKET)
57         target_link_libraries(${target} PRIVATE ql_app_socket)
58     endif()
59
60     if(QL_APP_FEATURE_AUDIO)
61         target_link_libraries(${target} PRIVATE ql_app_audio)
62         if(QL_APP_FEATURE_TTS)
63             add_library(ql_tts_api STATIC IMPORTED)
64             set_target_properties(ql_tts_api PROPERTIES IMPORTED_LOCATION ${SOURCE_TOP_DIR}/components/newlib/armca5/libql_api_tts.a)
65             target_link_libraries(${target} PRIVATE ql_app_tts ql_tts_api ${libm_file_name})
66         endif()
67     endif()

```

Figure 7: Link New Application Feature Component and Add Option Switch Control

Compile the firmware according to **Chapter 5** when the application feature component is added and the relevant configuration files are configured.

4.2.6. Feature Component Compilation Switch

QuecOpen CSDK provides *ql_app_feature_config.cmake* and *ql_app_feature_config.h.in* under the directory of *components/ql-application* to facilitate App side to configure or remove features. *ql_app_feature_config.cmake* configures the feature components to be compiled by the compilation script.;

ql_app_feature_config.h.in generates macro used to enable component. The generated macros will be used in the code. Configuration or removal of required feature requires simultaneous modifications in both files (see **Chapter 6** for details on the feature removal).

The content of *ql_app_feature_config.cmake* is shown in the following figure. If a feature component is set to "ON", it indicates that the feature component will be compiled when the compilation script is executed; If it is set to "OFF", it indicates that the feature component will not be compiled when the compilation script is executed.

```
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}")
```

Figure 8: Configure Compilation Switch

The content of *ql_app_feature_config.h.in* is shown in the following figure. After adding the compilation switch of feature components in *ql_app_feature_config.cmake*, refer to the following figure to add a macro

of the corresponding feature components:

