FC41D Getting Started Guide for FreeRTOS

Wi-Fi&Bluetooth Module Series

Version: 1.0.0

Date: 2022-04-20

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

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
-	2022-04-20	Brave LIU	Creation of the document
1.0.0	2022-04-20	Brave LIU	Preliminary

Contents

Figure Index

1 Overview

FC41D is a highly integrated dual-mode Bluetooth 5.2 and Wi-Fi 802.11n combo solution with complete hardware and software resources needed for Wi-Fi and Bluetooth applications. It integrates Bluetooth Classic i.e. Basic Rate (BR) and Enhanced Data Rate (EDR) as well as Bluetooth Low Energy (BLE) features fully compliant with the Bluetooth 5.2 specification. It supports AP and Station mode concurrently.

A 32-bit MCU running up to 120 MHz and built-in 256 KB RAM enable the chip to support multi-cloud connectivity. The MCU's extended instructions specifically for signal processing allow for efficient audio encoding and decoding.

FC41D includes a rich set of peripherals, such as PWM, I2C, UART for program download and burning, SPI, SDIO and IrDA. FC41D can provide up to six 32-bit high-speed PWM channels for high-quality LED control. Each of the two PWM channels can be configured as a phase-controlled differential mode to support motor and strip drive.

FC41D integrates a Packet Traffic Arbitration (PTA) interface to facilitate coexistence of Bluetooth and 802.11 WLAN.

FC41D embeds eFUSE and supports OTP read/write in Flash, which can be used to provide unique ID, code encryption and secure the debug interface. A built-in True Random Number Generator (TRNG) and a security module are integrated to ensure secure communication and fast authentication and network connectivity.

FC41D supports low power sleep modes where the MCU can enters sleep modes with a micro amp level. In deep sleep mode, the chip can run a 32-bit clock with a few microamperes of current and can be woken up by this clock or by any GPIO.

FC41D supports Bluetooth Classic i.e. Basic Rate (BR) and Enhanced Data Rate (EDR) as well as all Bluetooth LE 5.2 rates and features, including Long Range, High Data Rate, and angle-of-arrival (AoA) and angle-of-departure (AoD) positioning with up to four antennas.

2 Hardware Description

2.1. Datasheet

https://github.com/bekencorp/bk7231n_freertos_aws/blob/master/projects/beken/doc/BK7231N%20Datasheet%20v1.0.pdf

2.2. Schematic

https://github.com/bekencorp/bk7231n_freertos_aws/blob/master/projects/beken/doc/SCH-BK7231X_QFN32_4X4_MODULE_V3.pdf

