EC2x&EG9x&EG25-G Series

QuecOpen CM256SM BT Application Note

LTE Standard Module Series

Version: 1.0.0

Date: 2020-11-05

Status: Preliminary

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company 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 office. 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 to support@quectel.com.

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

DISCLAIMER

WHILE QUECTEL HAS MADE EFFORTS TO ENSURE THAT THE FUNCTIONS AND FEATURES UNDER DEVELOPMENT ARE FREE FROM ERRORS, IT IS POSSIBLE THAT THESE FUNCTIONS AND FEATURES COULD CONTAIN ERRORS, INACCURACIES AND OMISSIONS. UNLESS OTHERWISE PROVIDED BY VALID AGREEMENT, QUECTEL MAKES NO WARRANTIES OF ANY KIND, IMPLIED OR EXPRESS, WITH RESPECT TO THE USE OF FEATURES AND FUNCTIONS UNDER DEVELOPMENT. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUECTEL EXCLUDES ALL LIABILITY FOR ANY LOSS OR DAMAGE SUFFERED IN CONNECTION WITH THE USE OF THE FUNCTIONS AND FEATURES UNDER DEVELOPMENT, REGARDLESS OF WHETHER SUCH LOSS OR DAMAGE MAY HAVE BEEN FORESEEABLE.

COPYRIGHT

THE INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF QUECTEL WIRELESS SOLUTIONS CO., LTD. TRANSMITTING, REPRODUCING, DISSEMINATING AND EDITING THIS DOCUMENT AS WELL AS USING THE CONTENT WITHOUT PERMISSION ARE FORBIDDEN. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

About the Document

Revision History

Version	Date	Author	Description
-	2020-10-05	Young XU	Creation of the document
1.0.0	2020-11-05	Young XU	Preliminary

Contents

Table Index

Figure Index

1 Introduction

Quectel LTE Standard EC2x series, EG9x series and EG25-G modules support QuecOpen® solutions. The document is designed to quickly use the Bluetooth of CM256SM module (Hereinafter referred to as “CM256SM”) on Quectel LTE Standard QuecOpen modules (Hereinafter referred to as “the module”) to achieve related services. The AW-CM256SM IEEE 802.11 a/b/g/n/ac WIFI with Bluetooth 4.2 combo module is a highly integrated wireless local area network (WLAN) solution to let users enjoy the digital content through the latest wireless technology without using the extra cables and cords. It combines with Bluetooth 4.2 and provides a complete 2.4GHz Bluetooth system which is fully compliant to Bluetooth 4.2 and v2.1 that supports EDR of 2Mbps and 3Mbps for data and audio communications. It enables a high performance, cost effective, low power, compact solution that easily fits onto the SDIO and UART combo stamp module.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Module
EC2x series	EC25 series
	EC21 series
	EC20 R2.1
EG9x series	EG95 series
	EG91 series
EG25-G	EG25-G

2 Hardware Information

CM256SM and the module communicate through the four-wire serial port which supports hardware flow control. Besides, the module should reserve another GPIO pin as the power control pin of CM256SM. However, to enable the sleep and wake-up function, the module needs to reserve two additional pins: one pin serves as an input interrupt; the other pin as a common GPIO output pin.

The correspondence between the pins of the module and CM256SM is shown in the following figure:

Table 2: Correspondence between Pins of the Module and CM256SM

Quectel Module		CM256SM Module	
Pin No.	**Definition**	**Pin No.**	**Definition**
2	GPIO_10 (Output)	6	BT_WAKE (Input)
3	GPIO_42 (Input)	7	BT_HOST_WAKE (Output)
139	PMU_GPIO1019 (Output)	34	BT_REG_ON (Input)
37	UART_RTS_BLSP6/GPIO_22 (Input)	41	BT_UART_RTS_N (Output)
38	UART_RXD_BLSP6/GPIO_21 (Input)	42	BT_UART_TXD (Output)
39	UART_TXD_BLSP6/GPIO_20 (Output)	43	BT_UART_RXD (Input)
40	UART_CTS_BLSP6/GPIO_23 (Output)	44	BT_UART_CTS_N (Input)

3 Bluetooth Function Configuration

3.1. Match the Pin

According to ***Table 1***, BLSP needs to be multiplexed as UART, and that BT_REG_ON pin should be set as GPIO output, BT_HOST_WAKE pin as interrupt input, and BT_WAKE as GPIO output.