```
#ifndef _QL_APP_FEATURE_CONFIG_H_
#define _QL_APP_FEATURE_CONFIG_H_

// @AUTO_GENERATION_NOTICE@

/**
 * whether to enable APP feature FTP
 */
#define QL_APP_FEATURE_FTP

/**
 * whether to enable APP feature HTTP
 */
#define QL_APP_FEATURE_HTTP

/**
 * whether to enable APP feature MQTT
 */
#define QL_APP_FEATURE_MQTT

/**
 * whether to enable APP feature SSL
 */
#define QL_APP_FEATURE_SSL

/**
 * whether to enable APP feature FILE
 */
#define QL_APP_FEATURE_FILE

/**
 * whether to enable APP feature AUDIO
 */
#define QL_APP_FEATURE_AUDIO

/**
 * whether to enable APP feature TTS
 * if TTS is enabled, you need enable AUDIO too
 */
#define QL_APP_FEATURE_AUDIO_TTS
```

Figure 9: Add Macro Control

5 Application Compilation

5.1. Compilation Description

In the directory of *components\ql-config\download\pack* in QuecOpen CSDK, a folder named after a module model is included, which holds the default Kernel firmware of the current module version, the original application firmware stored by Quectel, and the relevant *.map* and *.elf* files, as well as the merged firmware with Kernel and application.

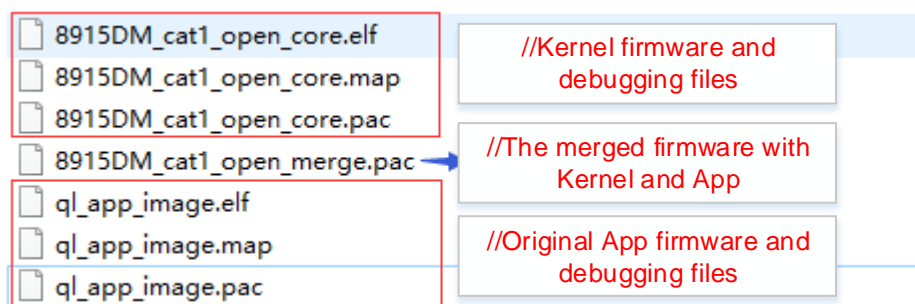


Figure 10: CSDK Original Firmware

The source code of the application is located in directory of *components\ql-application* and is compiled by using the compilation script file *build_all.bat* (the host system is Windows) or *build_all.sh* (the host system is Linux) in the root directory of QuecOpen CSDK. Both script files use the same command parameters, but *build_all.sh* should be executed by **bash** command, that is, **bash build_all.sh**. The following sections take Windows host system as an example to introduce the application compilation process.

5.2. Compilation Procedure

1. Open CLI or PowerShell (only Windows 10 supports PowerShell) and execute **build_all.bat -h** to query the usage. The common commands are as follows:

- Compile command: **build_all <r/new> <Project> <Version>[VOLTE] [DSIM] [debug/release] <r/new>**
 Compilation type. r indicates incremental compilation and w indicates new

compilation.

- <Project>** Quectel module model in use. This parameter must be the same as the folder name in the directory of *components\ql-config\download\pack*.
- <Version>** The name of the compiled application firmware, which can be customized.
- [VOLTE]** VoLTE feature (optional). VOLTE or NOVOLTE. Default: NOVOLTE.
- [DSIM]** Single/dual card (optional). SINGLESIM: single card. DOUBLESIM: dual card. Default: SINGLESIM.
- [debug/release]** Version (optional). A debug firmware is generated if you choose debug and an official releasable firmware is generated if you choose release. Default: release.

- Clears files generated during the previous compilation: **build_all clean**.

2. In CLI or PowerShell (only Windows 10 supports PowerShell), enter QuecOpen CSDK root directory and execute **build_all <[r/new]> <Project> <Version>[VOLTE] [DSIM] [debug/release]** to start compiling.

After the command is executed, the compilation script first checks whether the directory of *components\ql-config\download\pack* in QuecOpen CSDK contains a directory named **<Project>**, and if so, the compilation script continues compiling; If not, an error is reported and the compilation fails. The execution results are shown in the following figure:

```
PS E:\EC200UCNAA02A04V03M08_OCPSDK> ./build_all new EC200UCN_AA appimage

C:\Program Files\VanDyke Software\Clients\C:\Program Files\ARM\bin\win_32-pentium\C:\Perl\bin.C:\WINDOWS\system32.C:\W
INDOWS.C:\WINDOWS\System32\Wbem.C:\WINDOWS\System32\WindowsPowerShell\v1.0\C:\WINDOWS\System32\OpenSSH.C:\Program Files
\ARM\Java\JRE\150\02\win_32-pentium\jre1.5.0_02\bin.C:\Program Files\ARM\RD\Deprecated\1.3.1\1\windows.C:\Program Files
\ARM\RVCT\Programs\3.1\569\win_32-pentium.C:\Program Files\ARM\RVCT\Core\3.1\881\win_32-pentium\bin.C:\Program Files\ARM
\IDEs\Eclipse\Distribution\1.1\32\win_32-pentium\eclipse.C:\Program Files\ARM\Utilities\FLEXlm\10.8.5.0\1\win_32-pentium
.C:\Program Files\Perforce\C:\Users\neo.kong\AppData\Local\Microsoft\WindowsApps;.c:\program files\esafenet\cobra docgu
ard client

cleaning...

