BG96-QuecOpen
Application Note

LTE Module Series

Rev. BG96-QuecOpen_Application_Note_V1.2

Date: 2018-12-26

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.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:

- http://quectel.com/support/sales.htm
-

For technical support, or to report documentation errors, please visit:

http://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.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2018-05-02	Hyman DING	Initial
1.1	2018-08-09	Hyman DING	Add QFLOG function description for TX3.0.1 SDK
1.2	2018-10-17	Hyman DING Egbert XU Ethan YAN	Add QAPI function introduction.
1.3	2018-12-26	Egbert XU Hyman DING	Add QAPI function introduction and GPIO Mapping

Contents

Table Index

Figure Index

1 Introduction

This document mainly introduces how to establish BG96-QuecOpen compiler environment in Windows and Linux Operating Systems, how to compile user application in BG96-QuecOpen SDK and how to run and update user application based on BG96-QuecOpen solution.

2 BG96-QuecOpen Solution Overview

2.1. General Overview

Quectel BG96-QuecOpen provides an infrastructure for applications to dynamically load modules that are built from the resident component of the application. The module is useful for the following scenarios:

- Total application code size exceeds the available memory
- New application modules need to be added after the core image is deployed
- Partial firmware updates are required

Each module is built independently with a common preamble structure attached in the binary. The preamble contains various details about the module, including:

- a single thread entry point
- stack size priority
- module ID
- callback thread stack size/priority, and so on.

2.2. BG96-QuecOpen Architecture

The following diagram shows the architecture of BG96-QuecOpen.

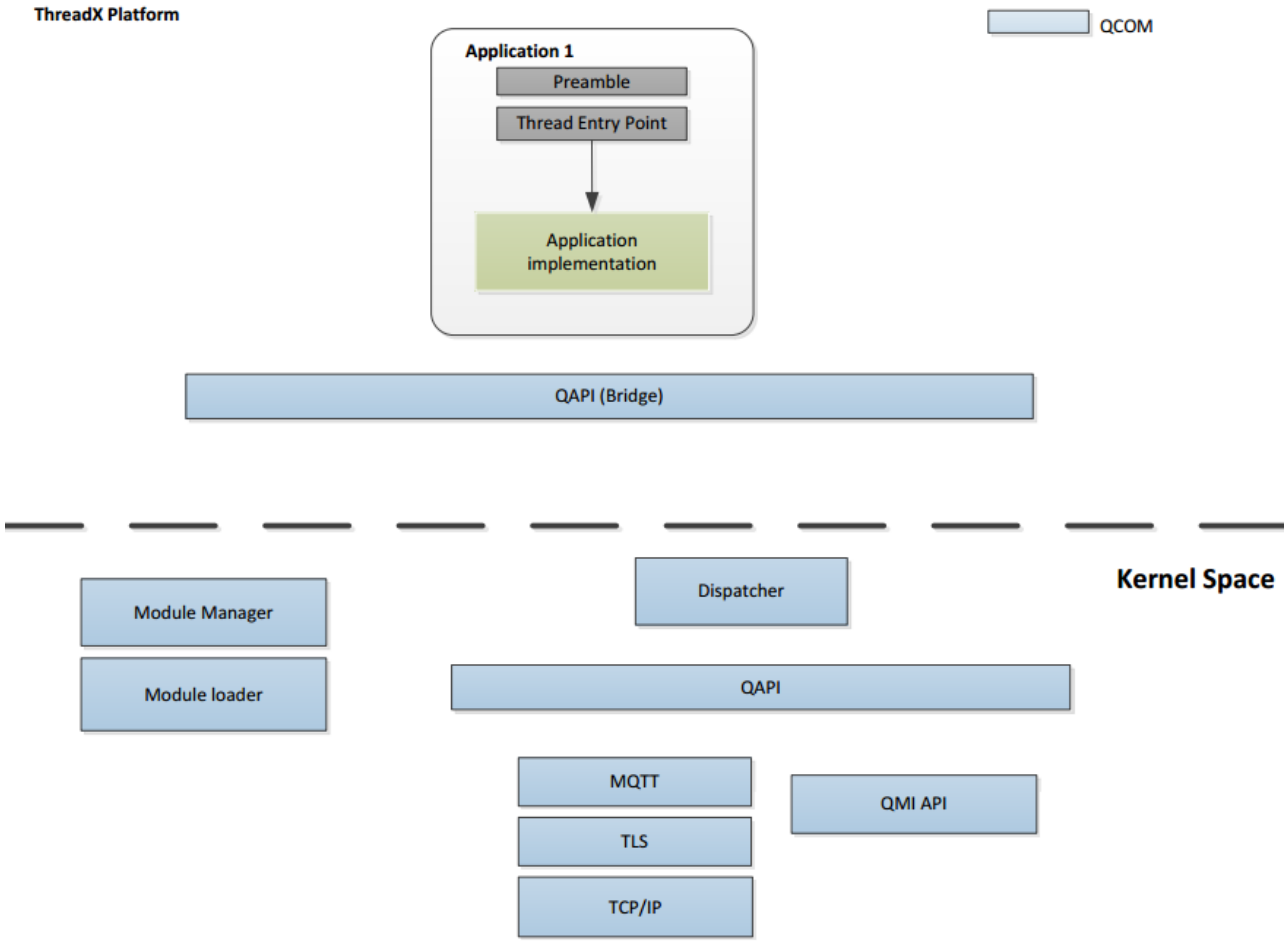

Figure 1: Architecture of BG96-QuecOpen

3 Setup Compiling Environment

Quectel BG96-QuecOpen includes two revisions based on different baseline of MDM9206. One is based on TX2.0 and the other is based on TX3.0.1. Compilation tools are different for different baselines.

While compiling BG96-QuecOpen user application, the host's operating system and compilation tools should meet the requirements shown below.

Table 1: Compiling Environment Requirement for TX2.0

Component	Source or Binary Only	Toolchain Required for Building Source	Cygwin	Supported Build Hosts
QuecOpen SDK	Source	ARM complier tools 5.05 (build 106)	Cygwin 2.8.0	Windows 7

Table 2: Compiling Environment Requirement for TX3.0.1

Component	Source or Binary Only	Toolchain Required for Building Source	Cygwin	Python	Supported Build Hosts
QuecOpen SDK	Source	LLVM 4.0.3	Cygwin 2.8.0	Python 2.7	Windows 7/ Linux

NOTE

Licensed users can download LLVM compiler through the Qualcomm ChipCode™ portal. Customers also can request this tool from Quectel.

3.1. Setup Compiling Environment for TX2.0

3.1.1. Download and Install ARM Compiler Tool

The following mainly introduces how to download and install ARM complier tool in Windows build environment.

3.1.1.1. Download ARM Compiler Tool

Step 1: Create an account in the following page: https://silver.arm.com.

Step 2: Open the ARM complier tool download page: https://silver.arm.com/browse.

(1) Click "**Downloads**" → "**Development Tools**" → "**DS-5 Development Studio**", as illustrated below:

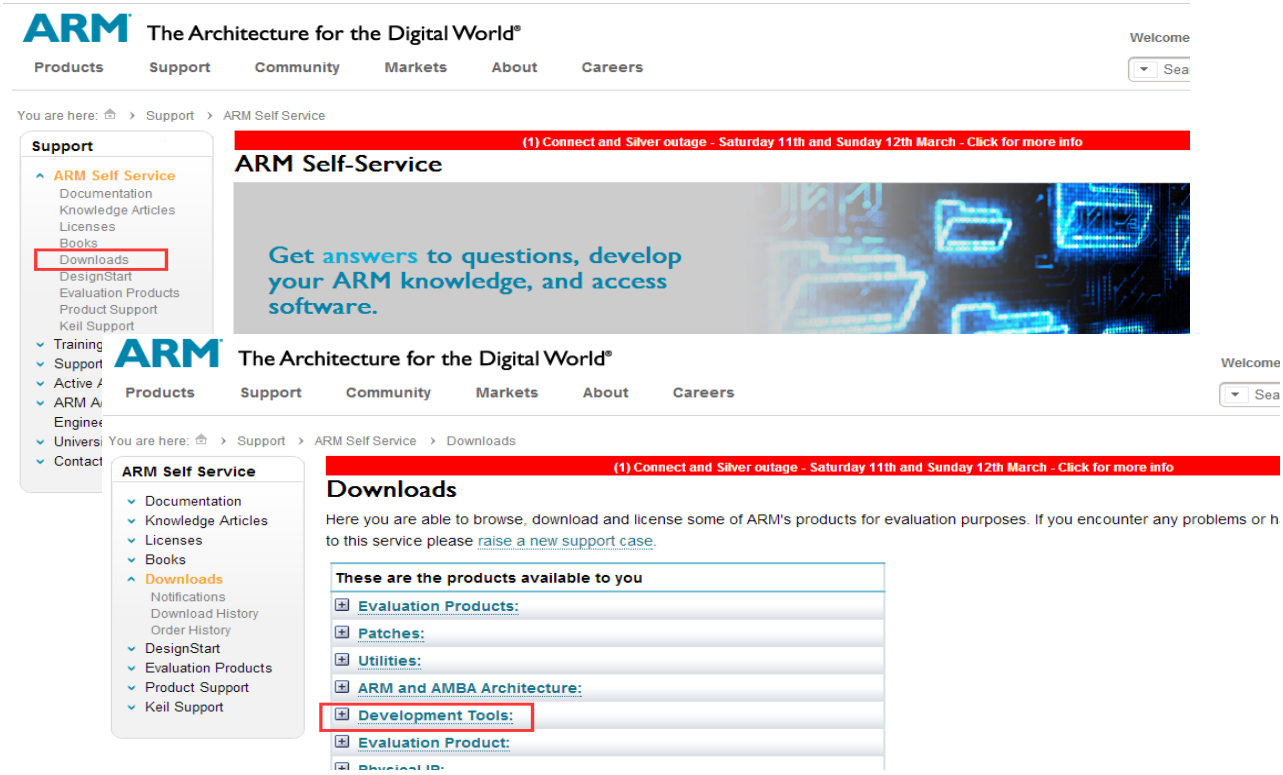

Figure 2: "Downloads" and "Development Tools" Pages

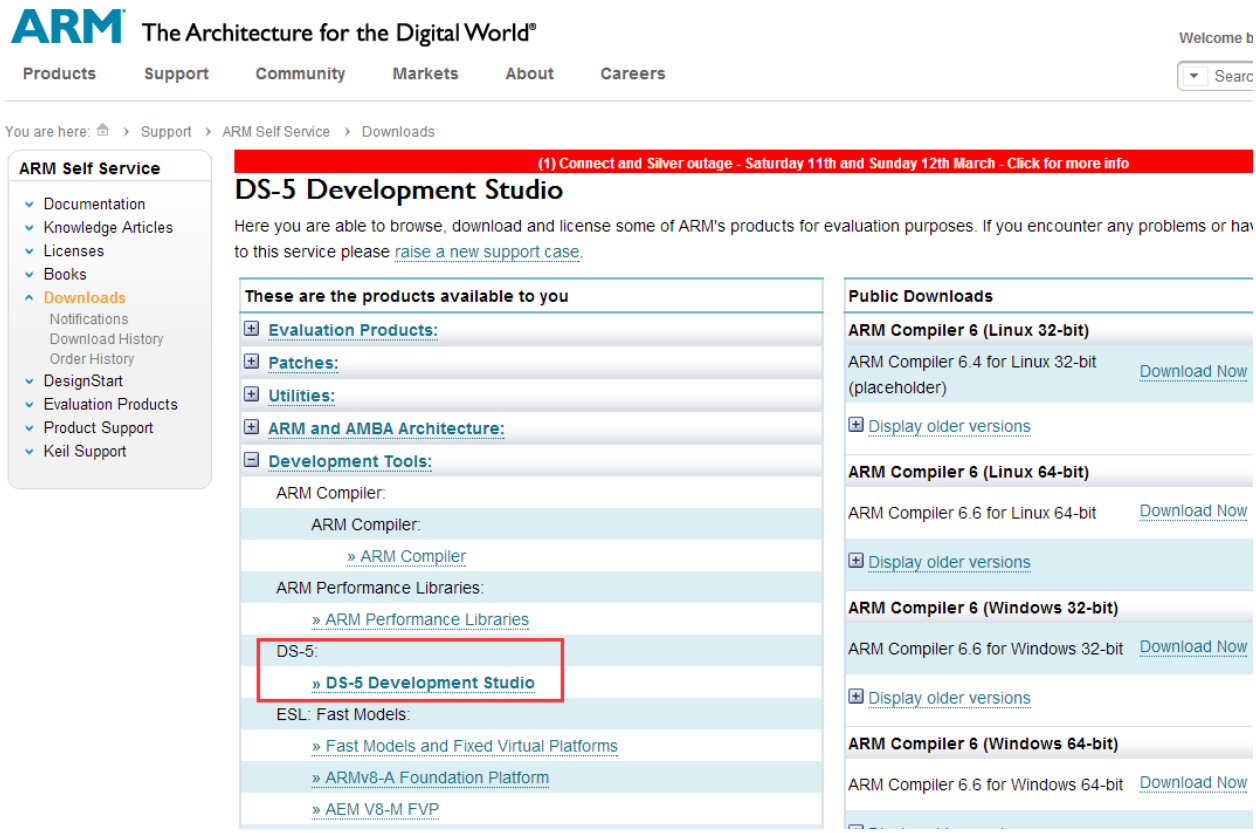

Figure 3: Click "DS-5 Development Studio"

(2) Under "**ARM Compiler 5 (Windows)**", click the "**Download Now**" button after "**ARM Compiler 5.05 update 1 (build 106) for Windows**" to download the corresponding ARM compiler tool for Windows.