3.1.1. Configure BLSP to UART6

The module has 6 BLSP controllers, all of which can be reused as UART/I2C/SPI peripheral interfaces. It should be noted that UART/I2C/SPI peripheral device interfaces cannot be used simultaneously. These interfaces support RTS/CTS. If BLSP is multiplexed as UART6 for Bluetooth communication, please disable spi_6 and enable blsp1_uart6 in the *mdm9607-mtp.dtsi* file.

```
&spi_6 {
-     status = "ok";
+     status = "disabled";
};
&blsp1_uart6 {
-     status = "disabled";
+     status = "ok";
};
```

3.1.2. Set BT_REG_ON Pin

QuecOpen SDK supports configuring the module pins as common GPIOs through the application layer for data output. Please configure the GPIOs according to the module's hardware design. For detailed information, please refer to ***Chapter 3.3.1***.

3.1.3. Set BT_HOST_WAKE and BT_WAKE Files

The BSA protocol stack requires that the GPIO IDs corresponding to BT_HOST_WAKE and BT_WAKE be written to the files in the paths */sys/class/bluetooth/wake_gpios/host_wake* and */sys/class/bluetooth/wake_gpios/dev_wake*. Therefore, the readable and writable files host_wake and dev_wake need to be created can read and write files.

Modify the hci_sysfs.c in the path *ql-ol-sdk/ql-ol-kernel/net/bluetooth/*, add the code as follows:

```
EC20CEFAG/ql-ol-sdk/ql-ol-kernel/net/bluetooth/hci_sysfs.c
2018-03-07 20:28:15.000000000 +0800
+++ hci_sysfs.c    2018-02-05 22:54:06.000000000 +0800
@@ -6,6 +6,10 @@
 #include <net/bluetooth/hci_core.h>

 static struct class *bt_class;
+static struct device *bt_wakeup_gpio;
+
+static struct device_attribute dev_wake;
+static struct device_attribute host_wake;

 static inline char *link_typetostr(int type)
 {
@@ -201,9 +205,70 @@
     device_initialize(dev);
 }

+
+static int dev_wake_value = -1;
+
+static ssize_t
+show_dev_wake(struct device *dev, struct device_attribute *attr, char *buf)
+{
+    return snprintf(buf, PAGE_SIZE, "%d\n", dev_wake_value);
+}
+
+static ssize_t
+store_dev_wake(struct device *dev, struct device_attribute *attr,
+           const char *buf, size_t count)
+{
+    kstrtou32(buf, 0, &dev_wake_value);
+    return count;
+}
+
+static int host_wake_value = -1;
+
+static ssize_t
+show_host_wake(struct device *dev, struct device_attribute *attr, char *buf)
+{
+    return snprintf(buf, PAGE_SIZE, "%d\n", host_wake_value);
+}
```

```
+
+static ssize_t
+store_host_wake(struct device *dev, struct device_attribute *attr,
+            const char *buf, size_t count)
+{
+    kstrtou32(buf, 0, &host_wake_value);
+    return count;
+}
+
+
 int __init bt_sysfs_init(void)
 {
+       int ret = -1;
+
      bt_class = class_create(THIS_MODULE, "bluetooth");
+             bt_wakeup_gpio = device_create(bt_class, NULL, 0, NULL, "wake_gpios");
+    if (IS_ERR(bt_wakeup_gpio))
+            printk("[Quinn] Create Failed\n");
+
+    dev_wake.show = show_dev_wake;
+    dev_wake.store = store_dev_wake;
+    sysfs_attr_init(&dev_wake.attr);
+    dev_wake.attr.name = "dev_wake";
+    dev_wake.attr.mode = S_IRUGO | S_IWUSR;
+    ret = device_create_file(bt_wakeup_gpio, &dev_wake);
+             if (ret) {
+            printk("[Quinn] Create dev_wake Failed\n");
+            device_remove_file(bt_wakeup_gpio, &dev_wake);
+    }
+
+    host_wake.show = show_host_wake;
+    host_wake.store = store_host_wake;
+    sysfs_attr_init(&host_wake.attr);
+    host_wake.attr.name = "host_wake";
+    host_wake.attr.mode = S_IRUGO | S_IWUSR;
+    ret = device_create_file(bt_wakeup_gpio, &host_wake);
+             if (ret) {
+            printk("[Quinn] Create dev_wake Failed\n");
+            device_remove_file(bt_wakeup_gpio, &dev_wake);
+    }

      return PTR_ERR_OR_ZERO(bt_class);
 }
```