cleaning done

components\ql-config\build\EC200UCN_AA\ql_libs\libql_at.a

E:\EC200UCNAA02A04V03M08_OCPSDK\tools\win32.E:\EC200UCNAA02A04V03M08_OCPSDK\tools.E:\EC200UCNAA02A04V03M08_OCPS
DK\prebuilds\win32\nanopb.E:\EC200UCNAA02A04V03M08_OCPSDK\prebuilds\win32\gcc-rv32-elf\bin.E:\EC200UCNAA02A04V03
M08_OCPSDK\prebuilds\win32\gcc-mips-rda-elf\bin.E:\EC200UCNAA02A04V03M08_OCPSDK\prebuilds\win32\gcc-arm-none-eabi
\bin.E:\EC200UCNAA02A04V03M08_OCPSDK\prebuilds\win32\python3.E:\EC200UCNAA02A04V03M08_OCPSDK\prebuilds\win32\cm
ake\bin.E:\EC200UCNAA02A04V03M08_OCPSDK\prebuilds\win32\bin.C:\Program Files\VanDyke Software\Clients\C:\Program Fil
es\ARM\bin\win_32-pentium.C:\Perl\bin.C:\WINDOWS\system32.C:\WINDOWS\System32\Wbem.C:\WINDOWS\System32\Windows
PowerShell\v1.0.C:\WINDOWS\System32\OpenSSH.C:\Program Files\ARM\Java\JRE\150\02\win_32-pentium\jre1.5.0_02\bin.C:\P
rogram Files\ARM\RD\Deprecated\1.3.1\1\windows.C:\Program Files\ARM\RVCT\Programs\3.1\569\win_32-pentium.C:\Program Fil
es\ARM\RVCT\Core\3.1\881\win_32-pentium\bin.C:\Program Files\ARM\IDEs\Eclipse\Distribution\1.1\32\win_32-pentium\eclipse.C
:\Program Files\ARM\Utilities\FLEXlm\10.8.5.0\1\win_32-pentium.C:\Program Files\Perforce\C:\Users\neo.kong\AppData\Loca
l\Microsoft\WindowsApps;.c:\program files\esafenet\cobra docguard client

target dir: E:\EC200UCNAA02A04V03M08_OCPSDK\components\ql-config\build\EC200UCN_AA/

-- Could NOT find Git (missing: GIT_EXECUTABLE)
Curr Proj: EC200UCN_AA, QL_CSDK_BUILD_ON

cat1_UIS8915DM_BB_RF_SS_NoVolte_cus, components\hal\config\8910\partinfo_8910_8m_opencpu_novolte.json, 0x980000, 0x6800
0, off, off, off