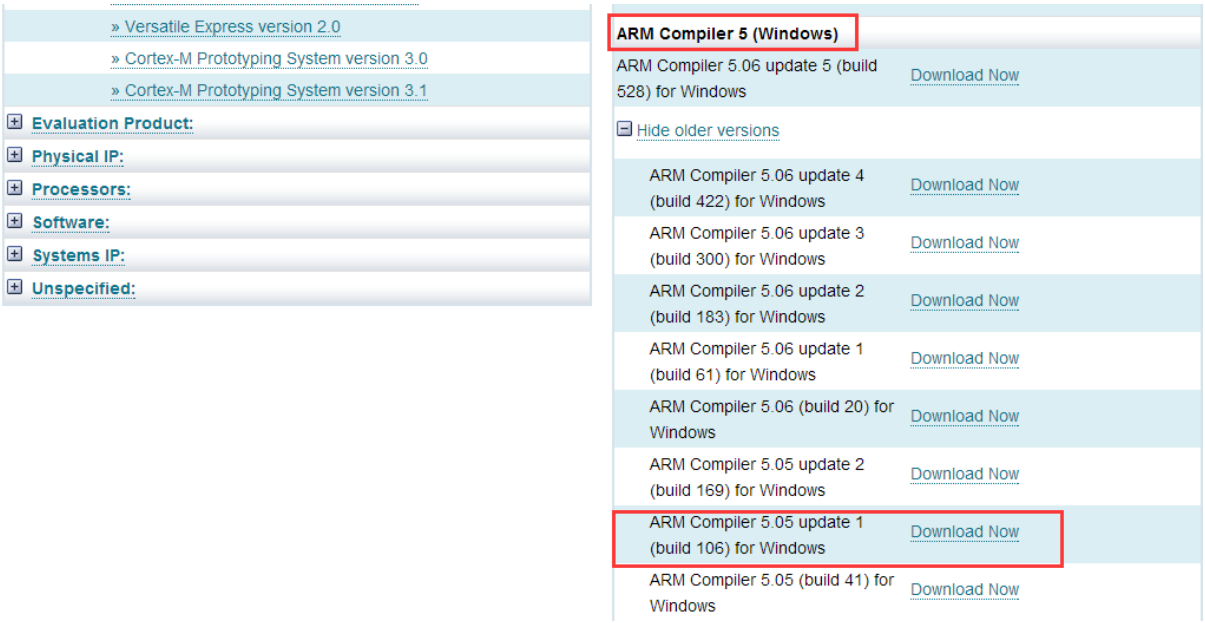

Figure 4: Download the Corresponding Tool

(3) After clicking "**Download Now**", there is a need to confirm the details shown as below:

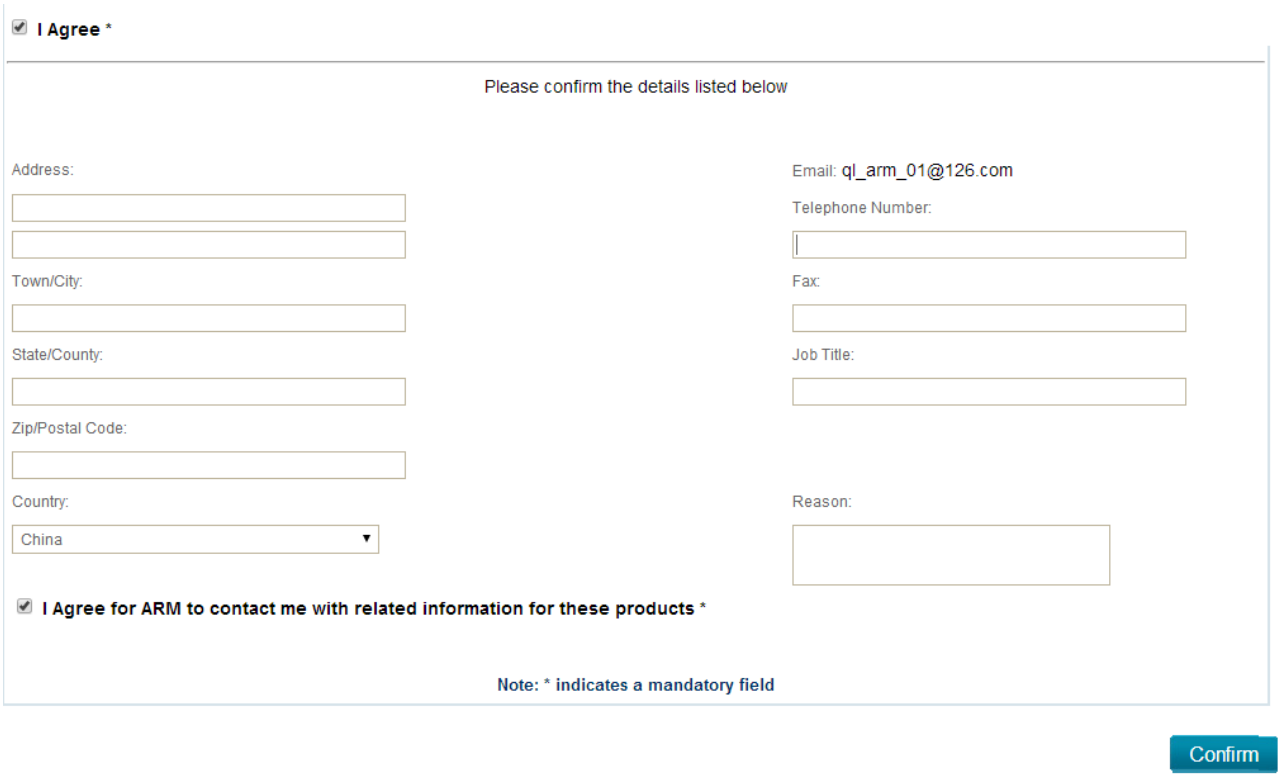

Figure 5: Confirmation of Details

(4) Finally click "**Confirm**" button and then the tool packet will be downloaded.

3.1.1.2. Install ARM Compiler Tool

After downloading ARM compiler tools, you can follow the steps illustrated below to finish installation of ARM compiler tool.

Step 1: Run "ARM Compiler 5 Setup" program and then click "**Next**".

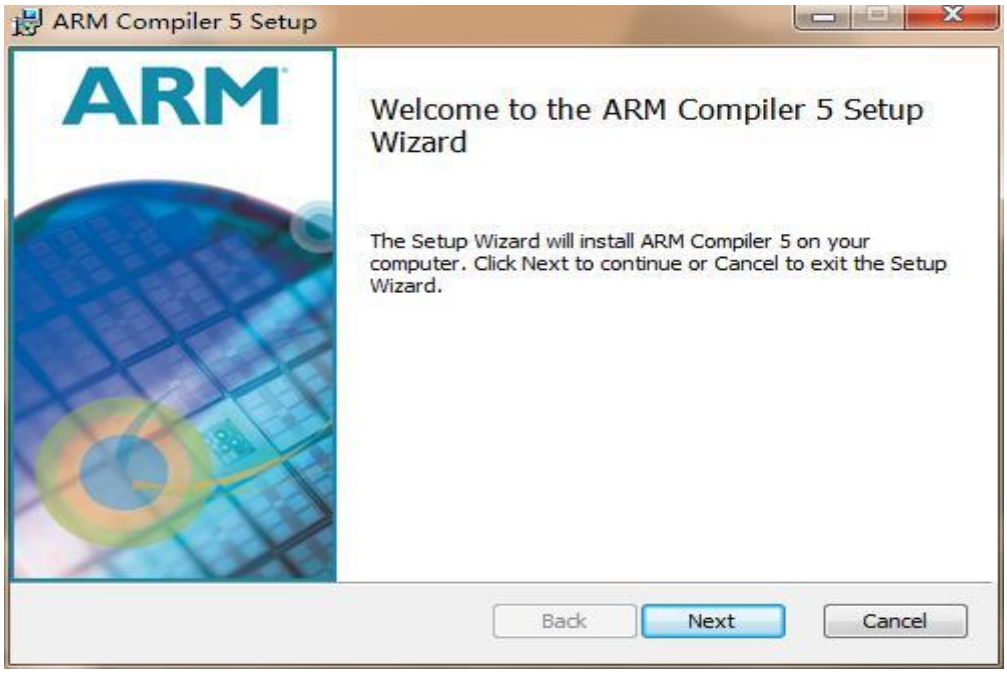

Figure 6: ARM Compiler 5 Setup

Step 2: Accept the terms in the license agreement and then click "**Next**".

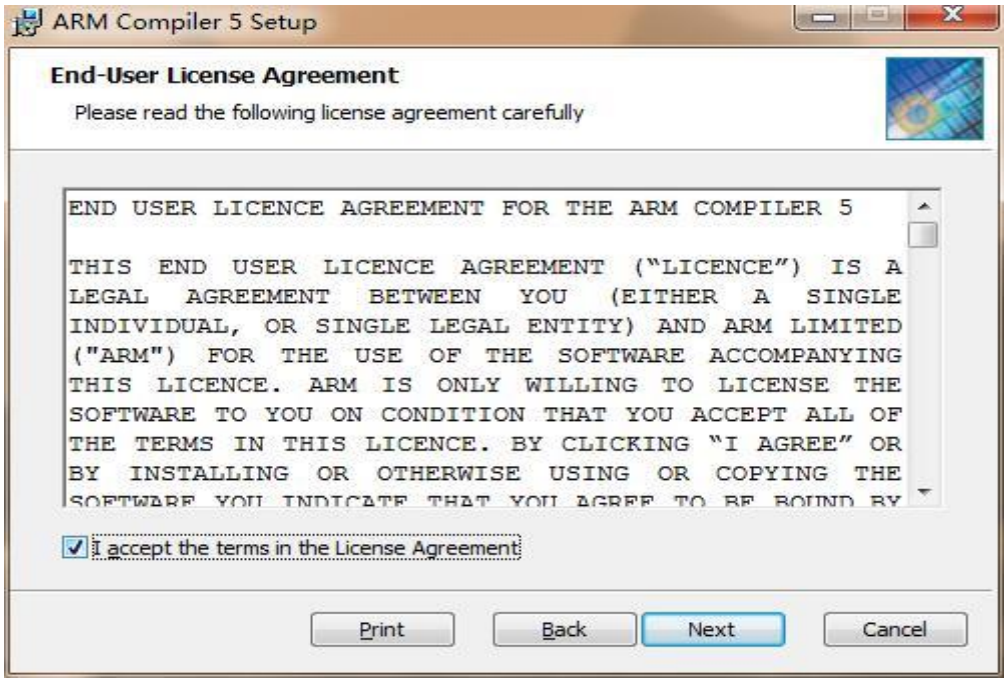

Figure 7: End-User License Agreement

Step 3: Select the way you want features to be installed.

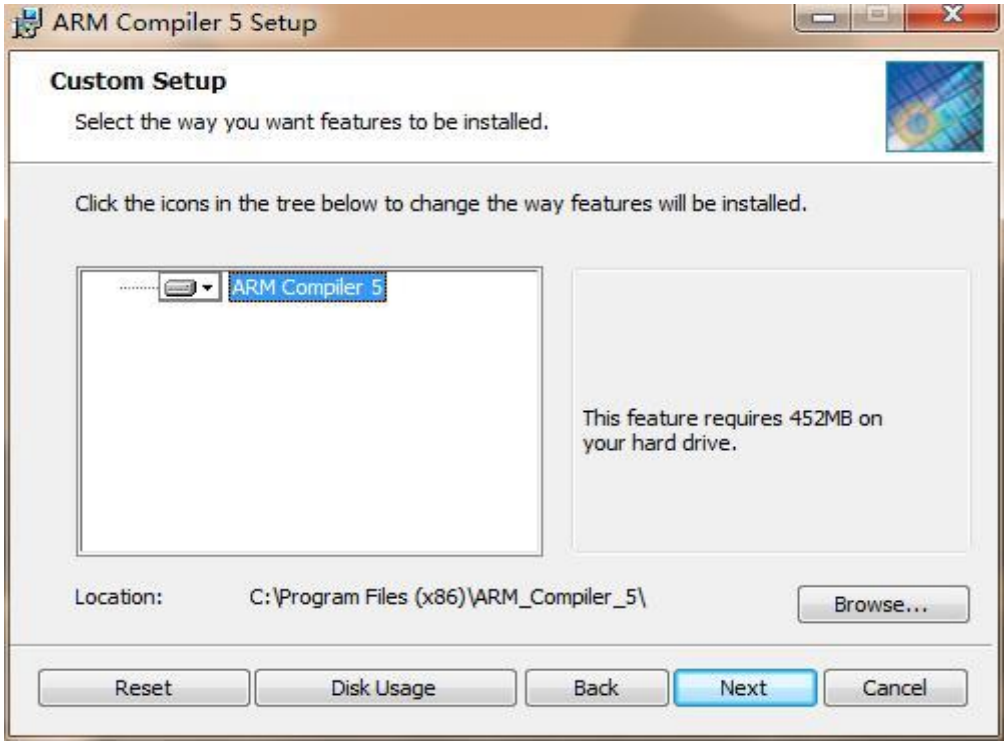

Figure 8: Custom Setup

Step 4: Ignore "**System Pending Reboot**" warning, and click "**Next**".

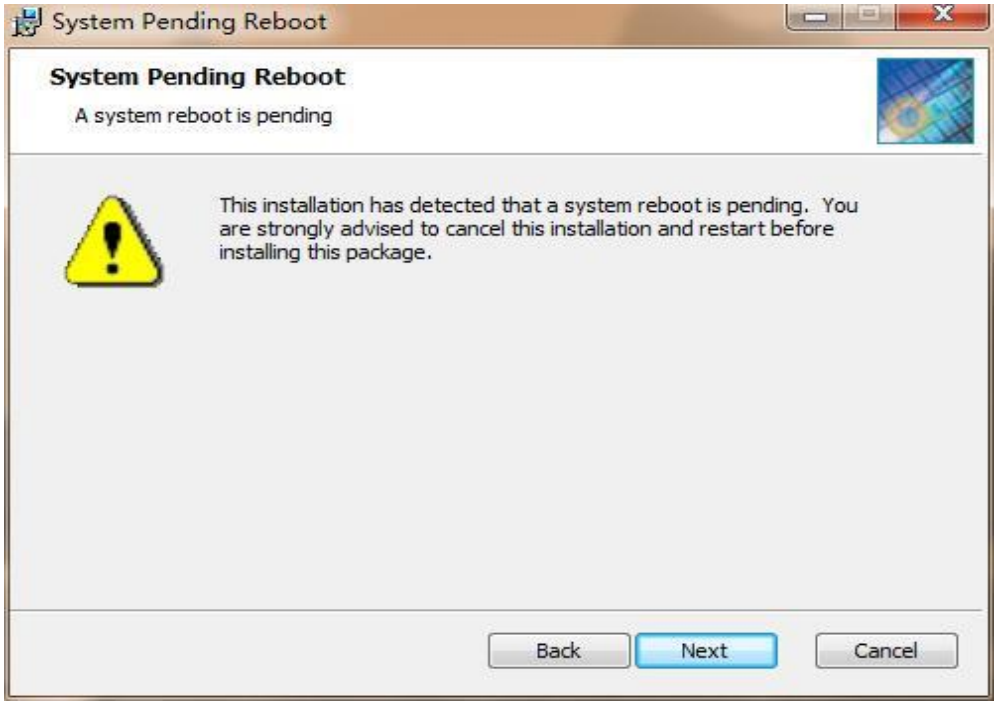

Figure 9: "System Pending Reboot" Warning

Step 5: Click "**Install**" to begin the installation, then wait while the setup wizard installs ARM compiler 5.

