

# **EC200U Series** QuecOpen **Virtual AT Port API** **Reference Manual**

**LTE Standard Module Series**

Version: 1.0

Date: 2021-11-19

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](mailto: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](mailto: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
-	2021-11-05	Neo KONG	Creation of the document
1.0	2021-11-19	Neo KONG	First official release

## Contents

About the Document.....	3
Contents.....	4
Table Index.....	5
<b>1 Introduction .....</b>	<b>6</b>
<b>2 Virtual AT Port API.....</b>	<b>7</b>
2.1. Header File.....	7
2.2. API Overview.....	7
2.3. API Description.....	7
2.3.1. ql_virt_at_open.....	7
2.3.1.1. ql_virt_at_port_number_e .....	8
2.3.1.2. ql_errcode_virt_at_e.....	9
2.3.2. ql_virt_at_close .....	9
2.3.3. ql_virt_at_write .....	10
2.3.4. ql_virt_at_read .....	10
<b>3 Virtual AT Port Development Example .....</b>	<b>12</b>
3.1. Development Example on Application.....	12
3.1.1. Virtual AT Port Demo Description.....	12
3.1.2. Demo Self-start .....	13
3.2. Virtual AT Port Debugging .....	14
3.2.1. Preparation.....	14
3.2.2. Debugging.....	14
<b>4 Appendix References .....</b>	<b>15</b>

## Table Index

Table 1: API Overview .....	7
Table 2: Related Documents.....	15
Table 3: Terms and Abbreviations .....	15

# 1 Introduction

Quectel EC200U series modules support QuecOpen<sup>®</sup> solution. QuecOpen is an embedded development platform based on RTOS, which is intended to simplify the design and development of IoT applications. For more information on QuecOpen<sup>®</sup>, see **document [1]**.

This document introduces the virtual AT of App layer, including related API, development demo and debugging methods of Quectel EC200U series modules.

# 2 Virtual AT Port API

## 2.1. Header File

*ql\_api.virt.h*, the header file of virtual AT port API, is located in the directory of *components\ql-kernel\inc*. Unless otherwise specified, the header files mentioned in this document are all located in this directory.

## 2.2. API Overview

Table 1: API Overview

Function	Description
<i>ql_virt_at_open()</i>	Enables virtual AT ports.
<i>ql_virt_at_close()</i>	Disables virtual AT ports.
<i>ql_virt_at_write()</i>	Writes AT commands through virtual AT ports.
<i>ql_virt_at_read()</i>	Reads AT commands responses through virtual AT ports.

## 2.3. API Description

### 2.3.1. ql\_virt\_at\_open

This function enables virtual AT ports.

- **Prototype**

```
ql_errcode_virt_at_e ql_virt_at_open(ql_virt_at_port_number_e port, ql_virt_at_callback virt_at_cb)
```



- **Parameter**

*port*:

[In] Virtual AT ports. See **Chapter 2.3.1.1** for details.

*virt\_at\_cb*:

[In] Callback function, which is used to receive the notification that the virtual AT port returns the AT command response.

- **Return Value**

See **Chapter 2.3.1.2** for virtual AT port error codes.

### 2.3.1.1. ql\_virt\_at\_port\_number\_e

The enumeration of virtual AT ports number is defined as below:

```
typedef enum
{
    QL_VIRT_AT_PORT_0,
    QL_VIRT_AT_PORT_1,
    QL_VIRT_AT_PORT_2,
    QL_VIRT_AT_PORT_3,
    QL_VIRT_AT_PORT_4,
    QL_VIRT_AT_PORT_5,
    QL_VIRT_AT_PORT_6,
    QL_VIRT_AT_PORT_7,
    QL_VIRT_AT_PORT_8,
    QL_VIRT_AT_PORT_9,
    QL_VIRT_AT_PORT_MAX,
}ql_virt_at_port_number_e;
```

- **Member**

Member	Description
QL_VIRT_AT_PORT_0	Virtual AT port number: 0.
QL_VIRT_AT_PORT_1	Virtual AT port number: 1.
QL_VIRT_AT_PORT_2	Virtual AT port number: 2.
QL_VIRT_AT_PORT_3	Virtual AT port number: 3.
QL_VIRT_AT_PORT_4	Virtual AT port number: 4.
QL_VIRT_AT_PORT_5	Virtual AT port number: 5.

