OS-IoT BG96 Build, Installation and User Guide

Revision 1. January 18th, 2019.

Iain Sharp – ATIS

Generated as part of the OS-IoT project (os-iot.org).

Contents

Revision History

Version	Date	Notes
Initial	Jan 2019	First version
Rev 1	Jan 18th, 2019	Updates following testing on NB-IoT

Abbreviations

AE	Application Entity (oneM2M)
APN	Access Point Name (3GPP)
CA	Certificate Authority
CSE	Common Service Entity (oneM2M)
DAM	Dynamic Application Module (ThreadX)
EVB	Evaluation Board
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol (Secure)
NB-IoT	Narrow Band, Internet of Things (3GPP)
NEMA	National Electrical Manufactures Association (Standard for GPS info over serial interface)
PSM	Power Saving Mode
RAT	Radio Access Technology (3GPP)
SDK	Software Development Kit
SIM	Subscriber Identity Module
USB	Universal Serial Bus

Introduction

The BG96 is a cellular IoT module manufactured by Quectel based on the Qualcomm 9206 chipset. The module supports development of application software to run on the BG96 using the ThreadX Downloadable Application Module (DAM) mechanism. This allows applications to be built directly on the BG96 without the need for an external CPU – meaning that the number of parts and the power consumption for IoT devices is reduced compared to architectures that use an external CPU.

The OS-IoT BG96 project provides an open source implementation of a oneM2M client (i.e. an Application Entity, AE) in C that runs on the BG96. The project contains a reusable library of oneM2M functions as well as a demo application which shows how to use the library. The library provides support for the most important oneM2M resource types and HTTP or HTTPS communication to the oneM2M CSE server.

Development tools, SDKs and documentation are available from Quectel and Qualcomm. However, it is hard to follow the end-to-end process for getting started doing development as the tooling is rather new and documentation is confusing. This guide aims to cover the essentials to get to the point of installing and running a first application. It should be read in conjunction with the documentation provided in the SDKs.

In order to install the tools needed follow the instructions in the "Quectel QuecOpen Application Note" and the "Quectel Windows USB Driver Installation Guide".

Development Options

The BG96 can run firmware based on ThreadX version 2 or ThreadX Version 3. SDKs are available for both versions, but the ThreadX V2 SDK requires an ARM v5 compiler from ARM which is only available via a commercial license. The ThreadX Version 3 SDK uses a compiler provided by Qualcomm based on the CLANG/LLVM Open Source tool chain.

The OS-IoT BG96 project has been developed and tested using the ThreadX Version 3 firmware and corresponding SDK only.

SDK files may be obtained from either Quectel or Qualcomm. Other tools used may be obtained from Quectel. This document describes working with the Quectel ThreadX version 3 SDK as this seems to contain a superset of the Qualcomm SDK. Note however that Quectel has their own API extensions that are extend parts of the Qualcomm API. Therefore, even if building using an SDK from Quectel you may choose to just use the Qualcomm API features.

Though the SDK can run on Linux or Windows most of the tooling for the BG96 is windows-only, so this document assumes that the SDK will be run on Windows.

Equipment and Initial Configuration

This guide is based on using the Quectel BG96 UMTS&LTE EVB (Evaluation Board) version 2.2. This is available commercially from various Quectel distributors. The EVB should be prepared in accordance with the documentation provided in the kit. Preparation includes:

- Installing the BG96 breakout board on the main EVB
- Mounting the cellular antenna (the larger and thicker of those provided) and connecting to the "main" socket (J301) on the BG96 breakout board.
- Insertion of a SIM in the EVB's SIM socket

It is not necessary to install a WiFi module, GPS antenna or codec module on the EVB.

Note that the power for the EVB is controlled by both a physical slide switch “POWER” S201 and a button “PWRKEY” S302. To power the board on set the POWER switch to the ON position and press the PWRKEY. A second press of the PWRKEY will power the EVB down. The “RESET” S303 button will reset the EVB.

The EVB kit includes a USB memory stick that contains the Windows drivers. Following the instructions provided install these drivers. Upon installation of the drivers the EVB provides three virtual COM ports over USB as well as two physical COM ports which can be accessed via the adaptor provided in the EVB kit. In this document the following naming is used for the ports:

- AT Port - the virtual AT Command Command COM port (also doubles as the port used for the Qualcomm QFLOG tool - see later)
- DM Port - the virtual COM port for device management
- NEMA Port - the virtual COM port for NEMA
- MAIN COM - The physical COM port on the left hand side of the EVB labelled “COM1 (MAIN)”
- DEBUG COM - The physical COM port on the right hand side of the EVB labelled “COM2 (DEBUG)”

Firmware Upgrade

Currently BG96 modules are being distributed with ThreadX v2 firmware. Therefore, it will be necessary to upgrade the firmware to ThreadX v3 before installing OS-IoT BG96. It is recommended that the following firmware version, or newer, is used:

- BG96MAR03A03M1G_01.002.01.002

Older versions of firmware may have issues with support for NB-IoT.

You will need to obtain from Quectel the following items:

- A set of suitable firmware update files. This consists of a folder called “update” which contains files with “.mbn” and other extensions.
- The “QFlash” firmware update application, version 4.9 was used in this example.