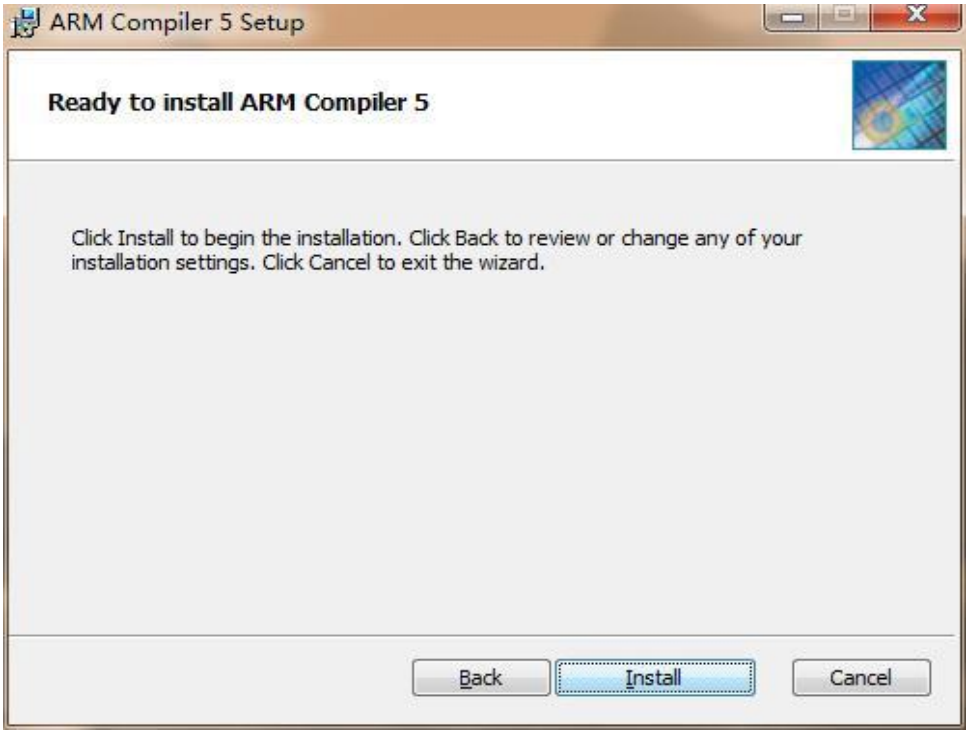

Figure 10: Ready to install ARM Compiler 5

Step 6: Click the "**Finish**" button to exit the setup wizard and complete the compiler tool installation.

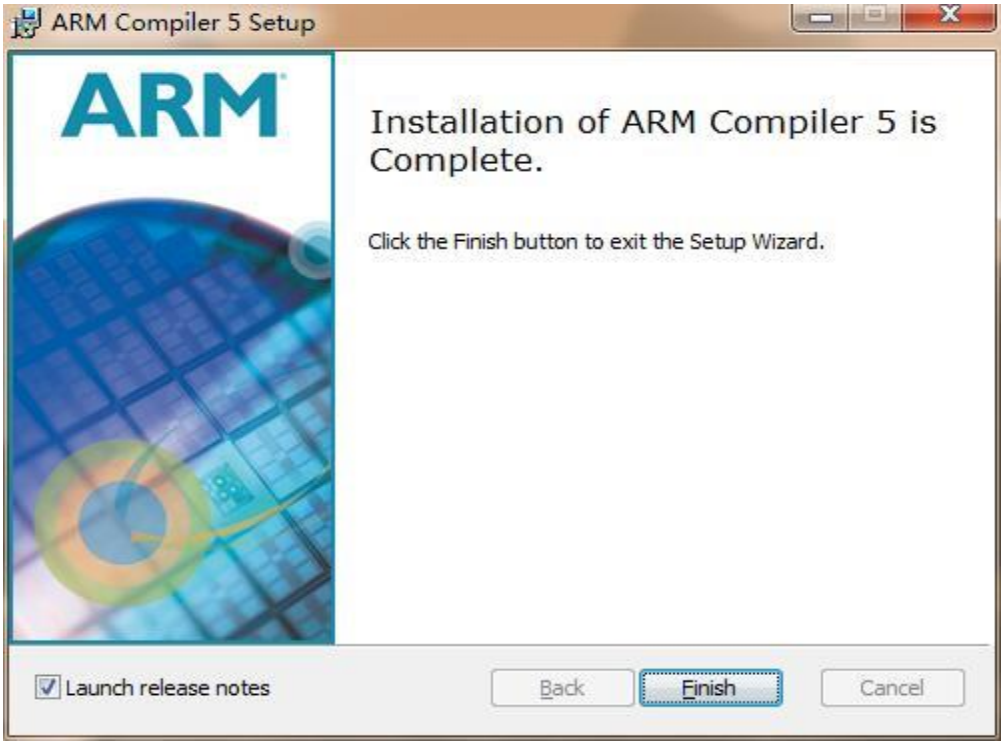

Figure 11: Finish Installation of ARM Compiler Tool

After successful installation of ARM compiler 5, there is a need to restart the computer to make the compilation tool take effect.

3.1.2. Download and Install Cygwin

3.1.2.1. Download Cygwin

Open the Cygwin download page shown as below to download the corresponding revision of Cygwin for Windows: https://cygwin.com/install.html.

3.1.2.2. Install Cygwin

To install the environment where you can compile the BG96 QuecOpen application, please follow the steps below:

Step 1: Run "**Cygwin Setup**" program and then click "**Next**".

Figure 12: Cygwin Setup Program

Step 2: Choose the installation type.

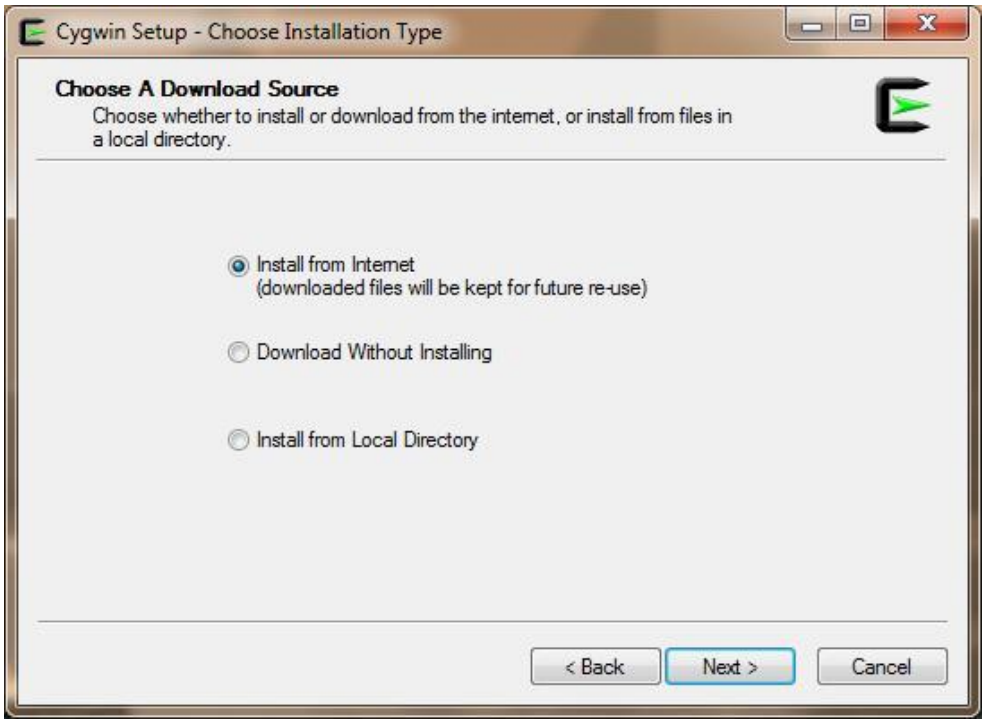

Figure 13: Choose Installation Type

Step 3: Select the directory where you want to install Cygwin, and also please choose a few installation parameters.

Figure 14: Choose Installation Directory and Parameters

Step 4: Select a directory where you want the setup to store the downloaded installation files.

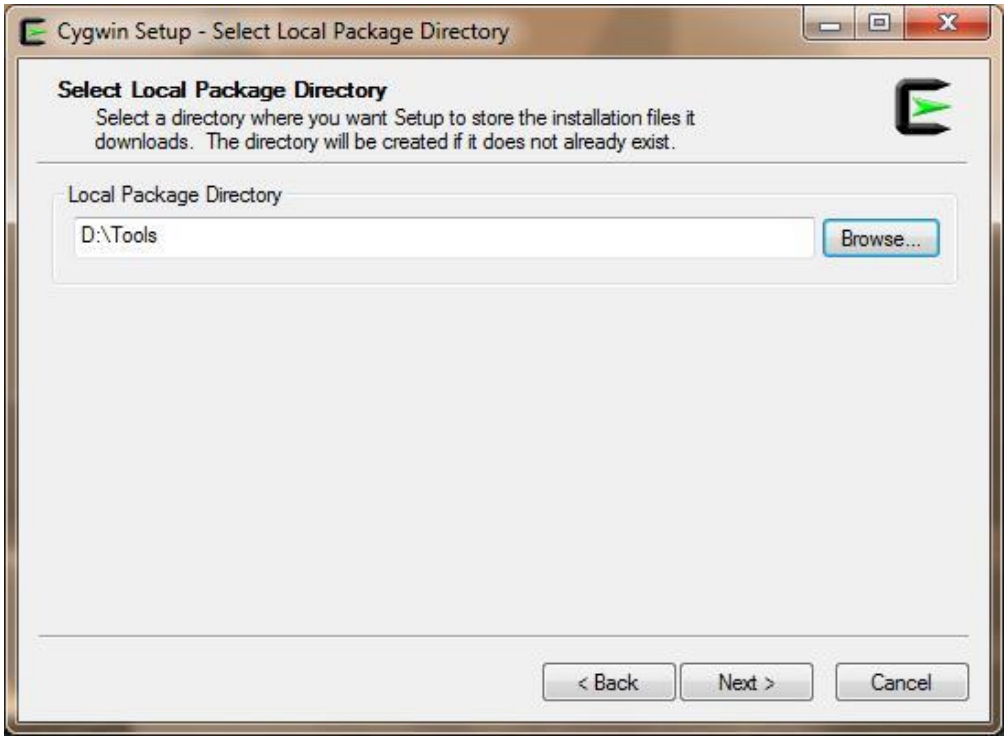

Figure 15: Select Local Package Directory

Step 5: Select the type of internet connection.

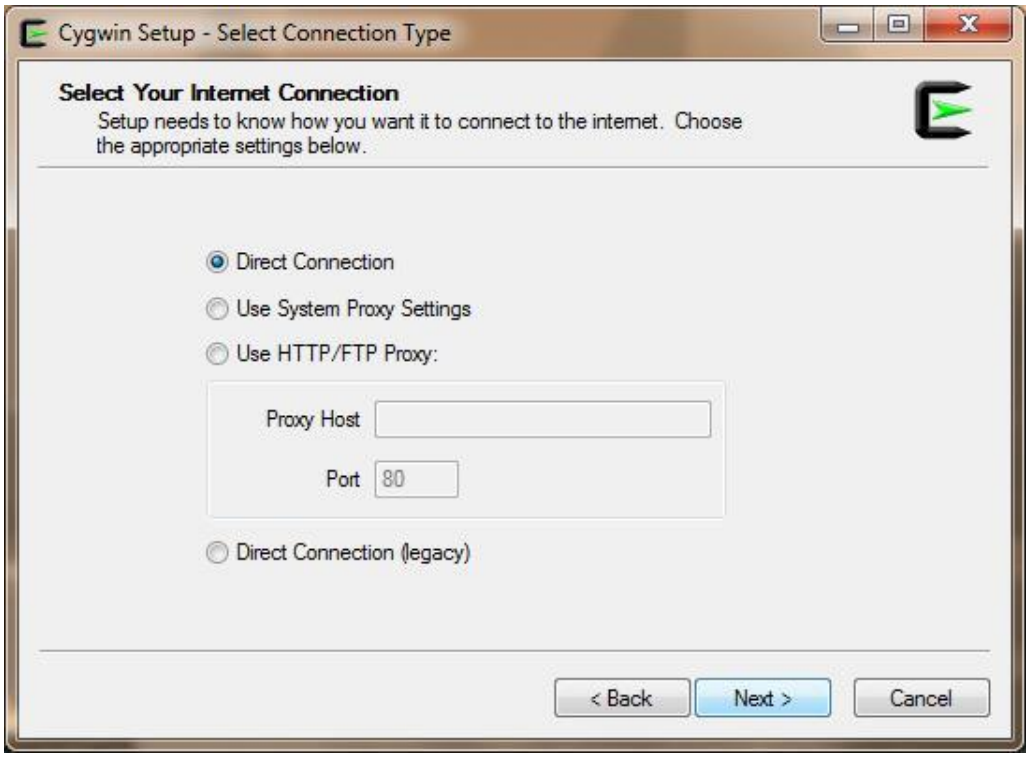

Figure 16: Select Internet Connection Type

Step 6: Then you will see the installation progress. After it completes, please choose a site from the list.