<code>QL_VIRT_AT_PORT_6</code>	Virtual AT port number: 6.
<code>QL_VIRT_AT_PORT_7</code>	Virtual AT port number: 7.
<code>QL_VIRT_AT_PORT_8</code>	Virtual AT port number: 8.
<code>QL_VIRT_AT_PORT_9</code>	Virtual AT port number: 9.

### 2.3.1.2. `ql_errcode_virt_at_e`

The virtual AT port error codes indicate whether the function is executed successfully. If the function execution fails, error reasons are returned. The enumeration of virtual AT port error codes is defined as follows:

```
typedef enum
{
    QL_VIRT_AT_SUCCESS = QL_SUCCESS,
    QL_VIRT_AT_INVALID_PARAM_ERR    = 1|QL_VIRT_AT_ERRCODE_BASE,
    QL_VIRT_AT_EXECUTE_ERR,
    QL_VIRT_AT_OPEN_REPEAT_ERR
}ql_errcode_virt_at_e;
```

- **Member**

Member	Description
<code>QL_VIRT_AT_SUCCESS</code>	Successful execution.
<code>QL_VIRT_AT_INVALID_PARAM_ERR</code>	Invalid parameter.
<code>QL_VIRT_AT_EXECUTE_ERR</code>	Failed execution.
<code>QL_VIRT_AT_OPEN_REPEAT_ERR</code>	Error in repeatedly enabling virtual AT port.

### 2.3.2. `ql_virt_at_close`

This function disables virtual AT ports.

- **Prototype**

```
ql_errcode_virt_at_e ql_virt_at_close(ql_virt_at_port_number_e port)
```

- **Parameter**

*port*:

[In] Virtual AT ports. See **Chapter 2.3.1.1** for details.

- **Return Value**

See **Chapter 2.3.1.2** for virtual AT port error codes.

### 2.3.3. ql\_virt\_at\_write

This function writes AT commands through virtual AT ports.

- **Prototype**

```
ql_errcode_virt_at_e ql_virt_at_write(ql_virt_at_port_number_e port, unsigned char *data, unsigned int data_len)
```

- **Parameter**

*port*:

[In] Virtual AT ports. See **Chapter 2.3.1.1** for details.

*data*:

[In] Pointer to AT command string to be written.

*data\_len*:

[In] Length of AT command string to be written. Unit: byte.

- **Return Value**

See **Chapter 2.3.1.2** for virtual AT port error codes.

### 2.3.4. ql\_virt\_at\_read

This function reads AT commands responses through virtual AT ports.

- **Prototype**

```
ql_errcode_virt_at_e ql_virt_at_read(ql_virt_at_port_number_e port, unsigned char *data, unsigned int data_len)
```

- **Parameter**

*port*:

[In] Virtual AT ports. See **Chapter 2.3.1.1** for details.

*data*:

[Out] Pointer to AT command string that has been read.

*data\_len*:

[In] Length of AT command string that has been read. Unit: byte.

- Return Value

See *Chapter 2.3.1.2* for virtual AT port error codes.

# 3 Virtual AT Port Development Example

This chapter introduces how to use the above API for virtual AT port development and debugging on application.

## 3.1. Development Example on Application

### 3.1.1. Virtual AT Port Demo Description

*virt\_at\_demo.c*, the example file of virtual AT port, is provided in Quectel EC200U series QuecOpen SDK code.

*virt\_at\_demo.c* file mainly contains settings such as enabling virtual AT ports, writing AT commands through the virtual AT ports, and reading AT commands responses. The entry function is *ql\_virt\_at\_app\_init()*, as shown below.

```

-void ql_virt_at_app_init()
{
    QIOSStatus err = QL_OSI_SUCCESS;
    ql_task_t virt_at_task = NULL;

    err = ql_rtos_task_create(&virt_at_task, 1024, APP_PRIORITY_NORMAL, "ql_virtatdemo", ql_virt_at_demo_thread, NULL, 1);
    if( err != QL_OSI_SUCCESS )
    {
        QL_VIRTAT_DEMO_LOG("virt at demo task created failed");
        return;
    }
}
    
```

