EC200U Series QuecOpen Embedded Flash Partition Adjustment

LTE Standard Series

Version: 1.0

Date: 2022-02-15

Status: Released

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.

About the Document

Revision History

Version	Date	Author	Description
-	2021-12-21	Jensen FANG Mahat XU	Creation of the document
1.0	2022-02-15	Jensen FANG Mahat XU	First official release

Contents

About the Document **3**
Contents **4**
Table Index **5**
Figure Index **6**

1 Introduction **7**

2 Embedded Flash Partition **8**

3 Flash Partition Adjustment **10**
3.1. Flash Partition Adjustment Principle 10
3.2. Flash Partition Configuration File 10
3.3. Flash Partition Adjustment Method 13
3.3.1. Adjust the Size of Flash Partition 13
3.3.2. Add a Customized Flash Partition 14

4 Embedded Flash API **16**
4.1. Header File 16
4.2. Optional Feature API Configuration 16
4.3. API Description 16
4.3.1. ql_embed_nor_flash_write 16
4.3.1.1. ql_embed_nor_flash_e 17
4.3.2. ql_embed_nor_flash_read 18
4.3.3. ql_embed_nor_flash_erase 18

5 Embedded Flash Partition Adjustment Demo **20**
5.1. Modify FS Partition Information 20
5.2. Enable the Demo 21
5.3. Run the Demo 22

6 Appendix References **24**

Table Index

Table 1: Embedded Flash Partition Introduction 8
Table 2: Related Documents 24
Table 3: Terms and Abbreviations 24

Figure Index

Figure 1: COM Ports of Device Manager 23
Figure 2: Debugging Information of USB AP Log Port 23

1 Introduction

Quectel EC200U series module supports QuecOpen® solution. QuecOpen® is an embedded development platform based on RTOS. It is intended to simplify the design and development of IoT applications. For more information on QuecOpen®, see ***document [1]***.

This document introduces the embedded flash partition, partition adjustment, related API and demos on Quectel EC200U series module in QuecOpen® solution.

2 Embedded Flash Partition

The start address of the embedded flash partitions of EC200U series module is 0x60000000, and the size is 8 MB. The content and description of the embedded flash partitions are shown in the following table. The entire space has been allocated by default.

Table 1: Embedded Flash Partition Introduction

Partition Name	Start Address and Size	Macro Definition	Description
BOOT	0x60000000	**CONFIG_BOOT_FLASH_ADDRESS**	Store boot loader image
	64 KB	**CONFIG_BOOT_FLASH_SIZE**	
APP	0x60010000	**CONFIG_APP_FLASH_ADDRESS**	Store AP image in kernel layer
	2304 KB	**CONFIG_APP_FLASH_SIZE**	
APPIMG	0x60250000	**CONFIG_APPIMG_FLASH_ADDRESS**	Store App image in application layer
	1152 KB [1]/ 832 KB [2]	**CONFIG_APPIMG_FLASH_SIZE**	
FS	0x60370000 [1]/ 0x60320000 [2]	**CONFIG_FS_SYS_FLASH_ADDRESS**	File system partition
	1948 KB [1]/ 1536 KB [2]	**CONFIG_FS_SYS_FLASH_SIZE**	
MODEM	0x60560000 [1]/ 0x604A0000 [2]	**CONFIG_FS_MODEM_FLASH_ADDRESS**	Network protocol stack system partition
	2560 KB [1]/ 3328 KB [2]	**CONFIG_FS_MODEM_FLASH_SIZE**	
FACTORY	0x607E0000	**CONFIG_FS_FACTORY_FLASH_ADDRESS**	Store basic configurations and RF parameters
	128 KB	**CONFIG_FS_FACTORY_FLASH_SIZE**	

[1] Non-VoLTE version.
[2] VoLTE version.

NOTE

1. The partition adjustment of the embedded flash can only be performed among the three partitions of *APP*, *APPIMG* and *FS*.
2. The macro definition in the above table is located in the header file *hal_config.h* and it is automatically generated by the configuration script. The header file is located in the root directory of *out\8915DM_cat1_open_release\include* by default after SDK compilation.