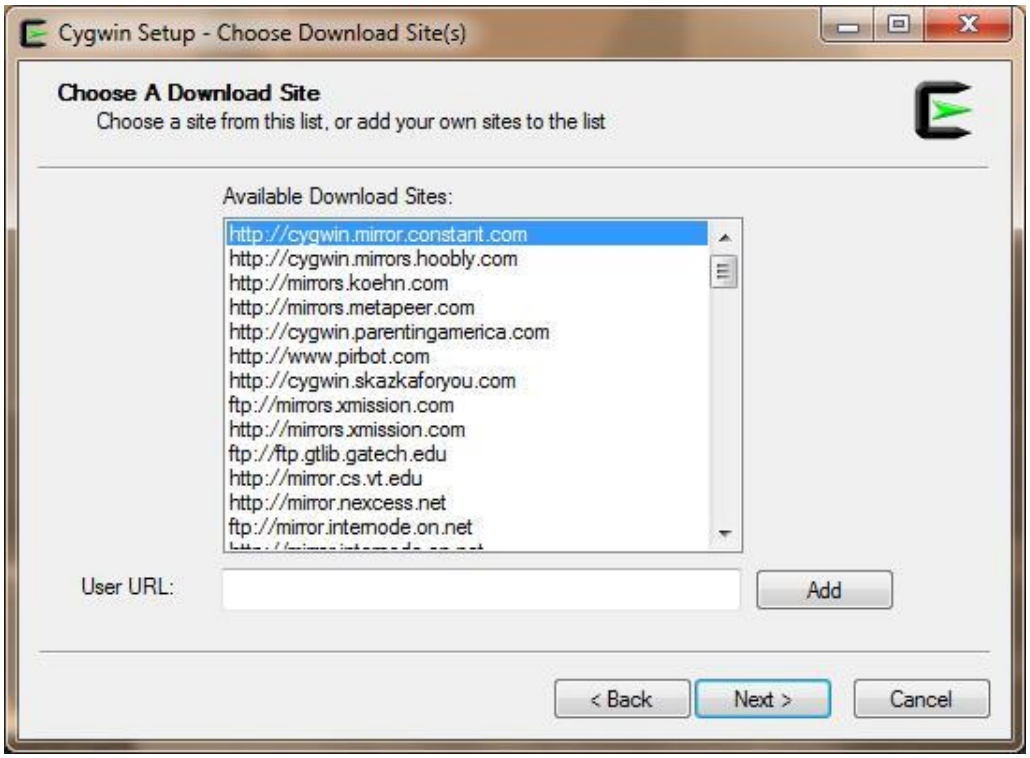

Figure 17: Choose a Download Site

Step 7: Please wait patiently during download or installation progress. When it completes, click "**Next**".

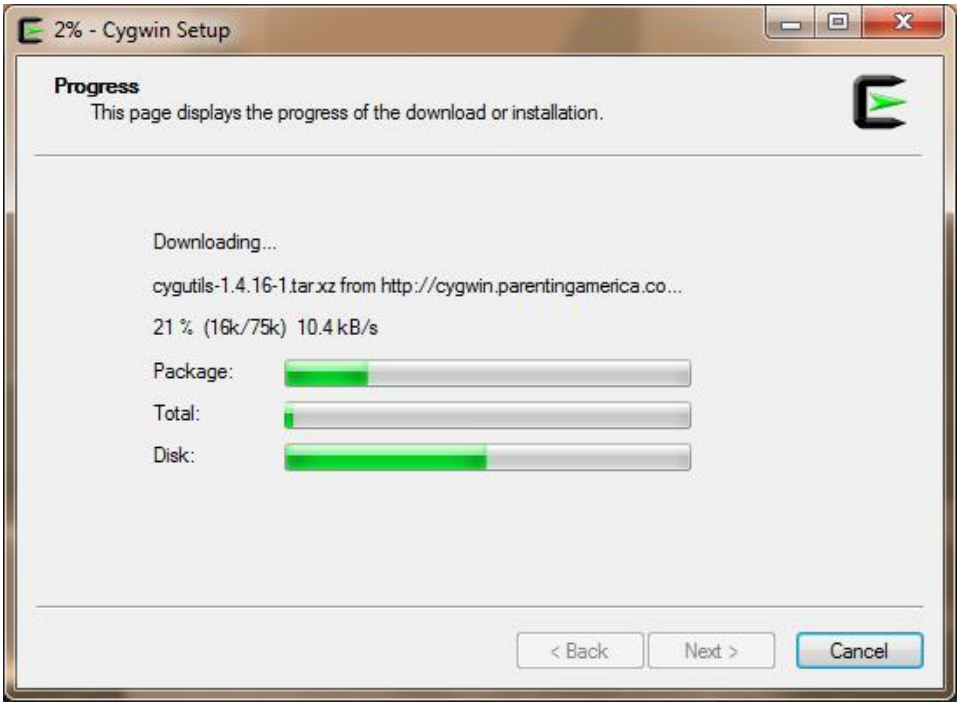

Figure 18: Process of Download or Installation

Step 8: Create an icon for Cygwin, and then click "**Finish**" to complete installation.

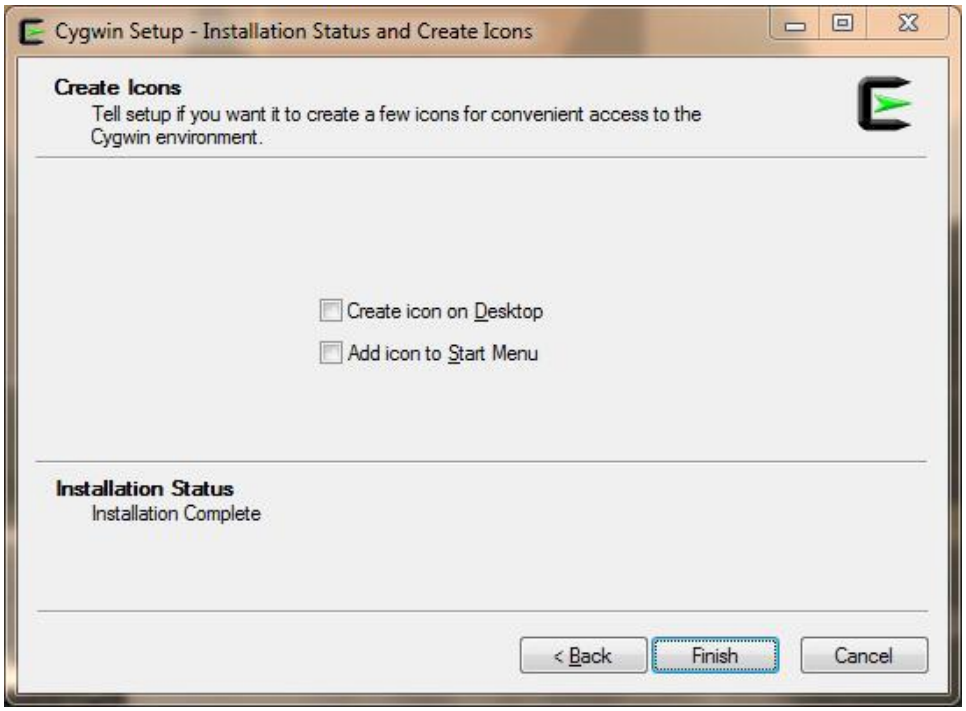

Figure 19: Create Icon and Complete Installation

After successful installation of both ARM compiler tool and Cygwin, customers can start compiling QuecOpen SDK. For details about compilation and running of QuecOpen SDK, please refer to ***Chapter 4*** and ***Chapter 5*** for details.

3.2. Setup Compiling Environment for TX3.0.1

3.2.1. Download LLVM

Licensed users can download LLVM compiler through the Qualcomm ChipCode™ portal. If Quectel customers have no license, please contact with Quectel.

3.2.1.1. For Windows 7 Build Hosts

Figure 20: LLVM Download for Windows 7 Build Hosts

3.2.1.2. For Linux Build Hosts

Figure 21: LLVM Download for Linux Build Hosts

After download, the tool can be used without installation procedure.

3.2.2. Download and Install Cygwin

Please refer to ***Chapter 3.1.2*** for details.

3.2.3. Download and Install Python

3.2.3.1. Download Python

Open the Python download page shown as below to download the corresponding revision of Python for Windows/Linux: https://www.python.org/download/releases/2.7/.

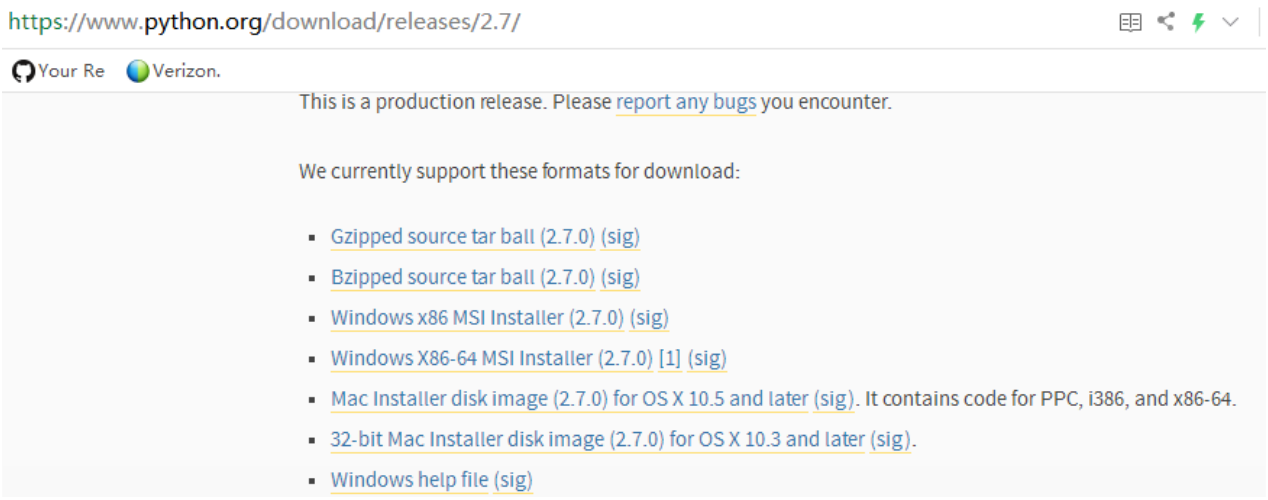

Figure 22: Python Download Page Screenshot

Download x86/x86-64 versions as needed.

3.2.3.2. Install Python

After download is completed, please follow the steps illustrated below to finish installation.

Step 1: Run "**Python-2.7.0.msi**" program and also please choose a few installation parameters, then click "**Next**".

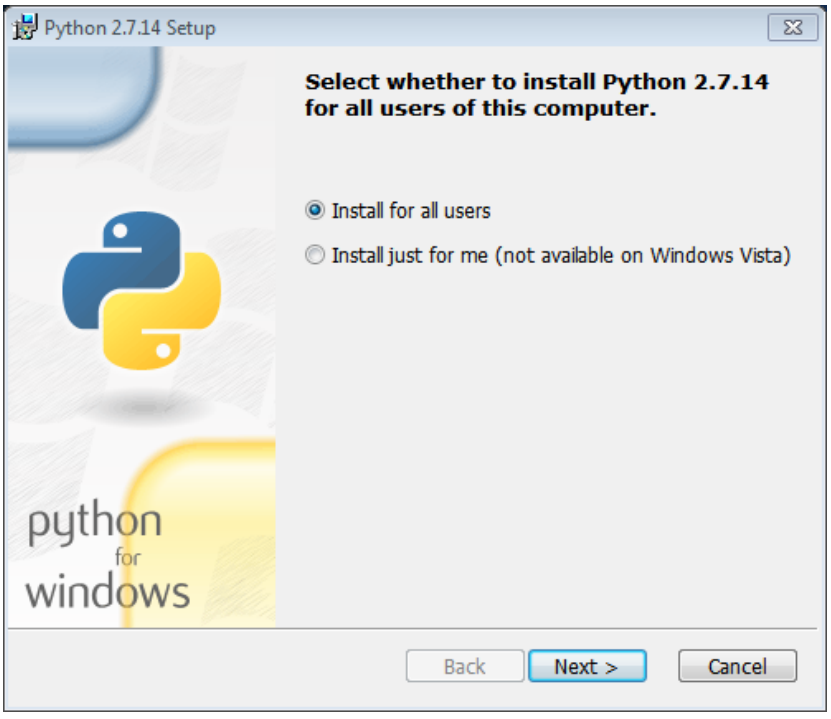

Figure 23: Python Setup

Step 2: Select the directory where Python is to be installed.

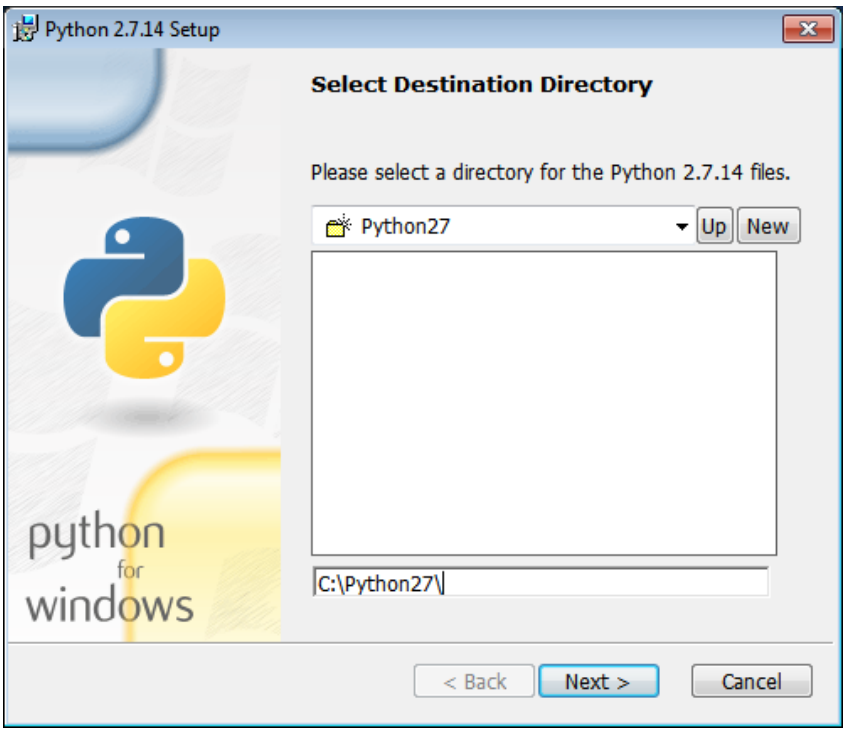

Figure 24: Select Installation Directory

Step 3: Options for customization. Please keep the default options.