2.3. Key Components

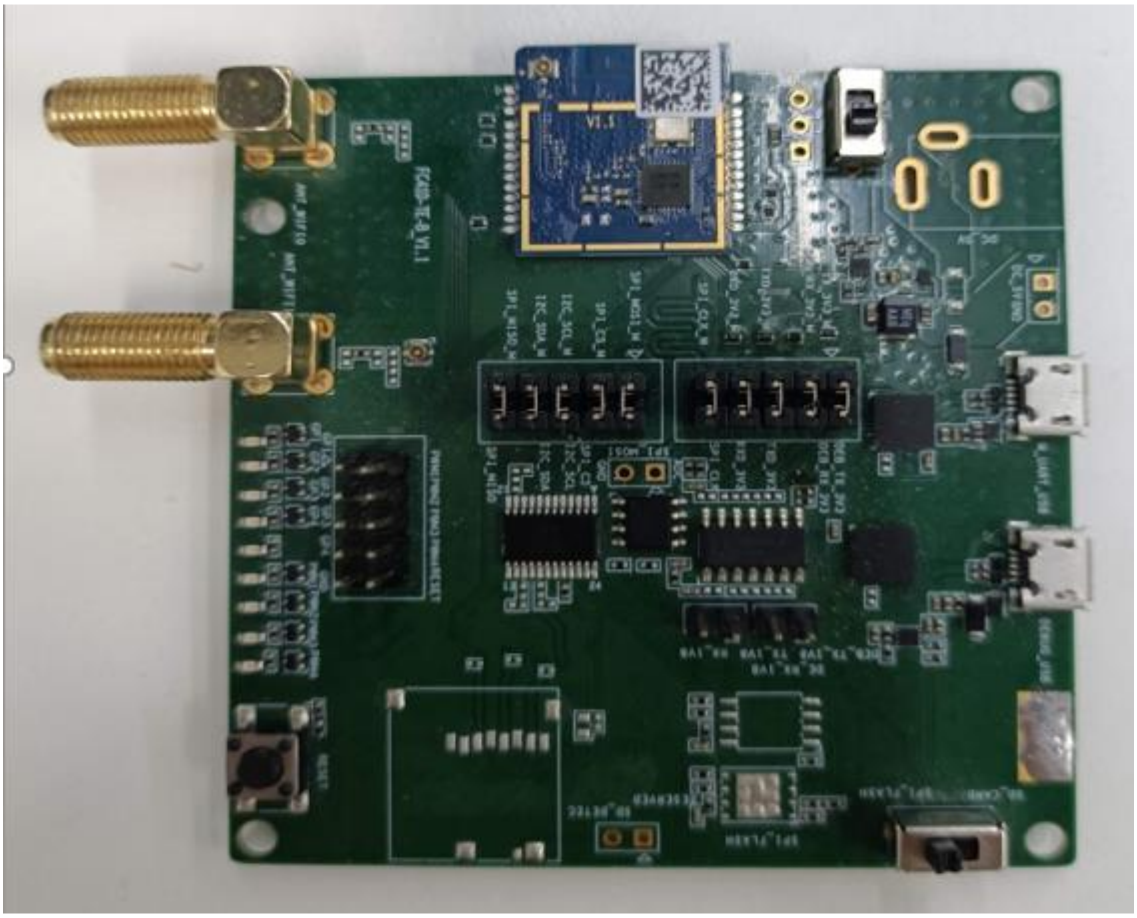


Figure 1: Key Component

2.4. Hardware Requirement to Run FreeRTOS Demo

2.4.1. Standard Kit Contents

a. FC41D demo board
b. USER-Serial cable (CH430)

2.4.2. User Provided Items

None

2.4.3. 3rd Party Purchasable Items

None

2.5. Additional Hardware References

See http://www.bekencorp.com/index/goods/detail/cid/13.html for details.

3 Set Up Development Environment

3.1. Supported IDEs

CLI based (CMake)
Under *bk7231_freertos_aws\sdk* directory, follow below steps to generate demo target:
a. Execute CMD: **generate_make.bat**, the script contains below command.
(**cmake -DVENDOR=beken -DBOARD=FC41D-DCOMPILER=arm-gcc -DAFR_ENABLE_TESTS=0 -DAFR_METADATA_MODE=1 -S. -Bbuild -G"Unix Makefiles"**)
b. Go to the build directory and execute **make** command. Thus can build out the image.

3.2. Toolchains

a. cmake: https://github.com/Kitware/CMake/releases/download/v3.22.0-rc2/cmake-3.22.0-rc2-windows-x86_64.msi
b. toolchain: https://developer.arm.com/-/media/Files/downloads/gnu-rm/10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-win32.exe
c. GNU make: https://sourceforge.net/projects/gnuwin32/

3.3. SDKs

None

3.4. Establishing a Serial Connection

a. Get the CH340 USB-Serial driver (CH341USB-Serial-driver.exe, which is a very common driver) from *projects/beken/tools* directory and double-click it to install the driver for the serial tool.
b. Connect the serial cable with the UART1 on the demo board as below (baud rate: 115200)

Figure 2: Establish a Serial Connection

3.5. Other Software Required to Develop and Debug Applications for the Device

None

3.6. Other Pre-requisites

None

3.7. Additional Software References

None

4 Set Up Hardware

UART1 can supply power and download image to board, so just connect the serial cable to UART1.