3 Flash Partition Adjustment

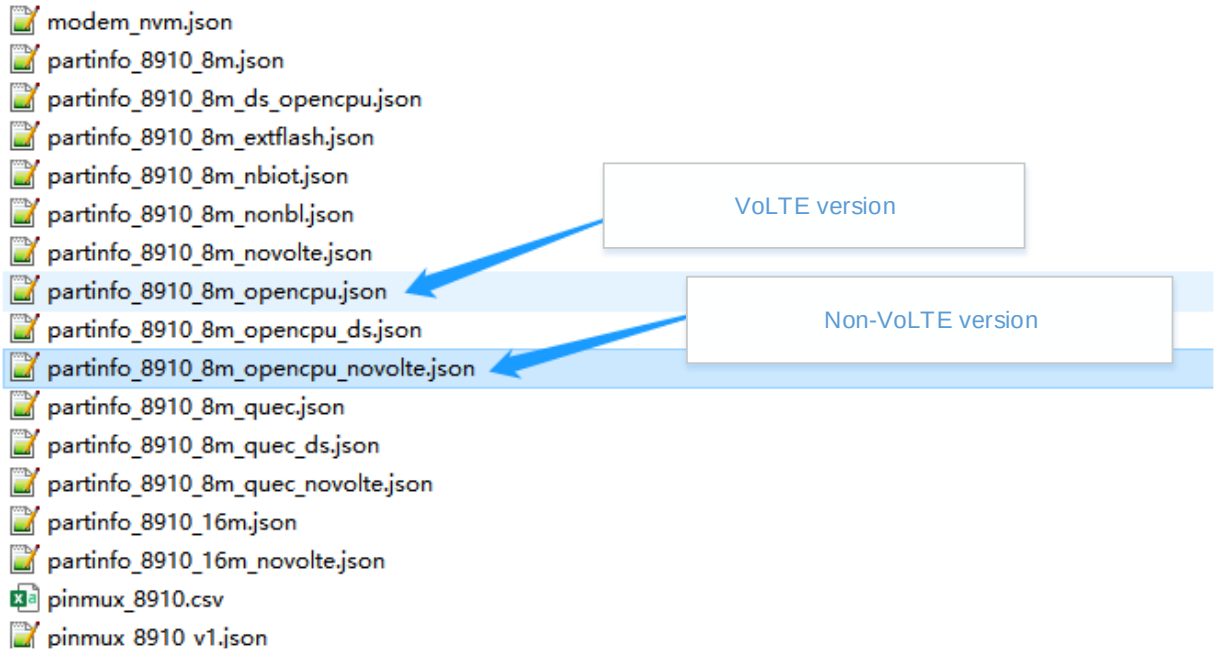
This chapter introduces how to adjust the embedded flash partition and its configuration file, how to modify the partition configuration file, and how to add the partition that can be directly operated through related API.