-- QL_APP_FEATURE_FTP ON
-- QL_APP_FEATURE_HTTP ON
-- FEATURE MMS is disabled at core!
-- QL_APP_FEATURE_MMS OFF
-- QL_APP_FEATURE_MQTT ON
-- QL_APP_FEATURE_SSL ON
-- QL_APP_FEATURE_PING ON
-- QL_APP_FEATURE_NTP ON
-- QL_APP_FEATURE_ALI_LINKSDK ON
-- QL_APP_FEATURE_QCLOUD_IOT ON
-- QL_APP_FEATURE_LBS ON
-- QL_APP_FEATURE_SOCKET ON
-- QL_APP_FEATURE_CTSREG ON
```

Figure 11: Compile Application

5.3. Compilation Result

During the compilation process, an *out* directory is generated in the QuecOpen CSDK root directory to store the generated compilation files.

After a successful compilation, a *target* directory is generated in the QuecOpen CSDK root directory, which stores the compiled Kernel firmware, target application firmware, and firmware containing Kernel and application. If only the application firmware needs to be updated, burn the newly generated application firmware; If both the application firmware and the Kernel firmware need updating, burn the newly generated firmware package containing the Kernel and the application.



Figure 12: Compilation Results

5.4. Compiled File Clearance

Open a CLI or PowerShell (only Windows 10 supports PowerShell) and execute **build_all clean** to clear the compiled files in the *out* directory in QuecOpen CSDK but will not delete the target firmware in the *target* directory. When the compilation command parameters are the same, the firmware generated in the previous compilation version stored in the *target* directory will be automatically cleared before the current compilation operation is successful. If the compilation command parameters are different, the firmware generated in the previous compilation version stored in the *target* directory will not be cleared.

6 Feature Removal

6.1. Feature Removal in Kernel

QuecOpen CSDK opens all feature components by default, and the removable libraries are listed in *CMakeLists.txt* in the root directory. Each library is removed according to the macro control of kernel feature components, but the basic libraries are not allowed to be removed. You can remove the required feature in *components\ql-config\build\{project model}\8915DM_cat1_open\target.config*. A new compilation is required after removal, and Kernel firmware uses the files generated after compilation. See the following table for the removable libraries.



Figure 13: Removable Libraries in Kernel

You can see the following modification configuration file *target.config* for feature removal:

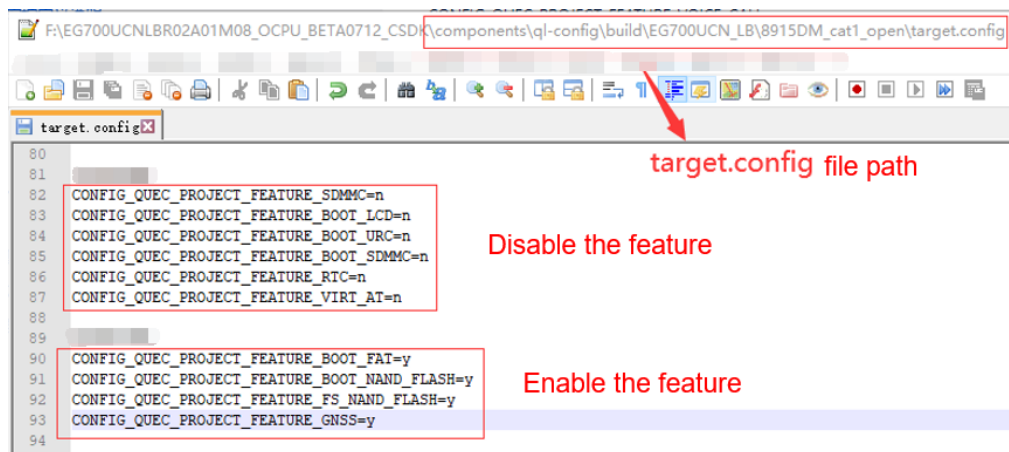


Figure 14: Feature Removal in Kernel

See the following table for the removable libraries in Kernel.

Table 2: The Removable Libraries in Kernel

Feature Macro	Feature	Dependency
CONFIG_QUEEC_PROJECT_FEATURE_USBNET	USBnet	-
CONFIG_QUEEC_PROJECT_FEATURE_GNSS	GNSS	CONFIG_QUEEC_PROJECT_FEATURE_UART
CONFIG_QUEEC_PROJECT_FEATURE_RTC	RTC	-
CONFIG_QUEEC_PROJECT_FEATURE_VIRT_AT	Virtual AT port	-