The Demo first enables three virtual AT ports, registers the corresponding callback functions, then writes AT commands through the three virtual AT ports, and reads AT commands responses through the callback function.

```

static void ql_virt_at_demo_thread(void *param)
{
    int ret = 0;
    QIOSStatus err = 0;

    char *cmd0 = "ATI\r\n";
    char *cmd1 = "AT^HEAPINFO\r\n";
    char *cmd2 = "at+qdbgcfg=\"oem\"\r\n";

    ret = ql_virt_at_open(QL_VIRT_AT_PORT_0,ql_virt_at0_notify_cb);
    if(QL_VIRT_AT_SUCCESS != ret)
    {
        QL_VIRTAT_DEMO_LOG("virt at0 open error,ret: 0x%x", ret);
        goto |exit;
    }
    QL_VIRTAT_DEMO_LOG("virt at0 open success");

    ret = ql_virt_at_open(QL_VIRT_AT_PORT_1,ql_virt_at1_notify_cb);
    if(QL_VIRT_AT_SUCCESS != ret)
    {
        QL_VIRTAT_DEMO_LOG("virt at1 open error,ret: 0x%x", ret);
        goto |exit;
    }
    QL_VIRTAT_DEMO_LOG("virt at1 open success");

    ret = ql_virt_at_open(QL_VIRT_AT_PORT_2,ql_virt_at2_notify_cb);

    if(QL_VIRT_AT_SUCCESS != ret)
    {
        QL_VIRTAT_DEMO_LOG("virt at2 open error,ret: 0x%x", ret);
        goto |exit;
    }
    QL_VIRTAT_DEMO_LOG("virt at2 open success");

    while(1)
    {
        QL_VIRTAT_DEMO_LOG("VAT0 -->: %s", cmd0);
        ql_virt_at_write(QL_VIRT_AT_PORT_0, (unsigned char*)cmd0, strlen((char *)cmd0));
        ql_rtos_task_sleep_ms(5000);

        QL_VIRTAT_DEMO_LOG("VAT1 -->: %s", cmd1);
        ql_virt_at_write(QL_VIRT_AT_PORT_1, (unsigned char*)cmd1, strlen((char *)cmd1));
        ql_rtos_task_sleep_ms(5000);

        QL_VIRTAT_DEMO_LOG("VAT2 -->: %s", cmd2);
        ql_virt_at_write(QL_VIRT_AT_PORT_2, (unsigned char*)cmd2, strlen((char *)cmd2));
        ql_rtos_task_sleep_ms(5000);
    }

    exit:
    err = ql_rtos_task_delete(NULL);
    if(err != QL_OSI_SUCCESS)
    {
        QL_VIRTAT_DEMO_LOG("task deleted failed");
    }
} « end ql_virt_at_demo_thread »
    
```

### 3.1.2. Demo Self-start

The above Demo is disabled by default in *ql\_init\_demo\_thread*. If necessary, uncomment *ql\_virt\_at\_app\_init()* to run the Demo, as shown below:

```

#ifdef QL_APP_FEATURE_HTTP_FOTA
    >> //ql_fota_http_app_init();
#endif

#ifdef QL_APP_FEATURE_DECODER
    //ql_decoder_app_init();
#endif

#ifdef QL_APP_FEATURE_KEYPAD
    //ql_keypad_app_init();
#endif

#ifdef QL_APP_FEATURE_RTC
    //ql_rtc_app_init();
#endif

#ifdef QL_APP_FEATURE_USB_CHARGE
    //ql_charge_app_init();
#endif









#ifdef QL_APP_FEATURE_VIRT_AT
    //ql_virt_at_app_init();
#endif
    
```

## 3.2. Virtual AT Port Debugging

### 3.2.1. Preparation

Virtual AT port function of EC200U series QuecOpen module can be debugged on Quectel LTE OPEN EVB.

Download the compiled version into the module, connect the USB port of LTE OPEN EVB and PC with a USB cable. The AP log port of USB is mainly used to display system debugging information. You can view the relevant information of Demo from the log printed by *cooltools*. See **document [2]** for log capturing methods.

-  Quectel Modem (COM8)
-  Quectel USB AP Log Port (COM14)
-  Quectel USB AT Port (COM5)
-  Quectel USB CP Log Port (COM11)
-  Quectel USB Diag Port (COM12)
-  Quectel USB MOS Port (COM13)
-  Quectel USB Serial-1 Port (COM9)
-  Quectel USB Serial-2 Port (COM10)

### 3.2.2. Debugging

*ql\_virt\_at\_app\_init* starts automatically after the module is booted, and AT command responses can be viewed through log information.