3.1. Flash Partition Adjustment Principle

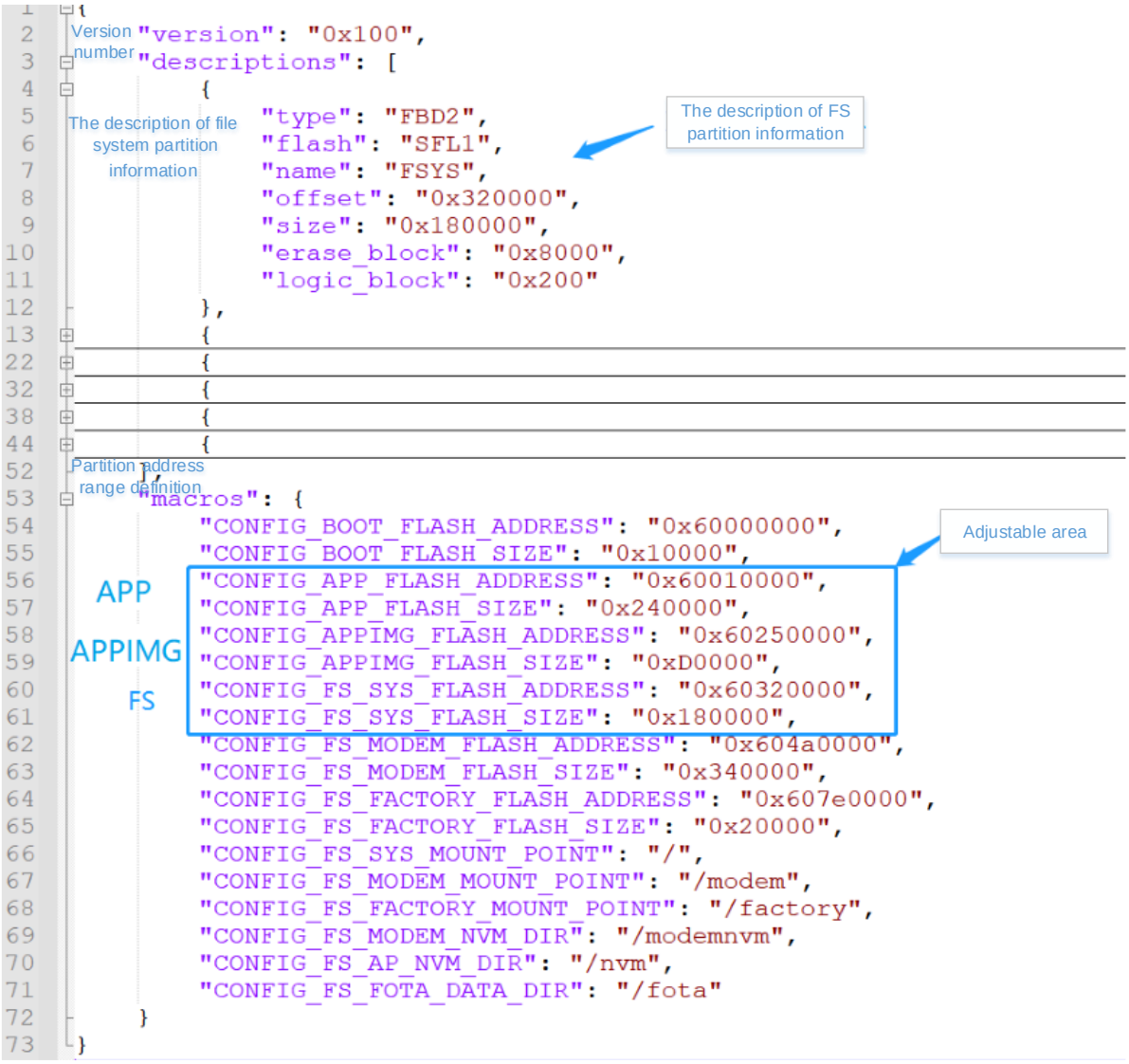
1. FOTA upgrade or downgrade is not allowed between firmware versions with different partitions.
2. The embedded flash partition adjustment can only be performed among the three partitions, *APP*, *APPIMG* and *FS*. The *APP* and *APPIMG* partitions should be 4K aligned, and the *FS* partition should be 32K aligned.
3. Execute **new** to compile after partition adjustment.

3.2. Flash Partition Configuration File

You can adjust the embedded flash partition of the module based on actual requirements. The embedded flash partition configuration file is located in the directory of *components\hal\config\8910* by default, and the corresponding file is shown in the figure below:



Partition configuration file contains three parts, version number (version), the description of file system partition information (descriptions) and partition address range definition (macros). Take *partinfo_8910_8m_opencpu.json* as an example, the partition configuration information is shown in the figure below:




```
{
    "version": "0x100",
    "descriptions": [
        {
            "type": "FBD2",
            "flash": "SFL1",
            "name": "FSYS",
            "offset": "0x320000",
            "size": "0x180000",
            "erase_block": "0x8000",
            "logic_block": "0x200"
        },
        {
        {
        {
        {
        {
    ],
    "macros": {
        "CONFIG_BOOT_FLASH_ADDRESS": "0x60000000",
        "CONFIG_BOOT_FLASH_SIZE": "0x10000",
        "CONFIG_APP_FLASH_ADDRESS": "0x60010000",
        "CONFIG_APP_FLASH_SIZE": "0x240000",
        "CONFIG_APPIMG_FLASH_ADDRESS": "0x60250000",
        "CONFIG_APPIMG_FLASH_SIZE": "0xD0000",
        "CONFIG_FS_SYS_FLASH_ADDRESS": "0x60320000",
        "CONFIG_FS_SYS_FLASH_SIZE": "0x180000",
        "CONFIG_FS_MODEM_FLASH_ADDRESS": "0x604a0000",
        "CONFIG_FS_MODEM_FLASH_SIZE": "0x340000",
        "CONFIG_FS_FACTORY_FLASH_ADDRESS": "0x607e0000",
        "CONFIG_FS_FACTORY_FLASH_SIZE": "0x20000",
        "CONFIG_FS_SYS_MOUNT_POINT": "/",
        "CONFIG_FS_MODEM_MOUNT_POINT": "/modem",
        "CONFIG_FS_FACTORY_MOUNT_POINT": "/factory",
        "CONFIG_FS_MODEM_NVM_DIR": "/modemnvm",
        "CONFIG_FS_AP_NVM_DIR": "/nvm",
        "CONFIG_FS_FOTA_DATA_DIR": "/fota"
    }
}
```

NOTE

If you want to know whether the module is VoLTE version or not, please contact Quectel Technical Supports.

3.3. Flash Partition Adjustment Method

3.3.1. Adjust the Size of Flash Partition

In the partition configuration file, you can adjust the start address and range among *APP*, *APPIMG* and *FS* only.

- When adjusting the size of *APP* and *APPIMG*, you only need to modify the information of partition address range definition (macros).
- When adjusting the size of *FS* partition, you need to modify the information of partition address range definition (macros) and *FS* partition description (descriptions).

Take *partinfo_8910_8m_opencpu_novolte.json* as an example, the partition information to be adjusted is shown in the figure below:

```
{
    "version": "0x100",
    "descriptions": [
        {
            "type": "FBD2",
            "flash": "SFL1",
            "name": "FSYS",
            "offset": "0x370000",
            "size": "0x1F0000",
            "erase_block": "0x8000",
            "logic_block": "0x200"
        },
        {
        {
        {
        {
        {
    ],
    "macros": {
        "CONFIG_BOOT_FLASH_ADDRESS": "0x60000000",
        "CONFIG_BOOT_FLASH_SIZE": "0x10000",
        "CONFIG_APP_FLASH_ADDRESS": "0x60010000",
        "CONFIG_APP_FLASH_SIZE": "0x240000",
        "CONFIG_APPIMG_FLASH_ADDRESS": "0x60250000",
        "CONFIG_APPIMG_FLASH_SIZE": "0x120000",
        "CONFIG_FS_SYS_FLASH_ADDRESS": "0x60370000",
        "CONFIG_FS_SYS_FLASH_SIZE": "0x1F0000",
        "CONFIG_FS_MODEM_FLASH_ADDRESS": "0x60560000",
        "CONFIG_FS_MODEM_FLASH_SIZE": "0x280000",
        "CONFIG_FS_FACTORY_FLASH_ADDRESS": "0x607e0000",
        "CONFIG_FS_FACTORY_FLASH_SIZE": "0x20000",
        "CONFIG_FS_SYS_MOUNT_POINT": "/",
        "CONFIG_FS_MODEM_MOUNT_POINT": "/modem",
        "CONFIG_FS_FACTORY_MOUNT_POINT": "/factory",
        "CONFIG_FS_MODEM_NVM_DIR": "/modemnvm",
        "CONFIG_FS_AP_NVM_DIR": "/nvm",
        "CONFIG_FS_FOTA_DATA_DIR": "/fota"
    }
}
```

Execute **new** to compile after partition adjustment. See ***document [1]*** for details.