3.1.4. Set BT_HOST_WAKE Pin

QuecOpen SDK supports configuring the module pins as interrupts through the application layer. Please modify file *ql_cm256sm_ble_sleep.c* in *ql-ol-sdk/ql-ol-extsdk/example/bt/cm256sm/app_ble_test/source* path to the pins used by the module's hardware. Please see the example as follows:

```
#if defined(__QUECTEL_PROJECT_AG35C__) || defined(__QUECTEL_PROJECT_AG35CE__) \
    || defined(__QUECTEL_PROJECT_AG35CEN__)|| defined(__QUECTEL_PROJECT_AG35CEVBM__)
static Enum_PinName BT_HostWakePin = PINNAME_GPIO3;
#else ←—LTE Standard QuecOpen modules
static Enum_PinName BT_HostWakePin = PINNAME_GPIO3;
#endif
```

3.1.5. Set BT_WAKE Pin

QuecOpen SDK supports configuring the module pins as common GPIOs through the application layer for data output. Please configure the GPIOs according to the module's hardware design. For detailed information, please refer to ***Chapter 3.3.2***.

3.2. Function of Waking up from Sleep

CM256SM uses two pins to realize the function of waking up from sleep, respectively:
BT_HOST_WAKE: This pin is an interrupt input pin for the module; it is an output pin for CM256SM.
BT_WAKE: This pin is a common output pin for the module; it is an input pin for CM256SM.

3.2.1. BT_HOST_WAKE Function

When there is no data interaction, BT_HOST_WAKE is in high level, allowing the module to sleep. When the mobile phone sends a request of reading and writing data, BT_HOST_WAKE will be pulled low by CM256SM. The module applies for the wakelock according to the interrupt and keeps it awake. After the phone disconnects from the CM256SM, the module will release the wakelock and then can go to sleep.

3.2.2. BT_HOSTWAKE Function

When there is no data interaction, BSA protocol stack will pull BT_WAKE pin to high, allowing the CM256SM to sleep. When the module needs to interact with CM256SM, the BSA protocol stack will pull down the BT_WAKE pin to wake up the CM256SM. For the introduction of BSA, please refer to ***Chapter 3.3***.

NOTE

The basis for CM256SM to determine whether it can send data to the module is CTS level, that is, if RTS level of the module is low, CM256SM will send data to the module. Please ensure that the module's RTS output is high in the sleep state to avoid data loss.

3.3. BSA Protocol Stack

The BSA (Bluetooth Simple API) protocol stack is based on the classic C/S structure and is designed to simplify the Bluetooth function. The executable program bsa_server running in the background of the module exists as a server in the BSA protocol stack. It is a Bluetooth protocol stack, which implements a variety of commonly used protocol stacks and profiles, and performs data interaction with the CM256SM based on HCI commands through the serial ports of LTE standard QuecOpen modules. One or more client programs can be written according to specific needs to communicate with bsa_server through the socket file.