The AP log port debugging information is shown below:

Index	Received	Tick	Level	Description
728	11:14:39.114	4365	QOPN/I	[ql_VIRTATDEMO][ql_virt_at0_notify_cb, 63] VAT0 <--: Quectel
734	11:14:39.114	4367	QOPN/I	[ql_VIRTATDEMO][ql_virt_at0_notify_cb, 63] VAT0 <--: OK
1506	11:14:44.055	20710	QOPN/I	[ql_VIRTATDEMO][ql_virt_at_demo_thread, 145] VAT1 -->: AT^HEAPINFO
1513	11:14:44.055	20723	QOPN/I	[ql_VIRTATDEMO][ql_virt_at1_notify_cb, 81] VAT1 <--: AT^HEAPINFO
1519	11:14:44.055	20728	QOPN/I	[ql_VIRTATDEMO][ql_virt_at1_notify_cb, 81] VAT1 <--: ^HEAPINFO: 0x809df180,6033024,3803112,3795232
1525	11:14:44.059	20730	QOPN/I	[ql_VIRTATDEMO][ql_virt_at1_notify_cb, 81] VAT1 <--: OK
1660	11:14:49.060	37093	QOPN/I	[ql_VIRTATDEMO][ql_virt_at_demo_thread, 149] VAT2 -->: at+qdbgcfg="oem"
1667	11:14:49.061	37103	QOPN/I	[ql_VIRTATDEMO][ql_virt_at2_notify_cb, 99] VAT2 <--: at+qdbgcfg="oem"
1673	11:14:49.061	37108	QOPN/I	[ql_VIRTATDEMO][ql_virt_at2_notify_cb, 99] VAT2 <--: +QDBGCFG: "oem","EC200U-CNAO OCPU RELEASE","NOOEM"
1679	11:14:49.061	37110	QOPN/I	[ql_VIRTATDEMO][ql_virt_at2_notify_cb, 99] VAT2 <--: OK
1735	11:14:54.062	53477	QOPN/I	[ql_VIRTATDEMO][ql_virt_at_demo_thread, 141] VAT0 -->: ATI
1742	11:14:54.062	53486	QOPN/I	[ql_VIRTATDEMO][ql_virt_at0_notify_cb, 63] VAT0 <--: ATI
1748	11:14:54.062	53490	QOPN/I	[ql_VIRTATDEMO][ql_virt_at0_notify_cb, 63] VAT0 <--: Quectel
1754	11:14:54.062	53491	QOPN/I	[ql_VIRTATDEMO][ql_virt_at0_notify_cb, 63] VAT0 <--: OK
1889	11:14:59.064	4325	QOPN/I	[ql_VIRTATDEMO][ql_virt_at_demo_thread, 145] VAT1 -->: AT^HEAPINFO
1896	11:14:59.064	4334	QOPN/I	[ql_VIRTATDEMO][ql_virt_at1_notify_cb, 81] VAT1 <--: AT^HEAPINFO
1902	11:14:59.064	4339	QOPN/I	[ql_VIRTATDEMO][ql_virt_at1_notify_cb, 81] VAT1 <--: ^HEAPINFO: 0x809df180,6033024,3803112,3795232
1908	11:14:59.064	4341	QOPN/I	[ql_VIRTATDEMO][ql_virt_at1_notify_cb, 81] VAT1 <--: OK
2081	11:15:04.075	20709	QOPN/I	[ql_VIRTATDEMO][ql_virt_at_demo_thread, 149] VAT2 -->: at+qdbgcfg="oem"
2088	11:15:04.076	20718	QOPN/I	[ql_VIRTATDEMO][ql_virt_at2_notify_cb, 99] VAT2 <--: at+qdbgcfg="oem"
2094	11:15:04.076	20723	QOPN/I	[ql_VIRTATDEMO][ql_virt_at2_notify_cb, 99] VAT2 <--: +QDBGCFG: "oem","EC200U-CNAO OCPU RELEASE","NOOEM"
2100	11:15:04.076	20725	QOPN/I	[ql_VIRTATDEMO][ql_virt_at2_notify_cb, 99] VAT2 <--: OK

# 4 Appendix References

**Table 2: Related Documents**

Document Name
[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
COM	Communication
EVB	Evaluation Board
IoT	Internet of Things
LTE	Long-Term Evolution
RTOS	Real-Time Operating System
PC	Personal Computer
SDK	Software Development Kit
USB	Universal Serial Bus