3.3.2. Add a Customized Flash Partition

When adding the customized flash partition, you can only adjust the start address and range size of the three partitions, *APP*, *APPIMG* and *FS*.

- When adding the flash partition in *APP* or *APPIMG*, you need to modify the information of partition address range definition (macros), as shown in the figure below:

```
{
    "version": "0x100",
    "descriptions": [
        {
        {
        {
        {
        {
        {
    ],
    "macros": {
        "CONFIG_BOOT_FLASH_ADDRESS": "0x60000000",
        "CONFIG_BOOT_FLASH_SIZE": "0x10000",
        "CONFIG_APP_FLASH_ADDRESS": "0x60010000",
        "CONFIG_APP_FLASH_SIZE": "0x240000",
        "CONFIG_APPIMG_FLASH_ADDRESS": "0x60250000",
        "CONFIG_APPIMG_FLASH_SIZE": "0x120000",
        "CONFIG_FS_SYS_FLASH_ADDRESS": "0x60370000",
        "CONFIG_FS_SYS_FLASH_SIZE": "0x1F0000",
        "CONFIG_FS_MODEM_FLASH_ADDRESS": "0x60560000",
        "CONFIG_FS_MODEM_FLASH_SIZE": "0x280000",
        "CONFIG_FS_FACTORY_FLASH_ADDRESS": "0x607e0000",
        "CONFIG_FS_FACTORY_FLASH_SIZE": "0x20000",
        "CONFIG_FS_SYS_MOUNT_POINT": "/",
        "CONFIG_FS_MODEM_MOUNT_POINT": "/modem",
        "CONFIG_FS_FACTORY_MOUNT_POINT": "/factory",
        "CONFIG_FS_MODEM_NVM_DIR": "/modemnvm",
        "CONFIG_FS_AP_NVM_DIR": "/nvm",
        "CONFIG_FS_FOTA_DATA_DIR": "/fota"
    }
}
```

Take the above figure as an example, modify "CONFIG_APPIMG_FLASH_SIZE" from 0x120000 to 0x11F000, that is, to reserve 4 KB of memory. Then the flash address [0x6036F000~0x60370000) area can be read, written and erased directly through API.

- When adding the flash partition in *FS*, you need to modify *FS* partition address range definition (macros) and *FS* partition description (descriptions), as shown in the figure below:

```
{
    "version": "0x100",
    "descriptions": [
        {
            "type": "FBD2",
            "flash": "SFL1",
            "name": "FSYS",
            "offset": "0x320000",
            "size": "0x180000",
            "erase_block": "0x8000",
            "logic_block": "0x200"
        },
        {
        {
        {
        {
        {
    ],
    "macros": {
        "CONFIG_BOOT_FLASH_ADDRESS": "0x60000000",
        "CONFIG_BOOT_FLASH_SIZE": "0x10000",
        "CONFIG_APP_FLASH_ADDRESS": "0x60010000",
        "CONFIG_APP_FLASH_SIZE": "0x240000",
        "CONFIG_APPIMG_FLASH_ADDRESS": "0x60250000",
        "CONFIG_APPIMG_FLASH_SIZE": "0xD0000",
        "CONFIG_FS_SYS_FLASH_ADDRESS": "0x60320000",
        "CONFIG_FS_SYS_FLASH_SIZE": "0x180000",
        "CONFIG_FS_MODEM_FLASH_ADDRESS": "0x604a0000",
        "CONFIG_FS_MODEM_FLASH_SIZE": "0x340000",
        "CONFIG_FS_FACTORY_FLASH_ADDRESS": "0x607e0000",
        "CONFIG_FS_FACTORY_FLASH_SIZE": "0x20000",
        "CONFIG_FS_SYS_MOUNT_POINT": "/",
        "CONFIG_FS_MODEM_MOUNT_POINT": "/modem",
        "CONFIG_FS_FACTORY_MOUNT_POINT": "/factory",
        "CONFIG_FS_MODEM_NVM_DIR": "/modemnvm",
        "CONFIG_FS_AP_NVM_DIR": "/nvm",
        "CONFIG_FS_FOTA_DATA_DIR": "/fota"
    }
}
```

Take the above figure as an example, modify "CONFIG_FS_SYS_FLASH_SIZE" and "size" in the *FS* partition description from 0x180000 to 0x178000, that is, to reduce the size of the *FS* partition by 0x8000 bytes (32 KB) and reserve 32 KB of memory. Then the flash address [0x60498000 ~0x604a0000) area can be read, written and erased directly through API.

