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

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.

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

Table Index

1 Introduction

Quectel EC200U series modules support QuecOpen® 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®, 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
ql_virt_at_open()	Enables virtual AT ports.
ql_virt_at_close()	Disables virtual AT ports.
ql_virt_at_write()	Writes AT commands through virtual AT ports.
ql_virt_at_read()	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.

QL_VIRT_AT_PORT_6	Virtual AT port number: 6.
QL_VIRT_AT_PORT_7	Virtual AT port number: 7.
QL_VIRT_AT_PORT_8	Virtual AT port number: 8.
QL_VIRT_AT_PORT_9	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
QL_VIRT_AT_SUCCESS	Successful execution.
QL_VIRT_AT_INVALID_PARAM_ERR	Invalid parameter.
QL_VIRT_AT_EXECUTE_ERR	Failed execution.
QL_VIRT_AT_OPEN_REPEAT_ERR	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

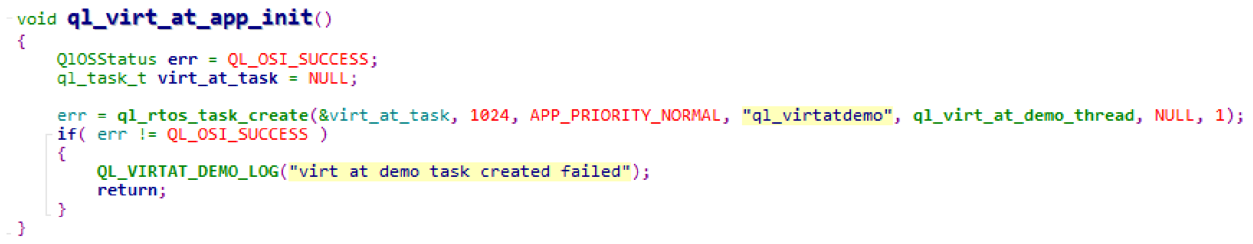
This chapter introduces how to use the above API for virtual AT port development and debugging on application.

3.1. Development Example on Application

3.1.1. Virtual AT Port Demo Description

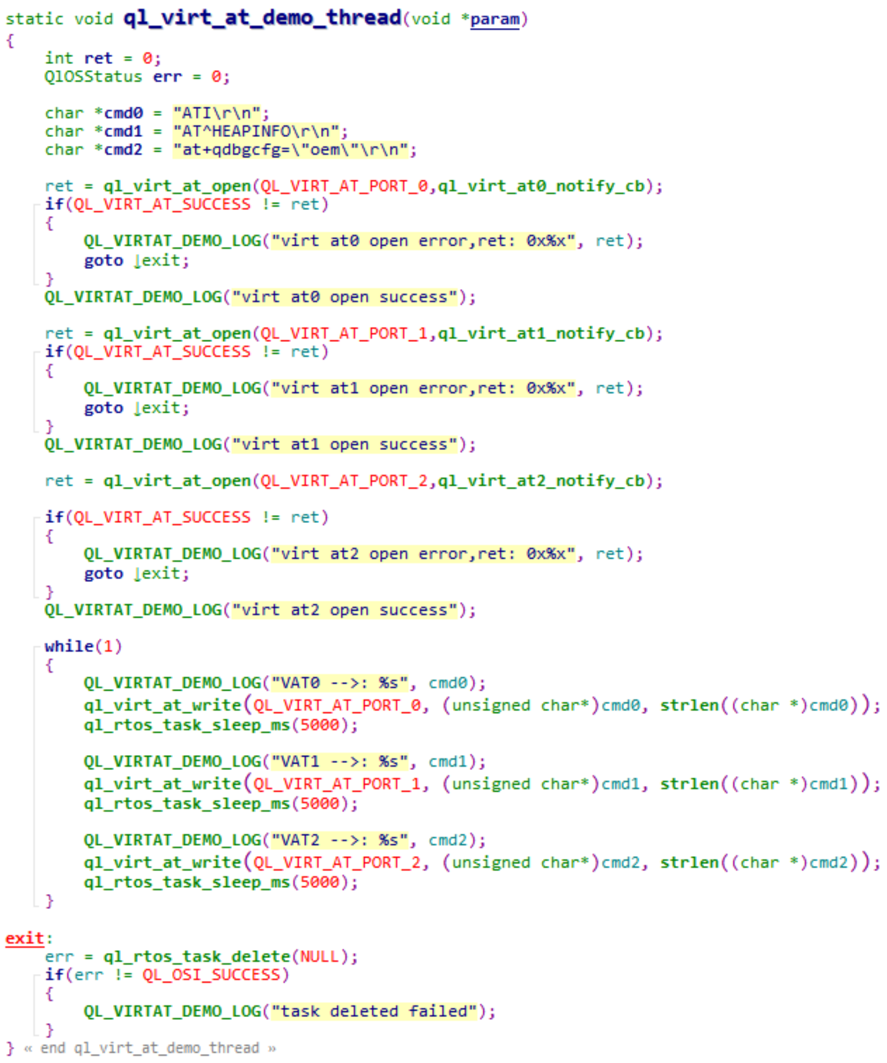
virt_at_demo.c, the example file of virtual AT port, is provided in Quectel EC200U series QuecOpen SDK code.