5 Setup AWS account and Permissions

To create an AWS account, see *Create and Activate an AWS Account*.

To add an IAM user to your AWS account, see *IAM User Guide*. To grant your IAM user account access to AWS IoT and FreeRTOS, attach the following IAM policies to your IAM user account:

- AmazonFreeRTOSFullAccess
- AWSIoTFullAccess

To attach the AmazonFreeRTOSFullAccess policy to your IAM user:

1. Browse to the *IAM console*, and from the navigation pane, choose **Users**.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter AmazonFreeRTOSFullAccess, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

To attach the AWSIoTFullAccess policy to your IAM user:

1. Browse to the *IAM console*, and from the navigation pane, choose **Users**.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter AWSIoTFullAccess, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

NOTE

The examples in this document are intended only for dev environments. All devices in your fleet must have credentials with privileges that authorize only intended actions on specific resources. The specific permission policies can vary for your use case. Identify the permission policies that best meet your business and security requirements. For more information, refer to *Example policies* and *Security Best practices*.

For more information about IAM and user accounts, see *IAM User Guide*.
For more information about policies, see *IAM Permissions and Policies*.

6 Provision the device with AWS IoT

Refer to *Register your board manually*.

Follow ***steps 1–6*** under the heading **To create an AWS IoT policy**. In step 1, note that the AWS region for your account can also be found in the drop-down between the account name and Support drop-downs in the top menu bar.

Follow ***steps 1-10*** under the heading **To create an IoT thing, private key, and certificate for your device**.

7 Download FreeRTOS

You can download the code from https://github.com/bekencorp/bk7231n_freertos_aws.git.

8 Configure FreeRTOS

Follow the instructions under the heading **Configuring the FreeRTOS Demo**.

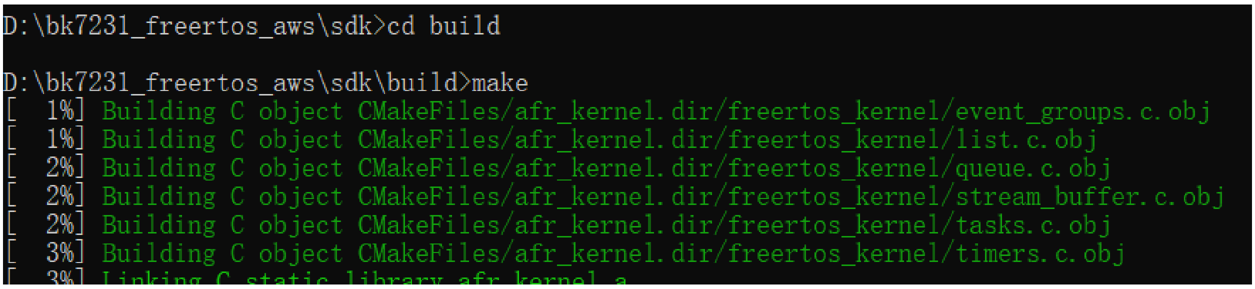
Actually we only need to configure following things:

1. Need to acquire the key and certificate from AWS IoT (refer to above instructions)
2. generate the *demos\include\aws_clientcredential_keys.h*
3. Need to define the device thing name and endpoint, and Wi-Fi client credentials in the *demos\include\aws_clientcredential.h*.