NOTE

1. When a flash partition is added in *APP* or *APPIMG*, the partition size and address must be 4K aligned.
2. When a flash partition is added in *FS*, the partition size and address must be 32K aligned.

4 Embedded Flash API

The embedded flash API described in this chapter can only read, write and erase the customized flash partitions. Before using the API, you need to make sure that the embedded flash partition is correctly reserved. Otherwise, all functions will return an error code.

4.1. Header File

ql_embed_nor_flash.h, the embedded Flash API, is located in the directory of *components\ql-kernel\inc* of the SDK. Unless otherwise specified, the header files mentioned in this document are all located in this directory.

4.2. Optional Feature API Configuration

The customized embedded flash partition and its API are optional features. You need to modify the feature macros control configuration file before using the optional features. The default path of the configuration file is *components\ql-config\build\EXX00UXX_XX\8915DM_cat1_open\target.config*.

EXX00UXX_XX refers to the module used by customers. Add the following configuration to the configuration file to enable API for the reserved embedded flash partition:

```
CONFIG_QUEC_PROJECT_FEATURE_EMBED_NOR_FLASH=y
```

4.3. API Description

4.3.1. ql_embed_nor_flash_write

This function writes the data to the specified address of embedded flash. The address to be written and the size of the data to be written do not require block alignment or page alignment. Before writing the data, you need to make sure that the sector where the address is written has been erased. Otherwise the written data will be different from the data read.

- **Prototype**

```
ql_embed_nor_flash_e ql_embed_nor_flash_write(uint32 write_addr,void *data,size_t size);
```

- **Parameter**

write_addr:
[In] The address to be written.

data:
[In] Point to the data to be written.

size:
[In] Length of the data. Unit: byte.

- **Return Value**

See ***Chapter 4.3.1.1*** for details.

4.3.1.1. ql_embed_nor_flash_e

The error codes of the embedded flash API indicate whether the API is executed successfully or not. If the function fails to execute, the reason for the error will be returned. The enumeration of the error codes is defined below:

```
typedef enum
{
  QL_EMBED_NOR_FLASH_SUCCESS = QL_SUCCESS,
  QL_EMBED_NOR_FLASH_WRITE_ERR=1|QL_EMBED_NOR_FLASH_ERRCODE_BASE,
  QL_EMBED_NOR_FLASH_READ_ERR,
  QL_EMBED_NOR_FLASH_ERASE_ERR,
  QL_EMBED_NOR_FLASH_OPERATE_ERR,
  QL_EMBED_NOR_FLASH_ADDRESS_ERR,
}ql_embed_nor_flash_e;
```

- **Member**

Member	Description
QL_EMBED_NOR_FLASH_SUCCESS	Successful execution
QL_EMBED_NOR_FLASH_WRITE_ERR	Fail to write the data
QL_EMBED_NOR_FLASH_READ_ERR	Fail to read the data
QL_EMBED_NOR_FLASH_ERASE_ERR	Fail to erase the data
QL_EMBED_NOR_FLASH_OPERATE_ERR	Operation failure
QL_EMBED_NOR_FLASH_ADDRESS_ERR	Illegal address

4.3.2. ql_embed_nor_flash_read

This function reads the data from the specified address of embedded flash. The address read and the size of the data read do not require block alignment or page alignment.

- **Prototype**

```
ql_embed_nor_flash_e ql_embed_nor_flash_read(uint32 read_addr,void *data,size_t size);
```

- **Parameter**

read_addr:
[In] The address read.

data:
[Out] Point to the data read.

size:
[In] Length of the data. Unit: byte.