CONFIG_QUEEC_PROJECT_FEATURE_FOTA	FOTA	-
CONFIG_QUEEC_PROJECT_FEATURE_SMS	SMS	CONFIG_QUEEC_PROJECT_FEATURE_HTTP
CONFIG_QUEEC_PROJECT_FEATURE_VOICE_CALL	Voice call	-
CONFIG_QUEEC_PROJECT_FEATURE_FTP	FTP	-
CONFIG_QUEEC_PROJECT_FEATURE_HTTP	HTTP	-
CONFIG_QUEEC_PROJECT_FEATURE_SSH2	SSH	-
CONFIG_QUEEC_PROJECT_FEATURE_MQTT	MQTT	-

_MQTT		
CONFIG_QUEEC_PROJECT_FEATURE_PING	Ping	-
CONFIG_QUEEC_PROJECT_FEATURE_NTP	NTP	-
CONFIG_QUEEC_PROJECT_FEATURE_MMS	MMS	-
CONFIG_QUEEC_PROJECT_FEATURE_LBS	LBS	CONFIG_QUEEC_PROJECT_FEATURE_HTTP
CONFIG_QUEEC_PROJECT_FEATURE_UART	UART	-
CONFIG_QUEEC_PROJECT_FEATURE_I2C	I2C	-
CONFIG_QUEEC_PROJECT_FEATURE_KEYPAD	Matrix keyboard	-
CONFIG_QUEEC_PROJECT_FEATURE_LCD	LCD	-
CONFIG_QUEEC_PROJECT_FEATURE_BT	BT	-
CONFIG_QUEEC_PROJECT_FEATURE_BT_HFP	BT HFP	CONFIG_QUEEC_PROJECT_FEATURE_BT
CONFIG_QUEEC_PROJECT_FEATURE_BT_A2DP_AVRCP	A2DP	CONFIG_QUEEC_PROJECT_FEATURE_BT
CONFIG_QUEEC_PROJECT_FEATURE_BLE_GATT	BLE	CONFIG_QUEEC_PROJECT_FEATURE_BT
CONFIG_QUEEC_PROJECT_FEATURE_CAMERA	Camera	-
CONFIG_QUEEC_PROJECT_FEATURE_SPI	SPI	-
CONFIG_QUEEC_PROJECT_FEATURE_SPI_FLASH	SPI4 Flash	CONFIG_QUEEC_PROJECT_FEATURE_SPI
CONFIG_QUEEC_PROJECT_FEATURE_SPI_NOR_FLASH	SPI4 NOR Flash	CONFIG_QUEEC_PROJECT_FEATURE_SPI CONFIG_QUEEC_PROJECT_FEATURE_SPI_FLASH
CONFIG_QUEEC_PROJECT_FEATURE_SPI_NAND_FLASH	SPI4 NAND Flash	CONFIG_QUEEC_PROJECT_FEATURE_SPI CONFIG_QUEEC_PROJECT_FEATURE_SPI_FLASH
CONFIG_QUEEC_PROJECT_FEATURE_SDMMC	SDMMC	-
CONFIG_QUEEC_PROJECT_FEATURE_PBK	PBK	-

CONFIG_QUEEC_PROJECT_FEATURE_FILE_ZIP	File compression	-
CONFIG_QUEEC_PROJECT_FEATURE_FS_NAND_FLASH	SPI4 NAND Flash (file system)	CONFIG_QUEEC_PROJECT_FEATURE_SPI CONFIG_QUEEC_PROJECT_FEATURE_SPI_FLASH CONFIG_QUEEC_PROJECT_FEATURE_SPI_NAND_FLASH
CONFIG_QUEEC_PROJECT_FEATURE_SPI6_EXT_NOR	SPI6 NOR FLASH (file system)	-
CONFIG_QUEEC_PROJECT_FEATURE_FILE_AT	File system AT command	-
CONFIG_QUEEC_PROJECT_FEATURE_CLOUDOTA	HTTP OTA	CONFIG_QUEEC_PROJECT_FEATURE_HTTP
CONFIG_QUEEC_PROJECT_FEATURE_VOLTE	VoLTE	-
CONFIG_QUEEC_PROJECT_FEATURE_AUDIO	Audio	-
CONFIG_QUEEC_PROJECT_FEATURE_USB	USB	-
CONFIG_QUEEC_PROJECT_FEATURE_LEDCFG	LED&PWM	-
CONFIG_QUEEC_PROJECT_FEATURE_WIFISCAN	Wi-Fi Scan	-

6.2. Feature Removal in BootLoader

The following features in BootLoader are removable. If a feature is not required, disable it in *target.config*; If a feature is required, enable the feature in *target.config* and then call the feature initialization function in *boot2_start_8910.c*. New compilation is required after feature removal.

Table 3: The Removable Libraries in BootLoader

Feature Macro	Feature	Dependency
CONFIG_QUEEC_PROJECT_FEATURE_BOOT_LCD	Boot LCD	CONFIG_QUEEC_PROJECT_FEATURE_LCD
CONFIG_QUEEC_PROJECT_FEATURE_BOOT_FAT	Boot FAT file system	-
CONFIG_QUEEC_PROJECT_FEATURE_BOOT_SDMMC	Boot SDMMC	CONFIG_QUEEC_PROJECT_FEATURE_BOOT_FAT

CONFIG_QUEC_PROJECT_FEATURE
_BOOT_NAND_FLASH

Boot NAND Flash

CONFIG_QUEC_PROJECT_FE
ATURE_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:

```

Feature initialization function

Figure 15: Feature Initialization Function

7 Preset File

A preset file is a specified initial file packaged in a firmware package, which is written directly to the module file system during firmware downloading. Preset files take up file system space, and the maximum size of a single preset file is currently limited to 512 KB. If the file size exceeds 512 KB, the file will be skipped when it is downloaded.

7.1. Enable Preset File Package

In the `ql_app_feature_config.cmake` file, you can configure whether to package the preset file through `QL_APP_PACK_FILE`, as shown in the following figure:

```
#####
# Quectel open sdk package config
#####
if (QL_APP_FEATURE_GNSS)
option(QL_APP_PACK_FILE "Enable pack file to firmware package" ON)  Pre-set file switch
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()
# Path of the pre-set file to be packaged
message(STATUS "QL_APP_PACK_FILE ${QL_APP_PACK_FILE} @ ${QL_APP_PACK_FILE_JSON_PATH}")
```

Figure 16: Preset File Package Compilation Configuration

NOTE

The preset files package compilation configuration is affected by the GNSS in SDK. When GNSS is enabled, the GNSS chip firmware (currently with the size about 238 KB) will be packaged into the APP firmware as a preset file by default. For details about the GNSS supported by the EC200U series QuecOpen module, see the hardware design documents corresponding to each module.

If other feature files need to be preset, you should add the corresponding feature switch. For example, if you want to preset audio file, add a switch for audio.

```
#####
# 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()
```

Figure 17: Add a Feature Switch for Preset File

7.2. Preset File Configuration

QL_APP_PACK_FILE_JSON_PATH in *ql_app_feature_config.cmake* file specifies the path of the specific configuration of the preset file. The script packs the preset file according to the specified *json* file. The *json* file in CSDK specifies the path of the GNSS chip firmware, which is represented by a relative path, that is, it is located in the same path where the *json* file is stored. You can refer to this format in the *json* file to add and modify in the "files" node in the figure below. Be careful not to preset too many files to avoid taking up too much file system space.

```
{
  "_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 substituted",
    "to the absolute 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"
    }
  ]
}
```

Figure 18: Preset Package Configuration File

The preset files usually are downloaded to the directory of *user*. which is convenient for you to query through the FILE API of module. Note that the total length of the file path and file name to be written cannot exceed 192 bytes.

7.3. Preset File Download and Upgrade

The preset file can also be downloaded, upgraded or deleted by FOTA.

- Upgrade the preset file in the old firmware version to that in the new firmware version.
- If the old firmware version does not have a preset file, you can add a preset file by FOTA.
- If the old firmware version has a preset file and the new firmware version does not have a preset file, the preset files will be deleted after FOTA upgrade;