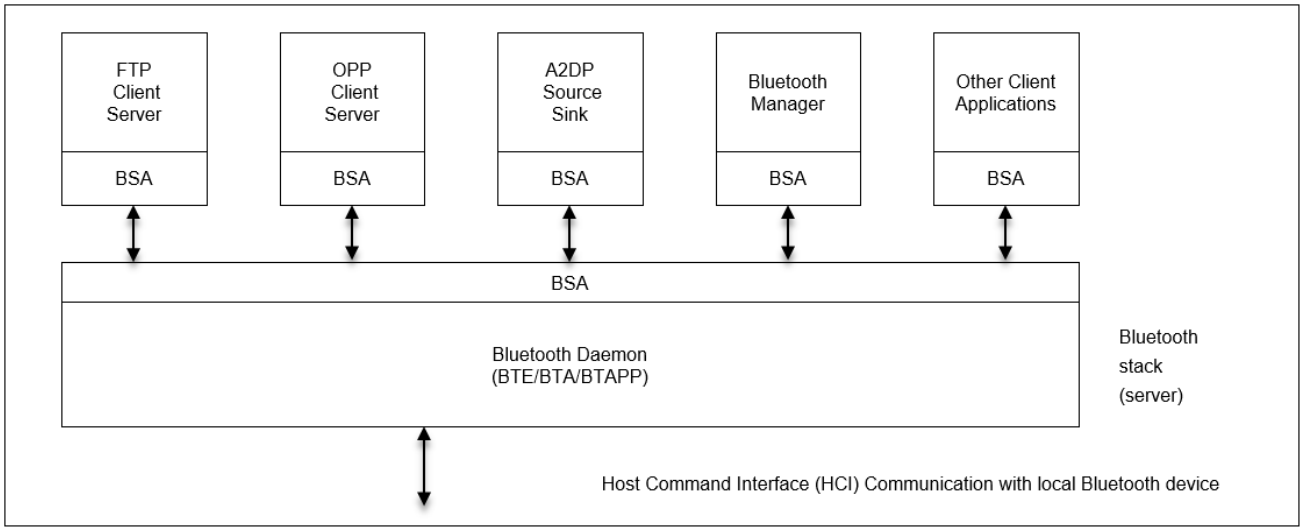

Figure 1: BSA Framework

Step 1: Power on and Reset.

The module needs to pull BT_REG_ON to high/low according to the power-on timing requirements of CM256SM. The specific process is: pull down BT_REG_ON → hold for 1–3 seconds → pull up BT_REG_ON. The script example is as follows:

```
echo 1019 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio1019/direction
echo 0 > /sys/class/gpio/gpio1019/value
```

```
sleep 3
echo 1 > /sys/class/gpio/gpio1019/value
```

NOTE

It is recommended to select the corresponding pins according to the hardware design of the module. If the actual circuit design changes, please change the connected circuit accordingly.

Step 2: Configure Pins for Module Wakeup.

BSA protocol stack writes the GPIO IDs of BT_HOST_WAKE and BT_WAKE into the files of *host_wake* and *dev_wake* respectively. The initialization state of the BT_WAKE pin should be set to low level, the example is as follows:

```
echo 10 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio10/direction
echo 0 > /sys/class/gpio/gpio10/value
echo 10 > /sys/class/bluetooth/wake_gpios/dev_wake
echo 42 > /sys/class/gpio/export
echo 42 > /sys/class/bluetooth/wake_gpios/host_wake
```

Step 3: Run bsa_server.