Figure 25: Options for Customization

Step 4: Please wait during installation process.

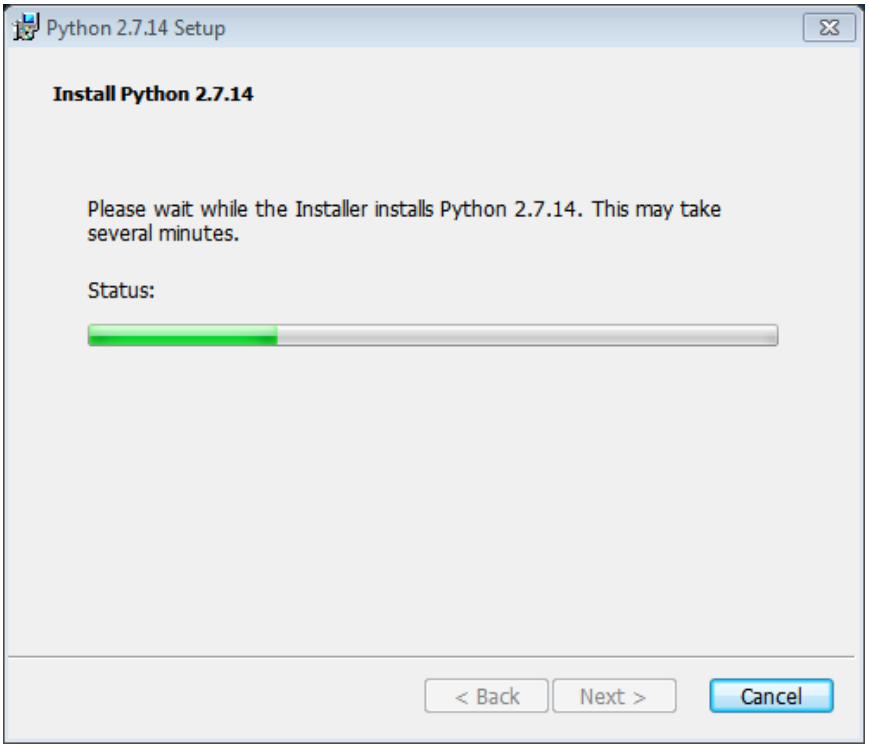

Figure 26: Installing

Step 5: Complete installation.

Figure 27: Installation Completed

4 Build QuecOpen Application

4.1. QuecOpen SDK Package based on TX2.0

4.1.1. SDK Package Structure

The following shows the folder structure of *Quectel_BG96_QuecOpen_SDK_Package* (based on TX2.0) which is created for non-licensed customers.

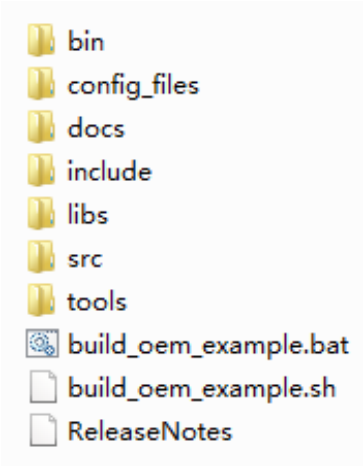

Figure 28: Folder Structure of BG96-QuecOpen SDK Package (TX2.0)

Table 3: Description of BG96-QuecOpen SDK Package Directories (TX2.0)

Directories	Description/Function
bin	Application gets created in this folder after successful compilation
config_files	Contains application related configuration file: *oem_app_path.ini*
docs	Guide documents
Include	Header files needed for compilation provided by Quectel
libs	Required libraries should be copied here

src	Application source code
tools	Tools for development
build_oem_example.bat	Batch script for Windows build hosts
build_oem_example.sh	Shell script for Windows build hosts
ReleaseNotes	Firmware release notes

4.1.2. Build QuecOpen User Application

Before application building, customers must set a correct path for the compiler tools in the build script.

In *build_oem_example.bat*:

```
set TOOL_PATH_ROOT=C:\compile_tools
set TOOLCHAIN_PATH=%TOOL_PATH_ROOT%\ARM_Compiler_5\bin
set LM_LICENSE_FILE=%TOOL_PATH_ROOT%\license.dat
```

In *build_oem_example.sh*:

```
TOOLCHAIN_PATH="C:/compile_tools/ARM_Compiler_5/bin"
export LM_LICENSE_FILE="C:/compile_tools/license.dat"
```

NOTE

The ARM compiler is not provided for free. If it is intended to be used, customers have to obtain the license first. Fortunately, a 30-day evaluation license is available for new users. Customer can request it on the ARM official website.

To build the example codes in the *Quectel_BG96_QuecOpen_SDK_Package*, customers just need to run the following command from command line in Cygwin or DOS window:

The following help commands can be used to know which examples are supported in this SDK package:

help build in Linux:

```
./build_oem_example.sh help
```

Or

help build in Windows:

```
build_oem_example.bat help
```

After input the help command, tips shown as below will be available:

```
Supported example :
   device_info [ cmd - build_oem_example.bat device_info ]
   gpio        [ cmd - build_oem_example.bat gpio        ]
   gpio_int    [ cmd - build_oem_example.bat gpio_int    ]
   gps         [ cmd - build_oem_example.bat gps         ]
   qt_gps      [ cmd - build_oem_example.bat qt_gps      ]
   http        [ cmd - build_oem_example.bat http        ]
   psm         [ cmd - build_oem_example.bat psm         ]
   rtc         [ cmd - build_oem_example.bat rtc         ]
   task_create [ cmd - build_oem_example.bat task_create ]
   tcp_client  [ cmd - build_oem_example.bat tcp_client  ]
   time        [ cmd - build_oem_example.bat time        ]
   timer       [ cmd - build_oem_example.bat timer       ]
   uart        [ cmd - build_oem_example.bat uart        ]
   atc_pipe    [ cmd - build_oem_example.bat atc_pipe    ]
   atc_sms     [ cmd - build_oem_example.bat atc_sms     ]
   i2c         [ cmd - build_oem_example.bat i2c         ]
   mqtt        [ cmd - build_oem_example.bat mqtt        ]
   spi         [ cmd - build_oem_example.bat spi         ]
   dns_client  [ cmd - build_oem_example.bat dns_client  ]
   adc         [ cmd - build_oem_example.bat adc         ]
   qt_adc      [ cmd - build_oem_example.bat qt_adc      ]
   nonip       [ cmd - build_oem_example.bat nonip       ]
   fota        [ cmd - build_oem_example.bat fota        ]
   lwm2m       [ cmd - build_oem_example.bat lwm2m       ]
```

Take UART compilation as an example:

In Cygwin:

New build:
./build_oem_example.sh uart

Clean build:
./ build_oem_example.sh -c

In DOS window:

New build:
build_oem_example.bat uart

Clean build:
build_oem_example.bat -c

Once the build process is completed, the application binary (e.g., *example_uart.bin*) will be created under the path */bin*.

4.2. QuecOpen SDK Package based on TX3.0.1

4.2.1. SDK Package Structure

The following shows the folder structure of *Quectel_BG96_QuecOpen_SDK_Package* (based on TX3.0.1) which is created for non-licensed customers.

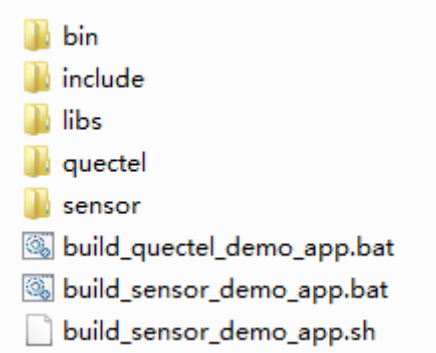

Figure 29: Folder Structure of BG96-QuecOpen SDK Package (TX3.0.1)

Table 4: Description of BG96-QuecOpen SDK Package Directories (TX3.0.1)

Directories	Description/Function
bin	Application gets created in this folder after successful compilation
Include	Header files needed for compilation provided by Quectel
libs	Required libraries should be copied here
quectel	Quectel example source codes
sensor	Qualcomm example source codes
build_quectel_demo_app.bat	Batch script for building Quectel example
Build_sensor_demo_app.bat	Batch script for building Qualcomm example
Build_sensor_demo_app.sh	Shell script for building Qualcomm example

4.2.2. Build QuecOpen User Application

Before application building, customers must set a correct path for the compiler tools in the build script.

In *build_quectel_demo_app.bat*:

```
set TOOL_PATH_ROOT=C:\compile_tools
set TOOLCHAIN_PATH=%TOOL_PATH_ROOT%\LLVM\4.0.3\bin
set
TOOLCHAIN_PATH_STANDARdS=%TOOL_PATH_ROOT%\LLVM\4.0.3\armv7m-none-eabi\libc\include
set LLVMLIB=%TOOL_PATH_ROOT%\LLVM\4.0.3\lib\clang\4.0.3\lib
set LLVMLINK_PATH=%TOOL_PATH_ROOT%\LLVM\4.0.3\tools\bin
set PYTHON_PATH=%TOOL_PATH_ROOT%\Python27\python.exe
```

In *build_quectel_demo_app.sh*:

```
set TOOL_PATH_ROOT=C:\compile_tools
set TOOLCHAIN_PATH=%TOOL_PATH_ROOT%\LLVM\4.0.3\bin
set
TOOLCHAIN_PATH_STANDARdS=%TOOL_PATH_ROOT%\LLVM\4.0.3\armv7m-none-eabi\libc\include
set LLVMLIB=%TOOL_PATH_ROOT%\LLVM\4.0.3\lib\clang\4.0.3\lib
set LLVMLINK_PATH=%TOOL_PATH_ROOT%\LLVM\4.0.3\tools\bin
set PYTHON_PATH=%TOOL_PATH_ROOT%\Python27\python.exe
```

To build the example codes in the *Quectel_BG96_QuecOpen_SDK_Package*, customers just need to run the following command from command line in Linux or Windows OS:

The following help commands can be used to know which examples are supported in this SDK package:

help build in Linux:

```
./build_quectel_demo_app.sh help
```

Or

help build in Windows:

```
build_quectel_demo_app.bat help
```

After input the help command, tips shown as below will be available:

```
Supported example :
   adc          [ cmd - build_quectel_demo_app.bat adc          ]
   device_info  [ cmd - build_quectel_demo_app.bat device_info  ]
   dns_client   [ cmd - build_quectel_demo_app.bat dns_client   ]
   gpio         [ cmd - build_quectel_demo_app.bat gpio         ]
   gpio_int     [ cmd - build_quectel_demo_app.bat gpio_int     ]
   gps          [ cmd - build_quectel_demo_app.bat gps          ]
   http         [ cmd - build_quectel_demo_app.bat http         ]
   i2c          [ cmd - build_quectel_demo_app.bat i2c          ]
   mqtt         [ cmd - build_quectel_demo_app.bat mqtt         ]
   psm          [ cmd - build_quectel_demo_app.bat psm          ]
   rtc          [ cmd - build_quectel_demo_app.bat rtc          ]
   spi          [ cmd - build_quectel_demo_app.bat spi          ]
   task_create  [ cmd - build_quectel_demo_app.bat task_create  ]
   tcp_client   [ cmd - build_quectel_demo_app.bat tcp_client   ]
   time         [ cmd - build_quectel_demo_app.bat time         ]
   timer        [ cmd - build_quectel_demo_app.bat timer        ]
   uart         [ cmd - build_quectel_demo_app.bat uart         ]
```

Take UART compilation as an example:

In Linux:

New build:
./build_quectel_demo_app.sh uart

Clean build:
./build_quectel_demo_app.sh -c

In Windows:

New build:
build_quectel_demo_app.bat uart

Clean build:
build_quectel_demo_app.bat -c

Once the build process is completed, the application binary (e.g., *quectel_demo_uart.bin*) will be created under the path */bin*.

5 Enable QFLOG based on TX3.0.1

QFLOG function designed for clashing and logging for the QuecOpen user application. QFLOG on the BG96 is disabled by default. Customer can use below AT commands to enable this function.

- **AT+QCFGEXT="qflogport",1**
- **AT+QCFGEXT="qflogen",1**

NOTEs

1. Please reset module after set these 2 AT commands.
2. Due to the USB composition is different with Qualcomm original configuration, these was no "**SER5**" port in the Device Management. We change the "**USB AT PORT**" function as "**SER5**" PORT function. So please use the "**USB AT PORT**" in you python script.

5.1. Setting up the CLI

- Install Python 3.6.1 [https://www.python.org/ftp/python/3.6.1/python-3.6.1-amd64.exe]
- Install pyserial 3.4 [pip install pyserial]
- Add the path to the project directory (up till ~\src) to the PYTHONPATH environment variable. Please find the steps to configure the said environment variable as follows:
  - Go to Computer -> Properties -> Advanced System Settings
  - Click Environment Variables
  - Create the PYTHONPATH system environment variable if it doesn't already exist
  - Add the QFLOG path to the project directory (up till ~\src) to the PYTHONPATH environment variable.