- **Return Value**

See ***Chapter 4.3.1.1*** for details.

4.3.3. ql_embed_nor_flash_erase

This function erases the specified sector of the embedded flash. The address and size of the erased partition must be 4K aligned. The minimum erasing unit is 4 KB.

- **Prototype**

```
ql_embed_nor_flash_e ql_embed_nor_flash_erase(uint32_t erase_addr,size_t size);
```

- **Parameter**

erase_addr:
[In] The address to be erased.

size:
[In] Length of the data. Unit: byte.

- **Return Value**

See ***Chapter 4.3.1.1*** for details.

5 Embedded Flash Partition Adjustment Demo

This chapter introduces how to modify the configuration file of default partition and enable the demo, how to add the customized embedded flash partition and test the writing and reading features on the partition. Users can modify the corresponding information based on their requirements.

5.1. Modify FS Partition Information

Open the *.json* configuration file, located in the directory of *components\hal\config\8910*. In the partition address range definition (macros), change the FS partition size, that is, change "CONFIG_FS_SYS_FLASH_SIZE" from 0x1F0000 to 0x1E8000, as shown in the figure below.

```
53      "macros": {
54          "CONFIG_BOOT_FLASH_ADDRESS": "0x60000000",
55          "CONFIG_BOOT_FLASH_SIZE": "0x10000",
56          "CONFIG_APP_FLASH_ADDRESS": "0x60010000",
57          "CONFIG_APP_FLASH_SIZE": "0x240000",
58          "CONFIG_APPIMG_FLASH_ADDRESS": "0x60250000",
59          "CONFIG_APPIMG_FLASH_SIZE": "0x120000",
60          "CONFIG_FS_SYS_FLASH_ADDRESS": "0x60370000",
61          "CONFIG_FS_SYS_FLASH_SIZE": "0x1E8000",
62          "CONFIG_FS_MODEM_FLASH_ADDRESS": "0x60560000",
```

At the same time, change "size" to 0x1E8000 in the descriptions. Because the size of the FS partition must be 32K aligned, the reserved size in the demo is 32 KB, and the readable and writable partition is: [0x60558000~0x60560000).

```
{
    "version": "0x100",
    "descriptions": [
        {
            "type": "FBD2",
            "flash": "SFL1",
            "name": "FSYS",
            "offset": "0x370000",
            "size": "0x1E8000",
            "erase_block": "0x8000",
            "logic_block": "0x200"
```

5.2. Enable the Demo

Macro control is enabled by default in the demo. After the API optional feature configuration of the customized embedded flash partition is enabled, and the partition information is modified correctly, you can find the file *ql_init.c* in the directory of *components/ql-application/init*, and uncomment *ql_embed_nor_flash_app_init()*, as shown in the figure below:

```
#ifdef QL_APP_FEATURE_SFTP
    //ql_sftp_app_init();
#endif

#ifdef QL_APP_FEATURE_MXML
    //ql_mxml_app_init();
#endif

#ifdef QL_APP_FEATURE_EMBED_NOR_FLASH
    ql_embed_nor_flash_app_init();
#endif

    ql_rtos_task_sleep_ms(1000); /*Chaos change: set to 1000 for the camera power on*/
    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: %s @ %s", QL_APP_VERSION, QL_APP_BUILD_RELEASE_TYPE);
    prvInvokeGlobalCtors();
    if(0 == strcasecmp(QL_APP_BUILD_RELEASE_TYPE, "release"))
    {
        ql_dev_cfg_wdt(1);
        //open the kernel log
        //ql_quec_trace_enable(1);
```