Please refer to README in SDK path *ql-ol-sdk/ql-ol-extsdk/example/bt/cm256sm* and import bsa_server and BCM434545.hcd files into the module.

bsa_server supports inputting multiple parameters to achieve different functions. Run **./bsa_server –hep** command to get detailed description of the parameters. Example commands are as follows:

```
./bsa_server -d /dev/ttyHSL1 -p BCM434545.hcd -lpm -b /tmp/btsnoop_lpm.cfa -all=6 >
/tmp/bsa_server_lpm.log &
```

- **-d**: device path and name of the communication serial port used by the module and CM256SM. Please fill in according to the actual hardware and driver design.
- **-p**: specifies the absolute path where the Bluetooth firmware is located
- **-lpm**: turn on the low-power mode
- **-b** and **-a**ll: both related to log collection

After bsa_server runs normally, a socket file named *bt-daemon-socket* will be generated. If the file is not generated, please check the hardware design and pin configuration again.

Step 4: Run bsa_client.

Please develop bsa_client application according to actual requirements. For details, please refer to the ***document [1]***.

Establish the connection with bsa_client through the generated socket file. Quectel provides the demo program. Please refer to README in SDK path *ql-ol-sdk/ql-ol-extsdk/example/bt/cm256sm*.

4 Functional Verification

Bluetooth Low Energy (BLE) function in a mobile phone is turned off by default. Therefore, the mobile phone needs to download dedicated software for functional verification. If it is an Android system, please download BLE Deng in the application store or search engine. If it is an IOS system, please download LightBlue in the application store. This chapter uses the BLE Deng under the Android system as an example for detailed introduction.

4.1. Basic Function Verification

Please turn on the Bluetooth function of the mobile phone and open the mobile phone application BLE Deng.

4.1.1. Scanning Verification

After opening BLE Deng, BLE Deng will automatically start scanning the Bluetooth devices nearby. As shown in the screenshot below, the mobile phone has found CM256SM.

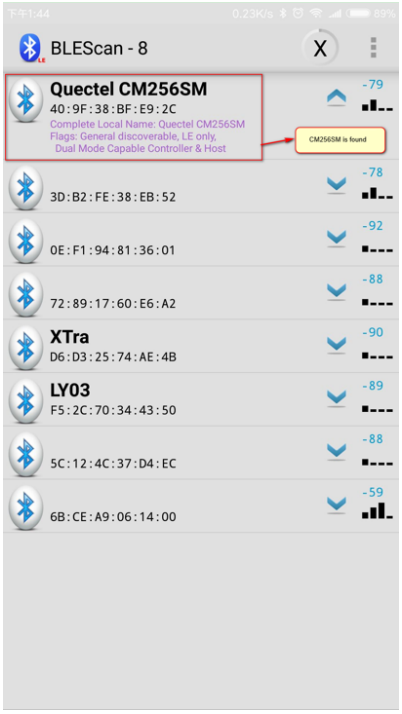

Figure 2: Scan BT Devices

4.1.2. Connecting Verification

Click on the **"Quectel CM256SM**" scanned in the screenshot above to connect to the CM256SM. After the connection is established, the program will perform "Service Discovery" and scan all services supported by CM256SM. As shown below:

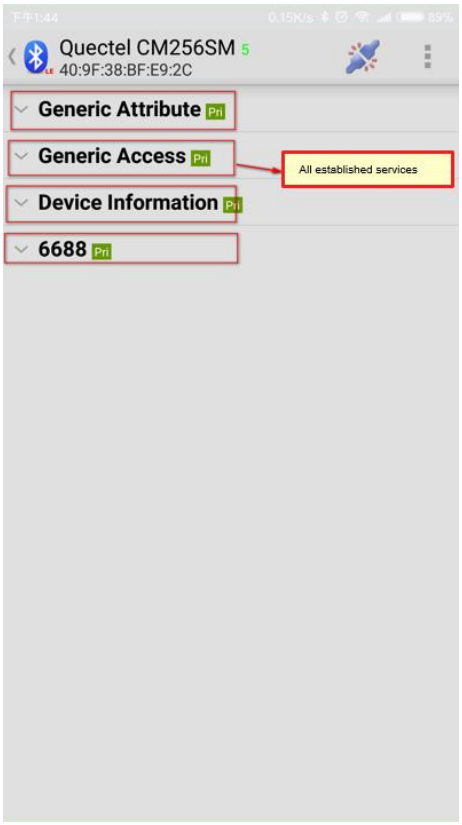

Figure 3: Connect to CM256SM

4.1.3. Interacting Verification

Select the service named "**Device Information**" and read the characteristic data of "Firmware Revision String". As shown below:

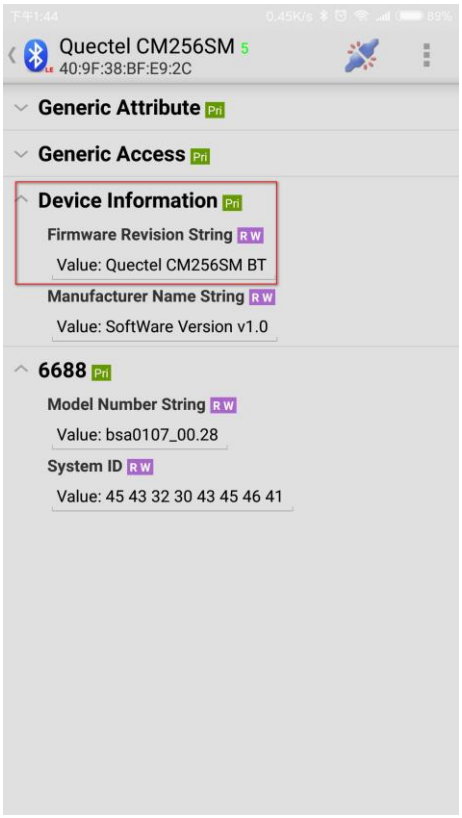

Figure 4: Read Characteristic Data of Firmware Revision String

4.2. Varification of Waking up from Sleep

Before testing the wake-up function, please insert the Bluetooth antenna for good signal quality.

4.2.1. Verification of Waking up the Module by CM256SM

When the AutoSleep function is enabled, the module enters the sleep mode, and the serial port will not be able to perform data interaction. For AutoSleep function, please refer to ***document [4]***.

At this time, the mobile phone can still scan CM256SM. When the mobile phone initiates the connection with CM256SM, CM256SM will pull down BT_HOST_WAKE to generate an interrupt and wake up the module. After the mobile phone is disconnected from CM256SM, if no other task holds the wakelock, the module will enter sleep mode, and its serial port cannot perform data interaction.

4.2.2. Verification of Waking up CM256SM by the Module

When there is no data interaction, the BSA protocol stack will pull up the BT_WAKE pin, allowing CM256SM to sleep. When the module needs to transmit data with CM256SM, the BSA protocol stack will

pull down the BT_WAKE pin to wake up the CM256SM. An oscilloscope can monitor the level change of BT_WAKE pin, test the current consumption of CM256SM, and verify whether the CM256SM sleeps and wakes up normally in real time.

5 Appendix A References

Table 3: Related Documents

SN	Document Name	Remark
[1]	BSA_API_Programming_Guide	BSA protocol stack API programming guide
[2]	Quectel_EC20_R2.1-QuecOpen_Hardware_Design	Quectel EC20 R2.1 QuecOpen hardware design
[3]	AW-CM256SM_DS_Rev 14_CYW	CM256SM WIFI/BT module specification
[4]	Quectel_EC2x&EG9x&EG25-G_QuecOpen_Low_Power_Mode_Application_Note	EC2x&EG9x&EG25-G QuecOpen low power mode application note

Table 4: Terms and Abbreviations

Abbreviation	Description
BLE	Bluetooth Low Energy
BLSP	BAM Low-Speed Peripherals
C/S structure	Client/Server Structure
CTS	Clear To Send,
GPIO	General-purpose input/output
HCI	Host Controller Interface
I2C	Inter-Integrated Circuit
RTS	Require To Send
SDK	Software Development Kit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver & Transmitter