5.2. Executing the CLI

- First send a hello command to check the connectivity.
  python QFLOG.py -p <COMPORT> HELLO
  If communication successfully, could get response as below:

```
F:\H4GD\Qualcomm\MDM9206\MCU_R04\apps_proc\QFLOG\src\QFLOGPackage>python QFLOG.py -p COM14 HELLO
2018-05-17 13:45:08,586 QFLOG    8000    3828    INFO     : Sending HELLO to device
2018-05-17 13:45:08,589 QFLOG    8000    8028    INFO     : Received ACK
```

Figure 30: QFLOG received ACK

- Get help with Help command
  python QFLOG.py -p COM14 –h

```
F:\H4GD\Qualcomm\MDM9206\MCU_R04\apps_proc\QFLOG\src\QFLOGPackage>python QFLOG.py -p COM14 -h
usage: QFLOG.py [-h] [-v] [-p PORT] {HELLO,PUSH,DELETE,VIEW_LOGS} ...

positional arguments:
  {HELLO,PUSH,DELETE,VIEW_LOGS}
                        Command
    HELLO               Hello
    PUSH                Push
    DELETE              Delete
    VIEW_LOGS           View Logs

optional arguments:
  -h, --help            Help
  -v, --version         Version
  -p PORT, --port PORT  Port
```

Figure 31: QFLOG Help Command

5.3. Flash and Upload File

- Push the binary image to the folder location where QFLOG Scripts are placed and run the scripts with PUSH command mentioned in the scripts help file.

  python QFLOG.py -p <COMPORT> PUSH -f <absolute bin path>

```
F:\H4GD\Qualcomm\MDM9206\MCU_R04\apps_proc\QFLOG\src\QFLOGPackage>python QFLOG.py -p COM14 PUSH -f ./env.bat
2018-05-17 14:07:47,235 QFLOG     700    6604   INFO     : Resetting the context by sending a HELLO packet
2018-05-17 14:07:47,236 QFLOG     700    7744   INFO     : Sending HELLO to device
2018-05-17 14:07:47,239 QFLOG     700   10316   INFO     : Received ACK
2018-05-17 14:07:47,491 QFLOG     700   10960   INFO     : Pushing env.bat to device
2018-05-17 14:07:47,501 QFLOG     700    2288   INFO     : Received ACK
2018-05-17 14:07:47,745 QFLOG     700   10960   INFO     : Push complete

F:\H4GD\Qualcomm\MDM9206\MCU_R04\apps_proc\QFLOG\src\QFLOGPackage>
```

Figure 32: QFLOG PUSH Command

- File will be available in /datatx/ directory. Module manager which is written should be written in such a way that the binary needs to be picked up from */datatx/<BIN_PUSHED>*.

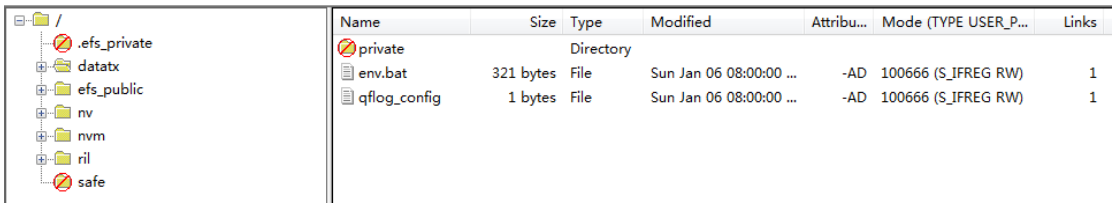

Figure 33: QFLOG push file successfully.

5.4. Logging

Customer can enable logging function in their QuecOpen application with below operation.

- Include *qflog_utils.h* file.

  The below statement is to be used for logging a message.

  QFLOG_MSG*(MSG_SSID_DFLT,MSG_MASK_2, "Presence sensor application successfully registered");*

  QFLOG_MSG*(MSG_SSID_DFLT,MSG_MASK_2, "Sensor time value is %d" , sensor_time_val);*

- Before receiving the logs please ensure to run the python script with the following command to receive logs. In the console.

  QFLOG.py -p <COMPORT> VIEW_LOGS

Customer can send *Ctrl + 'C'* to terminate the session.

6 Run QuecOpen Application

To run the QuecOpen application binary file, customers only need to upload the application binary image and *oem_app_path.ini* into the alternate file systems of BG96 by QEFS Explorer.

The *oem_app_path.ini* file includes the full path of the location of application binary image. This file must be stored in the */datatx/* directory.

After uploading these two files into alternate file systems, restart BG96 and the application binary image will be loaded into RAM and started by the Module Loader.

NOTE

For detailed usage of QEFS Explorer, please refer to *Quectel_BG96_QEFS_Explorer_User_Guide*.

7 Update QuecOpen Application

Customers can download their new application image form their own Revision Control Server. For example, they can use HTTP(s) to access the HTTP(s) server and download the upgrade image to the file system and store it in the specified path.

In the file system, there are two places which are used to store the application images. For instance, the path to existing UART application could be:

- */datatx/quectel_uart_demo.bin* (path #1)

An upgraded image can be downloaded into:

- */datatx/upgrade/quectel_uart_demo_upgrade.bin* (path #2)

Also, customers need modify the *oem_app_path.ini* file contents according to the format shown as below:

/datatx/quectel_uart_demo.bin:/datatx/ upgrade/quectel_uart_demo_upgrade.bin

- "/datatx/quectel_uart_demo.bin" indicates the full path of the boot-up image
- "/datatx/quectel_uart_demo_upgrade.bin" indicates the full path of the upgraded image

After download is completed, customers only need to restart the module.

While booting, the Module Loader can check:

- Whether the path #2 is present and will load it if it presents.
- If loading fails or the file is not present, it reverts to path #1 and will delete path #2.
- If it loads the new app successfully, it will copy path #2 to path #1, and delete path #2 after copy completed.

NOTES

1. ":" is the delimiter, and is used to distinguish the path of original version from that of upgraded version.
2. The firmware name and storage path of the old firmware and upgraded firmware are customer specific. Customers could modify them according to their individual demands. But the delimiter ":" must not be modified or replaced with other symbols.

8 QAPI Functions

The “qapi_quectel.h” header file declares the all QAPIs which designed by Quectel. These functions are essential to any customer’s applications. Please be sure to include the header file.

8.1. System API

8.1.1. API Functions

8.1.1.1. qapi_QT_Reset_Device

This function used to reset module.

- **Prototype**

```
qapi_Status_t qapi_QT_Reset_Device(uint16_t mode)
```

- **Parameters**

mode:
[in] Must be 0.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.1.1.2. qapi_QT_Sahara_Mode_Get

When module meets fatal error, module would enter into 2 modes, one is normal reset mode, and the other is Sahara dump mode which to collect dump files to help analyze the crash issue. This function used to get the NV item value of Sahara mode setting.

- **Prototype**

```
qapi_Status_t qapi_QT_Sahara_Mode_Get(QAPI_FATAL_ERR_MODE *mode)
```

```
typedef enum
{
    QAPI_FATAL_ERR_RESET = 0,
```

```
    QAPI_FATAL_ERR_SAHARA = 1,
    QAPI_FATAL_ERR_MAX
}QAPI_FATAL_ERR_MODE;
```

- **Parameters**

mode:
[in] Pointer, store the Sahara setting value which read from the NV item.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.1.1.3. qapi_QT_Sahara_Mode_Set

This function used to set the NV item value of Sahara mode setting. If customer want module automatically reset when meet crash, set the mode as QAPI_FATAL_ERR_RESET. If customer want module collects the DUMP file, set the mode as QAPI_FATAL_ERR_SAHARA. The Settings will take effect after the module is restarted.

- **Prototype**

```
qapi_Status_t qapi_QT_Sahara_Mode_Set(QAPI_FATAL_ERR_MODE mode)
```

```
typedef enum
{
    QAPI_FATAL_ERR_RESET = 0,
    QAPI_FATAL_ERR_SAHARA = 1,
    QAPI_FATAL_ERR_MAX
}QAPI_FATAL_ERR_MODE;
```

- **Parameters**

mode:
[in] Set the module behavior when module meet fatal error.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.1.1.4. qapi_QT_Get_Rel_Info

This function used to get apps kernel release information.

- **Prototype**

```
qapi_Status_t qapi_QT_Get_Rel_Info(qapi_QT_Rel_Info *rel_info);
```

```
typedef struct{
    char build_time[32];
    char build_date[32];
    char rel_time[32];
    char rel_date[32];
    char sw_version[128];
    char ql_sw_ver[64];
    char ql_sw_ver_sub[16];
}qapi_QT_Rel_Info;
```

- **Parameters**

rel_info:
[out] Pointer, store the information of app release

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.1.1.5. qapi_QT_Shutdown_Device

This function used to shutdown module.

- **Prototype**

```
qapi_Status_t qapi_QT_Shutdown_Device(void);
```

- **Parameters**

None

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.1.1.6. qapi_QT_RAI_Set

This function used to set the Release Assistant Information indicator.

- **Prototype**

```
qapi_Status_t qapi_QT_RAI_Set(qapi_DSS_Hndl_t hndl, uint8_t rai_type);
```

```
typedef void * qapi_DSS_Hndl_t;
uint8_t rai_type
```

- **Parameters**

hndl:
[in]Pointer to data service handle

rai_type:
[in]Type of Release Assistant Information indicator

0	No information available
1	No further uplink or downlink data transmission
2	Only a single downlink data transmission and no further uplink data

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.1.2. Example

1. Reset module example:

```
qapi_Status_t status = QAPI_QT_ERR_OK;

status = qapi_QT_Reset_Device(0);
if(status == QAPI_QT_ERR_OK)
{
    //Module will reset and auto startup.
}
else
{
    //QAPI invoked failed
    IOT_INFO("Reset module failed, status %x", status);
}
```

2. Sahara setting example:

```
qapi_Status_t status = QAPI_QT_ERR_OK;
QAPI_FATAL_ERR_MODE mode = QAPI_FATAL_ERR_MAX;

status = qapi_Status_t qapi_QT_Sahara_Mode_Set(0)
if(status == QAPI_QT_ERR_OK)
```

```
{
    //Module will automatically reset when meet crash
}
else
{
    //QAPI invoked failed
    IOT_INFO("Reset module failed, status %x", status);
}
```

3. Sahara getting example:

```
qapi_Status_t status = QAPI_QT_ERR_OK;
QAPI_FATAL_ERR_MODE mode = QAPI_FATAL_ERR_MAX;

status = qapi_Status_t qapi_QT_Sahara_Mode_Get(0)
if(status == QAPI_QT_ERR_OK)
{
    // Get sahara setting value
}
else
{
    //QAPI invoked failed
    IOT_INFO("Get sahara setting value failed, status %x", status);
}
```

4. Get release information

```
qapi_Status_t status = QAPI_QT_ERR_OK;
status =qapi_QT_Get_Rel_Info((qapi_QT_Rel_Info*)param_1);
if(status == QAPI_QT_ERR_OK)
{
    //Get apps release information
}
else
{
    //QAPI invoked failed
    IOT_INFO("Get apps release information failed, status %x", status);
}
```

5. Shutdown device

```
qapi_Status_t status = QAPI_QT_ERR_OK;
status = qapi_Status_t qapi_QT_Shutdown_Device(void);
```

```
if(status == QAPI_QT_ERR_OK)
{
    //Shutdown device
}
else
{
    //QAPI invoked failed
    IOT_INFO("Shutdown device failed, status %x", status);
}
```

6. RAI set

```
qapi_Status_t status = QAPI_QT_ERR_OK;
status = qapi_QT_RAI_Set((qapi_DSS_Hndl_t)param_1, (uint8_t) param_2);
if(status == QAPI_QT_ERR_OK)
{
    // set the Release Assistant Information indicator
}
else
{
    //QAPI invoked failed
    IOT_INFO("set the Release Assistant Information indicator failed, status %x", status);
}
```

8.2. ATC Pipe API

BG96 QuecOpen supports customer execute AT Comamnd in the user application. Maximum support to 2 channels.

8.2.1. API Functions

8.2.1.1. qapi_QT_Apps_AT_Port_Open

This function used to open an internal AT command pipe to execute AT command in the QuecOpen application.

- **Prototype**

```
qapi_Status_t qapi_QT_Apps_AT_Port_Open(qapi_at_port_t port_id, qapi_at_stream_id_t *stream_id,
qapi_at_resp_func_cb_t cb);
```

```
typedef enum qpi_at_port_e
{
```

```
    QAPI_AT_PORT_0 = 0,
    QAPI_AT_PORT_1,
    QAPI_AT_PORT_2,    //Reserved
    QAPI_AT_PORT_3,    //Reserved
    QAPI_AT_PORT_MAX
} qapi_at_port_t;

typedef int2   qapi_at_stream_id_t;

typedef void    (*qapi_at_resp_func_cb_t)(qapi_at_pipe_data_t *data);    //response callback
```

- **Parameters**

port_id:
[in] Port id of the ATC pipe. Customer could specifies a port for execute AT Command.

stream_id:
[out] Pointer, specific stream port returned by qapi_Status_t qapi_QT_Apps_AT_Port_Open()

cb:
[in] Response callback function.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.2.1.2. qapi_QT_Apps_AT_Port_Close

This function used to Close AT command port with stream port id.

- **Prototype**

```
void qapi_QT_Apps_AT_Port_Close(qapi_at_stream_id_t stream_id);

typedef int2   qapi_at_stream_id_t;
```

- **Parameters**

stream_id:
[in] Specific stream port returned by qapi_Status_t qapi_QT_Apps_AT_Port_Open()

- **Return Value**

None.

8.2.1.3. qapi_QT_Apps_Send_AT

This function used to send AT command in specifial stream port id.

- **Prototype**

```
qapi_Status_t qapi_QT_Apps_Send_AT(qapi_at_stream_id_t stream_id, const char *command_name);
```

```
typedef int2 qapi_at_stream_id_t;
const char *hex_str;
```

- **Parameters**

stream_id:
[in] specific stream port returned by qapi_Status_t qapi_QT_Apps_AT_Port_Open()

hex_str:
[in] String with hex format that will be sent

- **Return Value**

8.2.1.4. qapi_QT_Apps_Send_AT_HexByte

This function used to send string in specifial stream port id with hex format.

- **Prototype**

```
qapi_Status_t qapi_QT_Apps_Send_AT_HexByte(qapi_at_stream_id_t stream_id, const char *hex_str);
```

```
typedef int2 qapi_at_stream_id_t;
const char *hex_str;
```

- **Parameters**

stream_id:
[in] specific stream port returned by qapi_Status_t qapi_QT_Apps_AT_Port_Open()

hex_str:
[in] String with hex format that will be sent

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.2.1.5. qapi_QT_Apps_Send_AT_General

This function used to send typeless data in special stream port ID. Caller can tranmit any type of data through stream port. And response will be notified in callback which registed in qapi_QT_Apps_AT_Port_Open().