Modify the **FLASH_ADDR** in *components\ql-application\embed_nor_flash\embed_nor_flash_demo.c* to 0x60558000 at the same time, that is, the reserved start address of the flash partition after partition file modification in the demo, as shown in the figure below:

```
41
42  #define QL_APP_EMBED_NOR_FLASH_LOG_LEVEL            QL_LOG_LEVEL_INFO
43  #define QL_EMBED_NOR_FLASH_LOG(msg, ...)            QL_LOG(QL_APP_EMBED_NOR_FLASH_LOG_LEVEL
44
45  #define SECTOR_SIZE   4096
46  #define FLASH_ADDR    0x60558000    //Note: For the resevered flash partition address used by Demo, you can modify it based on your
47                                      requirements.
48  ql_task_t embed_nor_flash_task = NULL;
49
50   void static embed_nor_flash_demo_thread(void *param)
51  {
52      ql_errcode_e ret=QL_SUCCESS;
53
54      char flash_buff[128]={0};
55
56      char *test_write_str="123456789abcdefghijklmnqpqrst";
57
58      QL_EMBED_NOR_FLASH_LOG("==========embed flash demo start============");
59
60      ql_rtos_task_sleep_ms(10);
61
62      //erase FLASH_ADDR sector
63      ret=ql_embed_nor_flash_erase(FLASH_ADDR,SECTOR_SIZE);
64
65      if(ret!=QL_SUCCESS)
66      {
67          QL_EMBED_NOR_FLASH_LOG("embed nor flash erase faild,erase addr:0x%X",FLASH_ADDR);
68      }
69
70      ret=ql_embed_nor_flash_write(FLASH_ADDR,(void *)test_write_str,strlen(test_write_str));
71
```

5.3. Run the Demo

Recompile the firmware package and download it to the module. Then connect LTE OPEN EVB USB port to PC with a USB cable, and you can see the COM ports as shown below in the PC device manager interface. After using the *Trace Tool* in *coolwatcher_usb.exe* to capture the log, you can see the debugging information of the demo by USB AP log port. For details about log capture, see ***document [2]***.

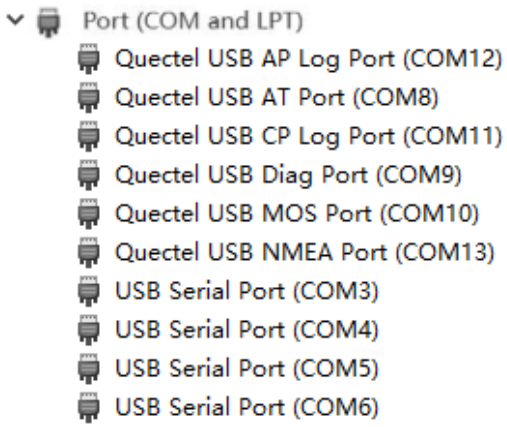


Figure 1: COM Ports of Device Manager

After the module boots, *ql_embed_nor_flash_app_init()* will be enabled automatically. You can see the information reading and writing of the embedded flash from the log. The debugging information of USB AP log port is shown in the figure below:

Received	Tick	Level	Description
20:07:19.888	6039	QOPN/I	[QL_APP_EMBED_NOR_FLASH][embed_nor_flash_demo_thread, 58] ==========embed flash demo start==========
20:07:19.990	7604	QOPN/I	[QL_APP_EMBED_NOR_FLASH][embed_nor_flash_demo_thread, 83] read addr 0x60558000,content:123456789abcdefghijklmnqpqrst
20:07:20.009	7929	QOPN/I	[QL_APP_EMBED_NOR_FLASH][embed_nor_flash_demo_thread, 87] ==========embed flash demo finished==========

Figure 2: Debugging Information of USB AP Log Port

In the logs above, as shown in the demo, from the resevered sector (0x60558000), the partition was erased, wtitten and read.

NOTE

For details about firmware compilation and download process, see ***document [1]***.

6 Appendix References

Table 2: Related Documents

Document Names
[1] Quectel_EC200U_Series_QuecOpen_CSDK_Quick_Start_Guide
[2] Quectel_EC200U_Series_QuecOpen_Log_Capture_Guide

Table 3: Terms and Abbreviations

Abbreviation	Description
AP	Application Processor
API	Application Programming Interface
App	Application
FS	File System
RTOS	Real-Time Operating System
SDK	Software Development Kit
VoLTE	Voice (voice calls) over LTE