9 Build the FreeRTOS Demo

1. Execute the script: **generate_make.bat**, which will generate one build folder automatically.

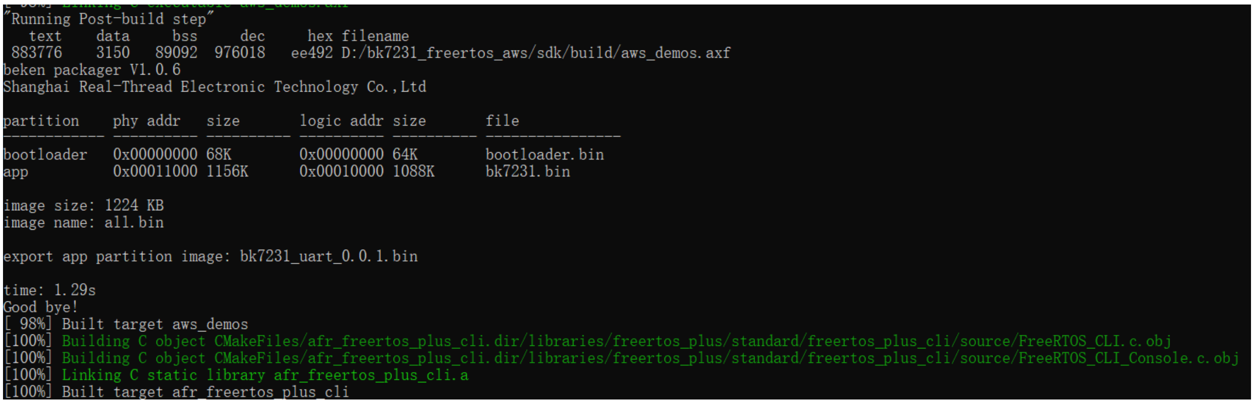
2. cd build folder, and execute command: **make**, it will make all objects.



```
D:\bk7231_freertos_aws\sdk>cd build

D:\bk7231_freertos_aws\sdk\build>make
[  1%] Building C object CMakeFiles/afr_kernel.dir/freertos_kernel/event_groups.c.obj
[  1%] Building C object CMakeFiles/afr_kernel.dir/freertos_kernel/list.c.obj
[  2%] Building C object CMakeFiles/afr_kernel.dir/freertos_kernel/queue.c.obj
[  2%] Building C object CMakeFiles/afr_kernel.dir/freertos_kernel/stream_buffer.c.obj
[  2%] Building C object CMakeFiles/afr_kernel.dir/freertos_kernel/tasks.c.obj
[  3%] Building C object CMakeFiles/afr_kernel.dir/freertos_kernel/timers.c.obj
```

3. The final target image is: *all.bin*.



```
"Running Post-build step"
   text    data     bss     dec     hex filename
 883776    3150   89092  976018   ee492 D:/bk7231_freertos_aws/sdk/build/aws_demos.axf
beken packager V1.0.6
Shanghai Real-Thread Electronic Technology Co.,Ltd

partition    phy addr   size       logic addr size       file
------------ ---------- ---------- ---------- ---------- ----------------
bootloader   0x00000000 68K        0x00000000 64K        bootloader.bin
app          0x00011000 1156K      0x00010000 1088K      bk7231.bin

image size: 1224 KB
image name: all.bin

export app partition image: bk7231_uart_0.0.1.bin

time: 1.29s
Good bye!
[ 98%] Built target aws_demos
[100%] Building C object CMakeFiles/afr_freertos_plus_cli.dir/libraries/freertos_plus/standard/freertos_plus_cli/source/FreeRTOS_CLI.c.obj
[100%] Building C object CMakeFiles/afr_freertos_plus_cli.dir/libraries/freertos_plus/standard/freertos_plus_cli/source/FreeRTOS_CLI_Console.c.obj
[100%] Linking C static library afr_freertos_plus_cli.a
[100%] Built target afr_freertos_plus_cli
```

10 Run the FreeRTOS Demo Project

After building FreeRTOS demo and get the image successfully, open the flash tool *projects\beken\tools\bk_writer_v6.13.exe*.