- **Prototype**

```
qapi_Status_t qapi_QT_Apps_Send_AT_General(qapi_at_stream_id_t stream_id, void *trans_data, uint32 trans_len);
```

```
typedef int2 qapi_at_stream_id_t;
void *trans_data;
uint32 trans_len;
```

- **Parameters**

stream_id:
[in] specific stream port returned by qapi_Status_t qapi_QT_Apps_AT_Port_Open()

trans_data:
[in] transmission data

trans_len:
[out] The length of data transmited

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.2.2. Example

1. AT port open example:

```
qapi_Status_t status = QAPI_QT_ERR_OK;

status = qapi_Status_t qapi_QT_Apps_AT_Port_Open (0，stream_id_0，function callback);
if(status == QAPI_QT_ERR_OK)
{
    //AT command port will open，and a callback will registered
}
```

```
else
{
    //IOT_INFO("AT command port open failed, status %x", status);
}
```

2. AT port close:

```
void qapi_QT_Apps_AT_Port_Close(stream_id_0);
```

3. Send AT

```
qapi_Status_t status = QAPI_QT_ERR_OK;

status = qapi_QT_Apps_Send_AT(stream_id_0, "ATI\r\n");
if(status == QAPI_QT_ERR_OK)
{
      //An AT command will send
}
else
{
    //IOT_INFO("An AT command send failed, status %x", status);
}
```

4. Send AT HexByte:

```
qapi_Status_t status = QAPI_QT_ERR_OK;

status=qapi_Status_t qapi_QT_Apps_Send_AT_HexByte(stream_id_0, hex_str);
if(status == QAPI_QT_ERR_OK)
{
      //A string with hex format will send
}
else
{
    //IOT_INFO("A string with hex format send failed, status %x", status);
}
```

5. Send AT General:

```
qapi_Status_t status = QAPI_QT_ERR_OK;

status= qapi_Status_t qapi_QT_Apps_Send_AT_General (stream_id_0, trans_data, trans_len);
if(status == QAPI_QT_ERR_OK)
```

```
{
    //A typeless data will send，and response will be notified in callback
}
else
{
    //IOT_INFO("A string with hex format send failed, status %x", status);
}
```

8.3. ADC API

8.3.1. API Functions

The module provides two analog-to-digital converter (ADC) interfaces. The voltage value on ADC pins can be read via **AT+QADC=<port>** command, through setting **<port>** into 0, 1.

8.3.1.1. qapi_QT_ADC_READ

This function used to read the value of ADC. The results will be stored in the result structure

- **Prototype**

```
qapi_Status_t qapi_QT_ADC_READ(const char *pChn_Name,qapi_ADC_Read_Result_t *result)
```

```
typedef struct
{
    AdcResultStatusType eStatus;
    uint32_t    nToken;
    uint32_t    nDeviceIdx;
    uint32_t    nChannelIdx;
    int32_t     nPhysical;
    uint32_t    nPercent;
    uint32_t    nMicrovolts;
    uint32_t    nCode;
} qapi_ADC_Read_Result_t;
```

- **Parameters**

pChn_Name:
[in] Channel name of the specified ADC channel.
result:
[out] The result of ADC channel read.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.3.2. Example

```
int quectel_task_entry (void)
{
    qapi_Status_t status = QAPI_QT_ERR_OK;
    qapi_ADC_Read_Result_t result;
    status = qapi_QT_ADC_Read(Channel_Name_ADC1, &result);
    if(status == QAPI_QT_ERR_OK)
    {
        // Read the value of ADC.
        qt_uart_dbg ("Channel_Name %s Voltage: %d mV\n",
                                Channel_Name_ADC1,result.nMicrovolts/1000);
    }
    else
    {
        //IOT_INFO("ADC value read failed, status %x", status);
    }
}
```

8.4. Location API

8.4.1. API Functions

BG96-QuecOpen includes a fully integrated global navigation satellite system solution that supports Gen8C-Lite of Qualcomm (GPS, GLONASS, BeiDou, Galileo and QZSS). BG96-QuecOpen supports standard NMEA-0183 protocol, and outputs NMEA sentences at 1Hz data update rate via USB interface by default.

8.4.1.1. qapi_QT_Loc_Start

This function used to Start GNSS positioning.

- **Prototype**

```
qapi_Status_t qapi_QT_Loc_Start(qtLocEventRegMaskT evt_mask, qapi_QT_Loc_CB_t cb)
```

```
typedef uint64_t qtLocEventRegMaskT;
#define QT_LOC_EVENT_MASK_POSITION_REPORT    ((qtLocEventRegMaskT)0x00000001ull)
```

```
#define QT_LOC_EVENT_MASK_GNSS_SV_INFO      ((qtLocEventRegMaskT)0x00000002ull)
#define QT_LOC_EVENT_MASK_NMEA              ((qtLocEventRegMaskT)0x00000004ull)

typedef void (*qapi_QT_Loc_CB_t)(char *nmea_data);
```

- **Parameters**

evt_mask:
[in] Specified the event.

cb :

[in] Callback function. When any event is triggered, will invoke this callback function.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.4.1.2. qapi_QT_Loc_Stop

This function used to stop GNSS positioning.

- **Prototype**

```
qapi_Status_t qapi_QT_Loc_Stop(void)
```

- **Parameters**

None.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.4.2. Example

```
void quectel_loc_nmea_cb(char *nmea_data)
{
    qapi_UART_Transmit(uart1_conf.hdlr, nmea_data, strlen(nmea_data), NULL);
    qapi_Timer_Sleep(50, QAPI_TIMER_UNIT_MSEC, true);
}
int quectel_task_entry(void)
{
    qapi_Status_t status = QAPI_QT_ERR_OK;

    status = qapi_QT_Loc_Start(QT_LOC_EVENT_MASK_NMEA, quectel_loc_nmea_cb);
```

```
    qt_uart_dbg(uart1_conf.hdlr,"START status %d", status);
    qapi_Timer_Sleep(120, QAPI_TIMER_UNIT_SEC, true);
    status = qapi_QT_Loc_Stop();
    qt_uart_dbg(uart1_conf.hdlr,"STOP status %d", status);
}
```

8.5. FOTA API

8.5.1. API Functions

Quectel BG96 module supports DFOTA (Delta Firmware Upgrade Over-the-air) function, which support upgrade module from an old firmware to a new firmware with a delta firmware package.

8.5.1.1. qapi_QT_Fota_Http_Download_Start

This function used to start fota package download.

- **Prototype**

```
qapi_Status_t qapi_QT_Fota_Http_Download_Start(char* url, qapi_Fota_Http_dl_CB_t response_cb);
```

```
typedef void(*qapi_Fota_Http_dl_CB_t)(int error_id);
/*
*error_id:[out]
*0              upgrade success
*504            firmware upgrade failed
*505            upgrade package not exist
*506            upgrade package check failed
*others         unknown error
*/
```

- **Parameters**

url:
[in] fota package path. For example: HTTP://220.180.239.212:8005/BG96_112A_119.zip. The url must be started with "http://" or "https://", not case sensitive.
response_cb:
[in] response callback, This callback will come when download failed or success.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.5.1.2. qapi_QT_Remove_Fota_Package

This function used to remove local fota package.

- **Prototype**

```
qapi_Status_t qapi_QT_Remove_Fota_Package(void);
```

- **Parameters**

None

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.5.1.3. qapi_QT_Fota_Update_Start

This function used to start Fota upgrade. Before use this QAPI, the delta firmware should be downloaded/uploaded to the App file system. Both the path and the package name cannot be modified. Customers can use the QEFS Explorer tool provided by Quectel to upload the upgrade package.

- **Prototype**

```
qapi_Status_t qapi_QT_Fota_Update_Start(qapi_QT_Fota_Response_CB_t response_cb, qapi_QT_Fota_Upgrade_Progress_CB_t upgrade_progress_cb);
```

```
typedef void (*qapi_QT_Fota_Response_CB_t)(short int error_id);  //response callback
typedef void (*qapi_QT_Fota_Upgrade_Progress_CB_t)(unsigned char phase, unsigned char percent);//response callback
```

- **Parameters**

response_cb:
[out] response callback,This callback will come when fota upgrade failed or success.

upgrade_progress_cb:
[out] upgrade progress callback, This callback will come every one percent.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.5.2. Example

- HTTP download start:

```
qapi_Status_t status = QAPI_QT_ERR_OK;
status = qapi_QT_Fota_Http_Download_Start(url, cb);
if(status == QAPI_QT_ERR_OK)
{
    // Fota_Http_Download_Start.
}
else
{
    //IOT_INFO("Fota_Http_Download_Start failed, status %x", status);
}
```

- Remove FOTA package:

```
qapi_Status_t status = QAPI_QT_ERR_OK;
status = qapi_Status_t qapi_QT_Remove_Fota_Package(void);
if(status == QAPI_QT_ERR_OK)
{
    // .Remove fota package
}
else
{
    //IOT_INFO("Remove fota package failed, status %x", status);
}
```

- Start FOTA upgrade

```
qapi_Status_t status = QAPI_QT_ERR_OK;
status=qapi_QT_Fota_Update_Start((qapi_QT_Fota_Response_CB_t)param_1,
                                 (qapi_QT_Fota_Upgrade_Progress_CB_t)param_2);
if(status == QAPI_QT_ERR_OK)
{
    // .Fota update start
}
else
{
    //IOT_INFO("Fota update start failed, status %x", status);
}
```

8.6. PSM API

8.6.1. API function

BG96 module has a PSM_IND pin to indicate that module is enable PSM or disable PSM. Sometimes customer want to use this pin as a normal GPIO which can use below QAPI to reconfigure it. The configuration will take effect after module reset.

8.6.1.1. qapi_QT_PSM_SrvCfg_Set

This function used to set PSM core server configuration

- **Prototype**

```
qapi_Status_t qapi_QT_PSM_SrvCfg_Set(int gpio_enable);
```

- **Parameters**

gpio_enable:
[in] Configure PSM_IND pin as a PSM indicator or as a GPIO..

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.6.1.2. qapi_QT_PSM_SrvCfg_Get

This QAPI used to get the current configuration.
.

- **Prototype**

```
qapi_Status_t qapi_QT_PSM_SrvCfg_Get(int *gpio_enable);
```

- **Parameters**

gpio_enable:
[out] Store current configuration.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.
.

8.6.2. Example

```
int quectel_task_entry(void)
{
    qapi_Status_t status = QAPI_QT_ERR_OK;
    int gpio_enable = 1;
    status= qapi_Status_t qapi_QT_PSM_SrvCfg_Set(gpio_enable);
    if(status == QAPI_QT_ERR_OK)
    {
        // .Get PSM core server configuration
        If(gpio_enable==0)
        {
            status= qapi_Status_t qapi_QT_PSM_SrvCfg_Set(int gpio_enable);
        }
       else
        {
          //IOT_INFO("PSM mode");
        }
    }
    else
    {
        //IOT_INFO("Get PSM core server configuration failed, status %x", status);
    }

}
```

8.7. WatchDog Services API

8.7.1. API function

BG96-QuecOpen module supports register a software watch dog to monitor specified task in the user application.

8.7.1.1. qapi_QT_Register_Wdog

This function used to for tasks to register a software dog services in the QuecOpen application.

- **Prototype**

```
qapi_Status_t qapi_QT_Register_Wdog( uint32 time,uint32 *dog_id);
```

- **Parameters**

time:
[in] starvation detection threshhold in milliseconds..

*dog_id:
[out] for feed watchdog function use.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.7.1.2. qapi_QT_Kick_AT_Wdog

This function used to report to the Watchdog task not timeout.

- **Prototype**

```
qapi_Status_t qapi_QT_Kick_Wdog(uint32 id );
```

- **Parameters**

id:
[in] specific id returned by qapi_Status_t qapi_QT_Register_Wdog().

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.7.1.3. qapi_QT_Stop_Wdog_AT

This function used to Set the sw dog_enable on/off.

- **Prototype**

```
qapi_Status_t qapi_QT_Stop_Wdog(boolean mode);
```

- **Parameters**

mode:
[in] enable or disable sw watchdog, default is disable.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.7.2. Example

8.7.2.1. Register Wdog Service

```
#define TIME 60000
#define ENABLE 1
#define DISABLE 0
Uint32 ID;
int quectel_task_entry(void)
{
    qapi_Status_t status = QAPI_QT_ERR_OK;
    int gpio_enable;
    status= qapi_QT_Register_Wdogt(TIME,&ID);
    if(status == QAPI_QT_ERR_OK)
    {
       qapi_QT_Stop_Wdog(DISABLE); //Set the sw dog_enable on, if you set true,you need not feed dog
       While(1)
       {
           qapi_Timer_Sleep(5, QAPI_TIMER_UNIT_MSEC, TRUE);
           qapi_QT_Kick_Wdog(ID);        // feed Wdog, if not feed Wdog in time.module will dump
       }
     }
}
```

8.7.2.2. Use WatchDog Sahara Mode

The default setting is Sahara DUMP mode when module met crash issue. Which means, module default setting is used for collect the DUMP file for furture analysis. But most of the time customer not want the module in DUMP mode because module always hanged and not reset itself. If custsomer want module reset automaticaly when met crash issue, should set module with Sahara reset mode by *qapi_QT_Sahara_Mode_Set*.

```
#define TIME 60000
#define ENABLE 1
#define DISABLE 0
Uint32 ID;
int quectel_task_entry(void)
{
          qapi_Status_t status = QAPI_QT_ERR_OK;
           qapi_QT_Sahara_Mode_Set(0);
           status= qapi_QT_Register_Wdogt(TIME,&ID);
           if(status == QAPI_QT_ERR_OK)
           {
```

```
            While(1)
            {
               qapi_Timer_Sleep(5, QAPI_TIMER_UNIT_MSEC, TRUE);
             }
          }
}
```

8.8. USB API

8.8.1. API function

BG96-QuecOpen module supports enable/disable the USB function through the QAPI in the QuecOpen user application. This configuration will take effect after module reset.