- If the preset files need to be ignored in FOTA upgrade, you can modify the value of *method* to “ignore” from “diff” in `<paccpio id="PREPACK" method="diff"> </paccpio>` of xml configuration file. See **document [2]** for details.

```

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

Figure 19: Ignore the Preset File in FOTA

8 Flash Partition Adjustment

Quectel EC200U Series QuecOpen module supports adjusting embedded flash partitions, including Kernel, application and file system partition size. See **document [3]** for specific implementation.

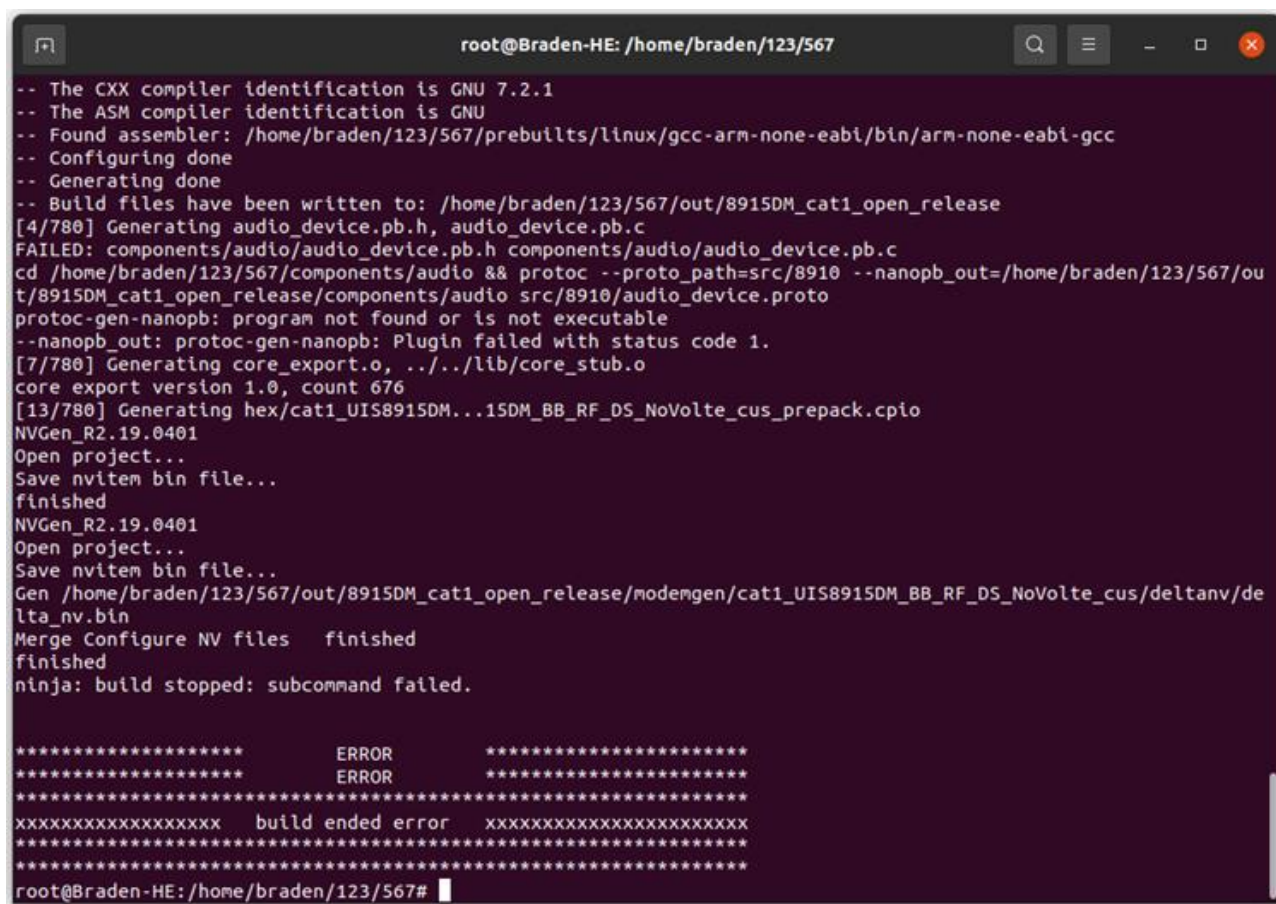
9 Firmware Burning

Quectel EC200U Series QuecOpen module supports burning firmware with QFlash. See **document [1]** for QFlash usage.

10 FAQ

10.1. Common Compilation Error in Linux

1. If the error shown in the following figure occurs, it indicates that the required software is not installed on the host system. Just reinstall the software and see **Chapter 3.1** for details.