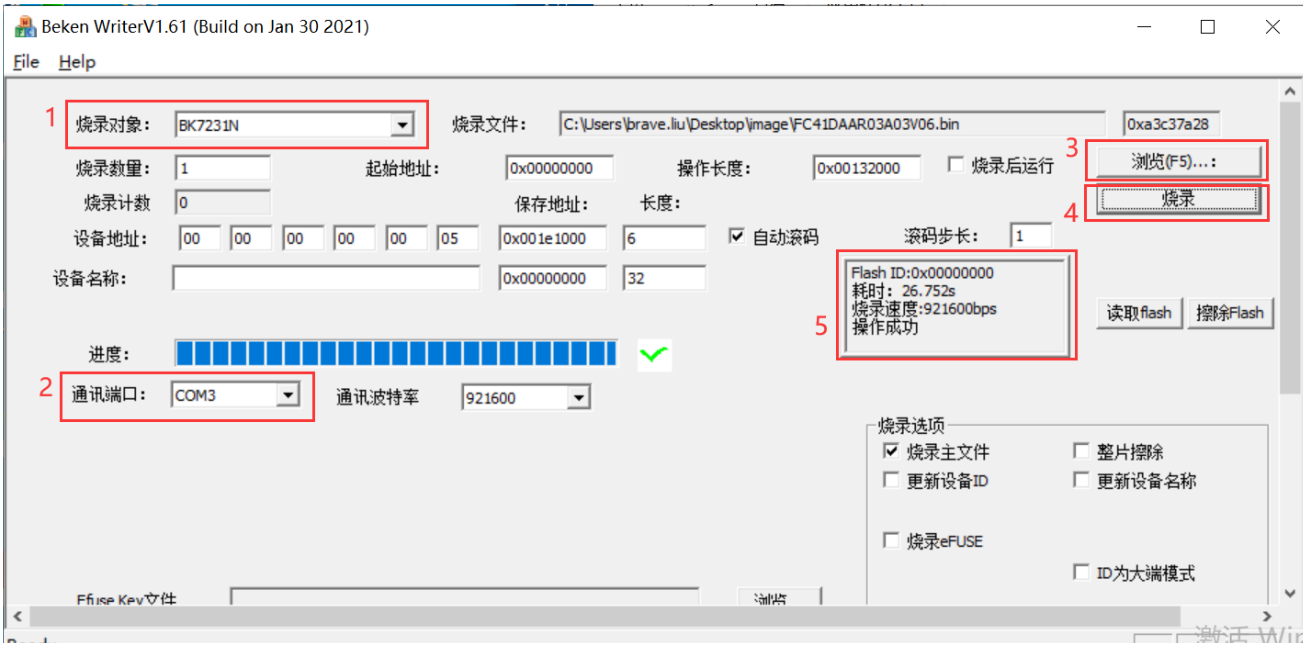


Figure 3: Flash Tool

You can follow the below steps to flash image to board:

1. Choose the chip type: BK7231N
2. Choose the serial port, e.g, COM3
3. Click "**浏览(F5)…：** " to select the image file to download, e.g, *all.bin*
4. Click "烧录" to download image to board. Note that after click the "烧录" button, switch VCC "**ON-OFF-ON**" quickly. Do not need to click "读取 **flash**" and "擦除 **flash**", both are for debug.
5. After more than ten seconds, you can see the "操作成功".

To verify MQTT messages are being received by AWS IoT Core:

From the AWS IoT Core console at *console.aws.amazon.com/iot*, select **Test** from the navigation pane, and choose MQTT test client. Choose **Subscribe to a topic** and Enter the topic (or use the # wildcard to see all topics), then choose **Subscribe**. You should see messages being received. Below is the MQTT demo example message exported from AWS IoT core:

https://github.com/bekencorp/bk7231n_freertos_aws/blob/master/projects/beken/doc/subscription.json.

Also you can close this tool, and open one serial tool, then you can see the demo project logs, below is an example:
https://github.com/bekencorp/bk7231n_freertos_aws/blob/master/projects/beken/doc/demo%20console%20log.txt.

11 Debugging

When connecting the demo board with computer via serial tool, switch VCC to ON.