8.8.1.1. qapi_QT_Usbmode_Set

This QAPI used to configure enable or disable USB function.

- **Prototype**

```
qapi_Status_t qapi_QT_Usbmode_Set(boolean mode);
```

- **Parameters**

mode:
[in] Enable or disable the USB function.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.8.1.2. qapi_QT_Get_USB_Event

This QAPI used to get the event of USB.
USB events list as below:

0 DEVICE CONNECT
1 DEVICE DISCONNECT
2 DEVICE SUSPEND
3 DEVICE RESUME
4 DEVICE RESUME COMPLETED
5 DEVICE REMOTE WAKEUP
6 DEVICE CONFIGURED

7 DEVICE UNCONFIGURED
8 DEVICE RESET
9 DEVICE SPEED CHANGE

- **Prototype**

```
qapi_Status_t qapi_QT_Gert_USB_Event(uint32 *usb_evt);
```

- **Parameters**

usb_evt:
[in] Pointer, store the USB event.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.8.2. Example

```
#define ENABLE 1
#define DISABLE 0
 int quectel_task_entry(void)
 {
     qapi_Status_t status = QAPI_QT_ERR_OK;
     status= qapi_Status_t qapi_QT_Usbmode_Set(ENABLE);    //Disable USB function
     if(status == QAPI_QT_ERR_OK)
     {
         while(1)
         {
             qapi_Timer_Sleep(5, QAPI_TIMER_UNIT_MSEC, TRUE);
         }
     }
 }
```

8.9. Random Number API

8.9.1. API function

This QAPI used to get the random number which generated by hardware random number generator.

8.9.1.1. qapi_QT_Random_Data_Get

This function used to get the random number.

- **Prototype**

```
qapi_Status_t qapi_QT_Random_Data_Get (uint16 prng_size, uint8* prng_data)
```

- **Parameters**

prng_size:
[in] Get the number of bytes required. Range(1,2,4,8…512)

prng_data:
[out] Pointer, store random number.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.9.2. Example

```
unsigned char dat_tab[20];

in quectel_task_entry(void)
{
    qapi_Status_t status = QAPI_QT_ERR_OK;
    status= qapi_QT_Random_Data_Get(10,dat_tab));          //get a random number of ten bytes
    if(status == QAPI_QT_ERR_OK)
    {
        for(i=0;i<10;i++)
        {
            qt_uart_dbg(uart1_conf.hdlr,"dat_tab[i]    %d",dat_ tab[i]);
         }
    }
}
```

8.10. PWRKEY API

8.10.1. API function

This function used to disable or enable the power off function of the Power Key button after module power on.

8.10.1.1. qapi_QT_PWRKEY_Switch_Set

- **Prototype**

```
qapi_Status_t qapi_QT_PWRKEY_Switch_Set(uint32 mode)
```

- **Parameters**

mode:
[in] Enable or disable the power off function of the Power Key.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.10.1.2. qapi_QT_PWRKEY_State_Get

This function used to get current configuration.

- **Prototype**

```
qapi_Status_t qapi_QT_PWRKEY_State_Get(uint32 *val);
```

- **Parameters**

val:
[out] Pointer, store current configuration.

- **Return Value**

QAPI_QT_ERR_OK on success, others on error.

8.10.2. Example

```
#define TRUE 1

in quectel_task_entry(void)
{
    qapi_Status_t status = QAPI_QT_ERR_OK;
    status= qapi_QT_PWRKEY_Switch_Set(TRUE);          //Disable the power off function of the
                                                      //Power Key
    if(status == QAPI_QT_ERR_OK)
    {
      qapi_QT_PWRKEY_State_Get(mode); //Get sw the PWRKEY state
       While(1)
       {
          qapi_Timer_Sleep(5, QAPI_TIMER_UNIT_MSEC, TRUE);
        }
    }
}
```

9 BG96-QuecOpen GPIO Mapping

This section mainly introduces the GPIO mapping of the BG96 QuecOpen, also including the mapping of related peripherals, like UART, IIC, SPI and so on.

9.1. GPIO Mapping

The following table shows the pin definition, and GPIO pull up/down resistance of BG96-QuecOpen module. Table alson includes the mapping of related peripherals.

Table 5: Multiplexing Pins

Pin Name	Pin No.	Mode 1	Mode 2	Mode 3	Mode 4	Reset[1)]	Interrupt*	Remark
GPIO01	4	GPIO_23	GPIO_23	SPI1_CLK	/	B-PU	No	BOOT_ CONFIG_4
GPIO02	5	GPIO_20	UART1_TX	SPI1_MOSI	/	B-PD	Yes	
GPIO03	6	GPIO_21	UART1_RX	SPI1_MISO	/	B-PD	Yes	
GPIO04	7	GPIO_22	GPIO_22	SPI1_CS_N	/	B-PD	Yes	
GPIO05	18	GPIO_11	/	SPI2_CLK	I2C2_SCL	B-PU	Yes	
GPIO06	19	GPIO_10	/	SPI2_CS_N	I2C2_SDA	B-PD	No	
GPIO07	22	GPIO_09	UART2_RX	SPI2_MISO	/	B-PD	Yes	
GPIO08	23	GPIO_08	UART2_TX	SPI2_MOSI	/	B-PD	Yes	
GPIO09	26	GPIO_15	GPIO_15	/	/	B-PD	No	
GPIO10	27	GPIO_12	UART3_TX	/	/	B-PD	Yes	
GPIO11	28	GPIO_13	UART3_RX	/	/	B-PD	Yes	
GPIO19	40	GPIO_19	/	/	I2C1_SCL	B-PD	No	
GPIO20	41	GPIO_18	/	/	I2C2_SDA	B-PD	No	
GPIO21	64	GPIO_07	/	/	/	B-PD	No	

NOTE

1. The pin functions in Model 1/2/3/4 take effect only after software configuration.
2. The BOOT_CONFIG pin (GPIO01) cannot be pulled up before startup.
3. "*" means under development.
4. "/" means not supported.
5. The "Pin Name" means these pins named in QuecOpen solution. The Corresponding pin number is "Pin No".
6. Mode 1 ~ Mode 4 just means those pins have 4 funtions with different configuration.

9.1.1. GPIOs

BG96-QuecOpen supports 14 GPIOs. Customer can configure each GPIO by QAPI in their own applciaiton. As an output function, customer can configure the driver strength for specified GPIO, max to 16mA.
Customer can use related function with QAPIs which descriptiped in "80-P8101-14 C Qualcomm Application Programming Interface for MDM9206 ThreadX OS Interface Specification", chapter 11.2, PMM APIs.
Some of GPIOs support Interrupt function. Details please refer to the Table 1. Customer can configure the specified GPIO as an interrupt with releated QAPIs which descriptiped in "80-P8101-14 C Qualcomm Application Programming Interface for MDM9206 ThreadX OS Interface Specification", chapter 11.1, "GPIO Interrupt Controller APIs. PMM APIs".

9.1.2. UART Interfaces

The module provides four UART interfaces: Main UART, UART1, UART2 and UART3.

- Main UART interface can only be used for AT command communication.
- UART1, UART2 and UART3 interfaces are used for communication and data transmission with peripherals, and can also be multiplexed into other functions.

The following tables show the pin definition of the four UART interfaces.

Table 6:Pin Definition of Main UART Interface

Pin Name	Pin No.	I/O	Description	Comment
DTR	30	DI	Data terminal ready	1.8V power domain
RXD	34	DI	Receive data	1.8V power domain
TXD	35	DO	Transmit data	1.8V power domain
CTS	36	DO	Clear to send	1.8V power domain
RTS	37	DI	Request to send	1.8V power domain

DCD	38	DO	Data carrier detection	1.8V power domain
RI	39	DO	Ring indicator	1.8V power domain

NOTE

Main UART interface only for AT Command communication. It cannot be confugired or used in customer own application.

Table 7:Pin Definition of UART1 Interface

Pin Name	Pin No.	I/O	Mode 1	Mode 2	Mode 3	Mode 4	Remark
GPIO01	4	IO	GPIO_23	GPIO_23	SPI1_CLK	/	1.8V power domain. Cannot be pulled up before startup.
GPIO02	5	IO	GPIO_20	**UART1_TX**	SPI1_MOSI	/	1.8V power domain.
GPIO03	6	IO	GPIO_21	**UART1_RX**	SPI1_MISO	/	1.8V power domain.
GPIO04	7	IO	GPIO_22	GPIO_22	SPI1_CS_N	/	1.8V power domain.

NOTES

1. In QuecOpen application, use “QAPI_UART_PORT_001_E” to select and configure UART1.
2. UART1 port does not support flow control.

Table 8:Pin Definition of UART2 Interface

Pin Name	Pin No.	I/O	Mode 1	Mode 2	Mode 3	Mode 4	Remark
GPIO05	18	IO	GPIO_11	/	SPI2_CLK	I2C2_SCL	1.8V power domain.
GPIO06	19	IO	GPIO_10	/	SPI2_CS_N	I2C2_SDA	1.8V power domain.
GPIO07	22	IO	GPIO_09	**UART2_RX**	SPI2_MISO	/	1.8V power domain.
GPIO08	23	IO	GPIO_08	**UART2_TX**	SPI2_MOSI	/	1.8V power domain.

NOTE

1. In QuecOpen application, use “QAPI_UART_PORT_002_E” to select and configure UART2.
2. UART2 port does not support flow control.

Table 9: Pin Definition of UART3 Interface

Pin Name	Pin No.	I/O	Mode 1	Mode 2	Mode 3	Mode 4	Remark
GPIO10	27	IO	GPIO_12	**UART3_TX**	/	/	1.8V power domain.
GPIO11	28	IO	GPIO_13	**UART3_RX**	/	/	1.8V power domain.

NOTE

In QuecOpen application, use “QAPI_UART_PORT_003_E” to select and configure UART3.

9.1.3. I2C Interfaces

BG96-QuecOpen provides two Inter-Integrated Circuit (I2C) interfaces for communication, which support high-speed mode and not support multi-master. I2C interfaces uses GPIOs configured as open-drain outputs, and the pull-up resistors should be provided externally.
The following table shows the pin definition.

Table 10: Pin Definition of the I2C1 Interface

Pin Name	Pin No.	I/O	Mode 1	Mode 2	Mode 3	Mode 4	Remark
GPIO19	40	IO	GPIO_19	/	/	**I2C1_SCL**	1.8V power domain.
GPIO20	41	IO	GPIO_18	/	/	**I2C2_SDA**	1.8V power domain.

NOTE

In QuecOpen application, use “QAPI_I2CM_INSTANCE_004_E” to select and configure I2C1.

Table 11: Pin Definition of the I2C2 Interface

Pin Name	Pin No.	I/O	Mode 1	Mode 2	Mode 3	Mode 4	Remark
GPIO05	18	IO	GPIO_11	/	SPI2_CLK	**I2C2_SCL**	1.8V power domain.
GPIO06	19	IO	GPIO_10	/	SPI2_CS_N	**I2C2_SDA**	1.8V power domain.

NOTE

1. In QuecOpen application, use “QAPI_I2CM_INSTANCE_005_E” to select and configure I2C2.

9.1.4. SPI Interfaces

BG96-QuecOpen provides two SPI interfaces which support only master mode with a maximum clock frequency up to 50MHz.
The following table shows the pin definition.

Table 12: Pin Definition of the SPI1 Interface

Pin Name	Pin No.	I/O	Mode 1	Mode 2	Mode 3	Mode 4	Remark
GPIO01	4	IO	GPIO_23	GPIO_23	**SPI1_CLK**	/	1.8V power domain. Cannot be pulled up before startup.
GPIO02	5	IO	GPIO_20	UART1_TX	**SPI1_MOSI**	/	1.8V power domain.
GPIO03	6	IO	GPIO_21	UART1_RX	**SPI1_MISO**	/	1.8V power domain.
GPIO04	7	IO	GPIO_22	GPIO_22	**SPI1_CS_N**	/	1.8V power domain.

NOTE

1. In QuecOpen application, use “QAPI_SPIM_INSTANCE_6_E” to select and configure SPI1.

Table 13: Pin Definition of the SPI2 Interface

Pin Name	Pin No.	I/O	Mode 1	Mode 2	Mode 3	Mode 4	Remark
GPIO05	18	IO	GPIO_11	/	**SPI2_CLK**	I2C2_SCL	1.8V power domain.
GPIO06	19	IO	GPIO_10	/	**SPI2_CS_N**	I2C2_SDA	1.8V power domain.
GPIO07	22	IO	GPIO_09	UART2_RX	**SPI2_MISO**	/	1.8V power domain.
GPIO08	23	IO	GPIO_08	UART2_TX	**SPI2_MOSI**	/	1.8V power domain.

NOTE

1. In QuecOpen application, use “QAPI_SPIM_INSTANCE_5_E” to select and configure SPI2.

10 Appendix A References

Table 14: Related Documents

SN	Document Name	Remark
[1]	Quectel_BG96_QEFS_Explorer_ User_Guide	QEFS Explorer tool user guide
[2]	Quectel_BG96-QuecOpen_Hardware_Design	BG96 QuecOpen Hardware Design
[3]	Quectel_BG96_DFOTA_User_Guide	Quectel BG96 DFOTA User Guide
[4]	Quectel_BG96_GNSS_AT_Commands_Manual	BG96 GNSS Manual
[5]	Quectel_BG96-QuecOpen_WatchDog_Application _Note	BG96 QuecOpen WatchDog Application Note
[6]	80-P8101-14 B Qualcomm Application Programming Interface for MDM9206 ThreadX OS Interface Specification	Qualcomm QAPI Specification
[7]	Quectel_BG96-QuecOpen_Random_Number_ Application_Note	BG96 QuecOpen Random Number Application Note
[8]	Quectel_BG96-QuecOpen_Compiler_Installation_ Guide	Quectel BG96 QuecOpen Compiler Installation
[9]	Quectel_BG96_AT_Commands_Manual	Quectel BG96 AT Commands Manual

Table 15: Terms and Abbreviations

Abbreviation	Description
API	Application Programming Interface
OS	Operating System
HTTP	Hyper Text Transfer Protocol
SDK	Software Development Kit