```

root@Braden-HE: /home/braden/123/567
-- The CXX compiler identification is GNU 7.2.1
-- The ASM compiler identification is GNU
-- Found assembler: /home/braden/123/567/prebuilts/linux/gcc-arm-none-eabi/bin/arm-none-eabi-gcc
-- Configuring done
-- Generating done
-- Build files have been written to: /home/braden/123/567/out/8915DM_cat1_open_release
[4/780] Generating audio_device.pb.h, audio_device.pb.c
FAILED: components/audio/audio_device.pb.h components/audio/audio_device.pb.c
cd /home/braden/123/567/components/audio && protoc --proto_path=src/8910 --nanopb_out=/home/braden/123/567/out/8915DM_cat1_open_release/components/audio src/8910/audio_device.proto
protoc-gen-nanopb: program not found or is not executable
--nanopb_out: protoc-gen-nanopb: Plugin failed with status code 1.
[7/780] Generating core_export.o, ../../lib/core_stub.o
core export version 1.0, count 676
[13/780] Generating hex/cat1_UIS8915DM...15DM_BB_RF_DS_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...
Gen /home/braden/123/567/out/8915DM_cat1_open_release/modemgen/cat1_UIS8915DM_BB_RF_DS_NoVolte_cus/deltanv/delta_nv.bin
Merge Configure NV files finished
finished
ninja: build stopped: subcommand failed.

***** ERROR *****
***** ERROR *****
*****
XXXXXXXXXXXXXXXXXXXX build ended error XXXXXXXXXXXXXXXXXXXX
*****
root@Braden-HE: /home/braden/123/567#

```

2. If there is an error message "dttools: error while loading shared libraries: libicui18n.so.55: cannot open shared object file: No such file or directory. Ninja: build stopped: subcommand failed", it indicates dttools is missing a component. You can install the component by executing the following command in sequence.

```
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/apploder/core_export.o lib/core_stub.o
cd /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_open_release/components/apploder && dt
ools expgen -p 8910 --export /home/q/Public/EC600UCNLBR02A01M08_OCPU_BETA0701_CCSDK/out/8915DM_cat1_open_release/
components/apploder/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/apploder/CMakeFiles/export_list_gen.dir/src/core_export.list.obj
dttools: 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/modengen/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/modengen/cat1_UIS8915DM_B
B_RF_SS_NoVolte_cus/deltanv/delta_nv.bin
Merge Configure NV files finished
finished
dttools: 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 *****
*****
xxxxxxxxxxxxxxxxxx build ended error xxxxxxxxxxxxxxxxxxxxxxxx
*****
```


11 Appendix References

Table 4: Related Documents

Document Name
[1] Quectel_QFlash_User_Guide
[2] Quectel_EC200U_Series_QuecOpen_FOTA_API_Reference_Manual
[3] Quectel_EC200U_Series_QuecOpen_Embedded_Flash_Partition_Adjustment_Guide

Table 5: Terms and Abbreviations

Abbreviation	Description
(U)SIM	(Universal) Subscriber Identity Module
A2DP	Advanced Audio Distribution Profile
ADC	Analog-to-Digital Converter
API	Application Programming Interface
App	Application
BLE	Bluetooth Low Energy
BT	Bluetooth
CLI	Command-line Interface
FOTA	Firmware Over-The-Air
FTP	File Transfer Protocol
GCC	GNU Compiler Collection
GNSS	Global Navigation Satellite System
GPIO	General-Purpose Input/Output

HFP	Hands-free Profile
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
IoT	Internet of Things
LBS	Location Based Services
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MMS	Multimedia Messaging Service
MQTT	Message Queuing Telemetry Transport
NTP	Network Time Protocol
OSI	Open System Interconnection Reference Model
PBK	Phonebook
PWM	Pulse Width Modulation
RTC	Real Time Clock
RTOS	Real-Time Operating System
SDMMC	Secure Digital/MultiMediaCard
SMS	Short Message Service
SPI	Serial Peripheral Interface
SSH	Secure Shell
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VoLTE	Voice (voice calls) over LTE
Wi-Fi	Wireless Fidelity