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()
{
    QlOSStatus 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;
    }
}
```

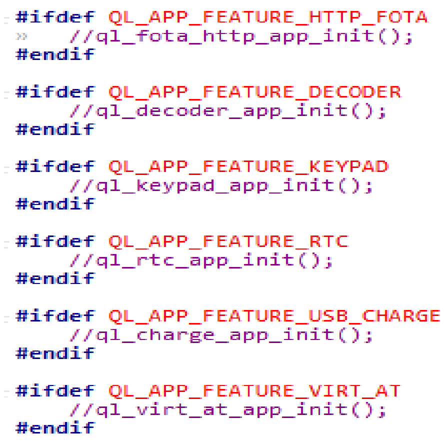
The Demo first enables three virtual AT ports, registers the corresponding callback functions, then writes AT commands through the three virtual AT ports, and reads AT commands responses through the callback function.



```
static void ql_virt_at_demo_thread(void *param)
{
    int ret = 0;
    QlOSStatus 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:



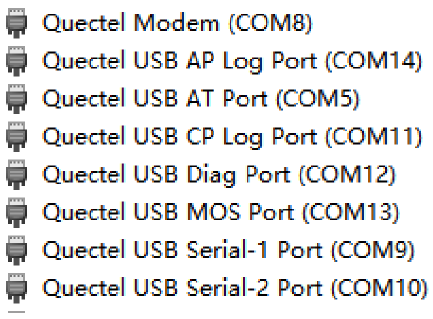
```
#ifdef QL_APP_FEATURE_HTTP_FOTA
    //ql_fota_http_app_init();
#endif

#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.

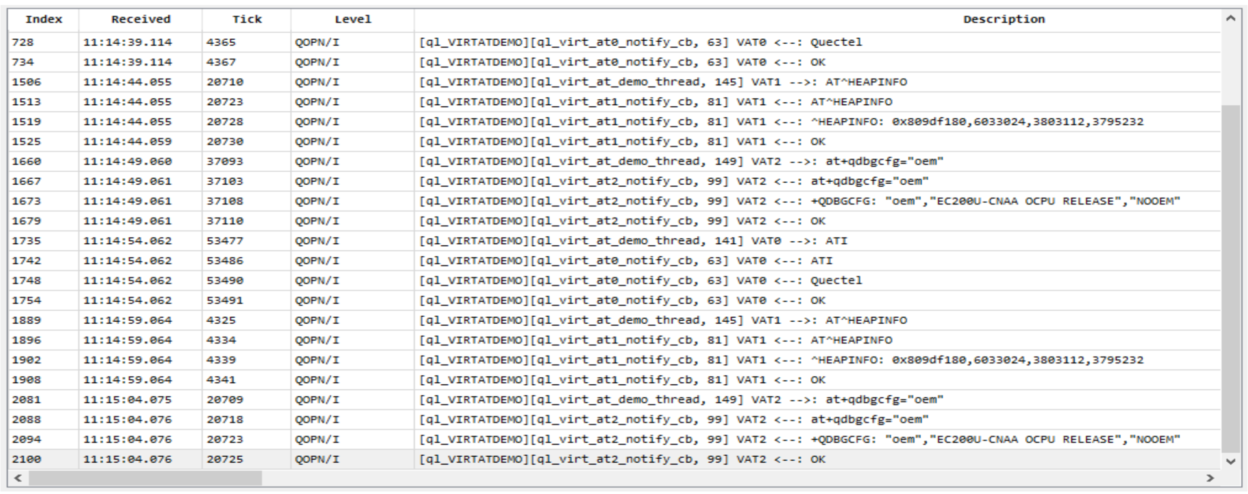


Quectel Modem (COM8)
Quectel USB AP Log Port (COM14)
Quectel USB AT Port (COM5)
Quectel USB CP Log Port (COM11)
Quectel USB Diag Port (COM12)
Quectel USB MOS Port (COM13)
Quectel USB Serial-1 Port (COM9)
Quectel USB Serial-2 Port (COM10)

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-CNAA 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-CNAA 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