Figure 4: Switch VCC to ON

You should see following initial logs on the serial console:

```
prvHeapInit-start addr:0x409950, size:157360
[Flash]id:0xeb6015
--write status reg:4004,2--
[Flash]init over
sctrl_sta_ps_init
SDK Rev: 3.0.36
cset:0 0 0 0
No TLV header found in flash
bandgap_calm_in_efuse=0x6b
[load]bandgap_calm=0x20->0x2b,vddig=4->5
[FUNC]rwnxl_init
```

```
IP Rev: W4-3.0.36-P0
[bk]tx_txdesc_flush
[FUNC]intc_init
[FUNC]calibration_main
get rfcali_mode:1
device_id=0x20521010
calibration_main over
NO TXPWR_TAB_TAB found in flash
Load default txpwr for b:0xd6b74
Load default txpwr for g:0xd6b82
fit n20 table with dist:4
Load default txpwr for n40:0xd6b90
Load default txpwr for ble:0xd6bd6
NO TXID found in flash, use def temp:330
temp in flash is:330
NO TXID found in flash, use lpf i&q:116, 116
NO TXID found in flash, use def xtal:38
xtal in flash is:38
xtal_cali:38
--init_xtal = 38
[FUNC]ps_init
int watchdog enabled, period=10000
task watchdog enabled, period=60000
[FUNC]func_init_extended OVER!!!

start_type:0
Initializing TCP/IP stack
bk_wlan_app_init finished
rf_thread_init ok
WIFI_On:166 conn_state=0
[sa_sta]MM_RESET_REQ
[bk]tx_txdesc_flush
[sa_sta]ME_CONFIG_REQ
[sa_sta]ME_CHAN_CONFIG_REQ
[sa_sta]MM_START_REQ
PSKC: ssid test, passphrase 12345678
sizeof(wpa_supplicant)=760
mm_add_if_req_handler:0
hapd_intf_add_vif,type:2, s:0, id:0
wpa_dInit
enter low level!
mac c8:47:8c:42:88:48
leave low level!
[net]addvif_idx:0
```

```
wpa_supplicant_req_scan
Setting scan request: 0.000000 sec
wpa_supplicant_scan
wpa_drv_scan
wpa_send_scan_req
no ht in scan
scan_start_req_handler
wpa_driver_scan_start_cb
0 504 [app] WiFi module initialized. Connecting to AP test

wpa_driver_scan_cb
Scan completed in 1.332000 seconds
get_scan_rst_null
WIFI_ConnectAP: conn_state=0 returnCode=1 ssid:test key:1234567      connect the WIFI AP
```

12 Troubleshooting

When IDT call the script to flash image to board to test, the script will start another CMD console automatically to flash image to board, sometimes it will hang on the "**Getting the bus**".

Please switch the VCC quickly "**ON->OFF->ON**"

13 Appendix Terms and Abbreviations

Table 1: Terms and Abbreviations

Abbreviation	Description
AoA	Angle-of-arrival
AoD	Angle-of-departure
AP	Access Point
AWS	Amazon Web Service
BLE	Bluetooth Low Energy
BR	Basic Rate
CLI	Common Layer Interface
CMD	Command
EDR	Enhanced Data Rate
GPIO	General-Purpose Input /Output Ports
I2C	Inter－Integrated Circuit
ID	Identification
IDE	Integrated Development Environment
IDT	Intelligent Data Terminal
IoT	Item of Thing
IrDA	Infra-red Data Association
LED	Light Emitting Diode
MCU	Micro Control Unit
MQTT	Message Queuing Telemetry Transport Protocol

OTP	One Time Programable
PTA	Packet Traffic Arbitration
PWM	Pulse Width Modulation
RAM	Random Access Memory
RTOS	Real-time Operating System
SDIO	Secure Digital Input and Output
SDK	Software Development Kit
SPI	Software Process Improvement
TRNG	True Random Number Generator
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VCC	Virtual Circuit Connection
WLAN	Wireless Local Area Networks