Connect the EVB to a PC and upgrade the firmware using the QFlash application as described in the Quectel documentation.

Other Quectel Tools

Before attempting development, it is worth obtaining and getting familiar with the other tools provided by Quectel as part of their development package. These are individually documented, so this section just provides a top-level explanation of their purpose.

- QCOM - this is a terminal emulator that allows you to send and receive to and from any COM port. The main practical use is to interact with the DEBUG COM on the EVB. Similar functionality to QCOM is available in many applications (e.g. PUTTY).
- Qnavigator - this is an application to get basic status information about the EVB (including the firmware version). It is also a convenient tool to send AT commands to the EVB. It can connect to the AT Port (if not used by QFLOG – see below) or the MAIN COM port on the EVB.

- QEFS_Explorer - this is a graphical application allows you to view and add files to the user file system on the BG96.It can be used as an alternative to the Qualcomm QFLOG tool for managing application files on the BG96. It uses the DM Port. Note that one problem with QEFS_Explorer is that it can crash if the EVB is rebooted while it is running. Note also that when uploading files a lot of options are presented. The default options appear to be correct.

Qualcomm QFLOG Tool

Qualcomm provide a Python tool called QFLOG which can be used to manage application files on the BG96. Qualcomm provide documentation for this in the "LTE IoT SDK User Guide for MDM". The file management capabilities seem to be a pure subset of those provided by Quectel QEFS_Explorer so it is a matter of preference which approach to use. QFLOG also provides logging and debugging functionality as described in the Qualcomm documentation.

In order to use QFLOG it is necessary to enable the QFLOG protocol on the AT Port. This is done by issuing the following AT commands and rebooting:

```
AT+QCFGEXT="qflogport",1
AT+QCFGEXT="qflogen",1
```

Note that if QFLOG is enabled then the AT Port is dedicated only for QFLOG and is no longer available for AT commands. AT commands can still be issued via the MAIN COM port.

Installing a User Application on the BG96

A user application (ThreadX DAMs) is installed on the BG96 by copying two files in to the "datatx" directory in the user file system. This can be done using QEFS_Explorer or QFLOG. The two files are:

- A ".bin" file which contains the compiled and linked application.
- A file "oem_app_path.ini" which contains the path to the "bin" file on the BG96. For example the file may contain the string "/datatx/quectel_demo_uart.bin"

Once these files are copied to the BG96 the application will be run on the next reboot.

Note that when an application is running on the BG96 its ".bin" file can become locked. If this happens then to remove the file you must first delete the oem_app_path.ini then reboot to start the BG96 without the application and then remove the ".bin" file. Section 6 of the "Quectel BG96 QuecOpen Application Note" describes how to upgrade the application software in a more convenient way that doing a total removal and installation of new software.

ATIS OS-IoT BG96 Contents

The source code and pre-built binary files for ATIS OS-IoT BG96 may be obtained using the following command:

```
git clone https://atis.codebasehq.com/atis-os-iot/atis-os-iot-bg96.git
```

The following table describes the main contents of the repository:

Location	Description
/bin	Pre-built binary for the BG96 and example configuration files

/build	Build intermediate products
/src	Source files for the library and demo application
build_dam_sample_app.bat	Windows script to build the demo application and install on a BG96
copy_file.bat	Windows script to copy a file from the /bin directory to the /datatx directory on the BG96. This can be used to upload configuration files.
delete_oem_app_path.bat	Windows script to delete the file /datatx/oem_app_path.ini from the BG96. This can be used to prevent the demo application being run on reboot of the BG96.
view_log.bat	Windows script to view log information using QFLOG.

Installing and Configuring the ATIS OS-IoT BG96 Demo Application

For initial testing the pre-built demo application may be installed on the BG96. To test operation, you will need:

- Access to a suitable mobile network – including a SIM card and network configuration information (e.g. the APN credentials)
- A compatible oneM2M CSE that is reachable via the mobile network and access credentials for that CSE.

The first step for installation is to prepare configuration files for your situation. Example configuration files may be found in the “/bin” directory. The http_config.json file is required and must be customized to your network. The security_onboard.json file is not required, but is needed if HTTPS is going to be used. For details about the contents of these files refer to the comments at the start of the /src/demo/http.c file. Onboarding security also requires additional files to contain the Certificate Authority (CA) information or the PSK key information.

Once you have prepared your configuration files copy the following files from /bin to the /datatx directory on the BG96:

- http_config.json
- http_dam.bin
- oem_app_path.ini
- security_onboard.json (if used)
- any security credential files referred to in security_onboard.json

Rebooting the BG96 will cause the demo application to run. It will initially onboard any security credentials as specified in the security_onboard.json file. Once this has been done the configuration in the http_config.json file will be used to open a dialog with the oneM2M CSE using the BG96’s radio interface. The demo application will:

1. Contact the CSE using HTTP or HTTPS and create an AE resource (if one doesn’t already exist) and a child Container resource.
2. Every minute create a ContentInstance resource in the Container resource. This currently contains a random number, but a “hook” is provided in the demo application to allow developers to insert their own code.

Debugging information is output to the DEBUG COM port. Use a USB to serial adaptor and terminal (e.g. Quectel QCOM) connected to the port with the following settings:

- Baud: 115200
- Stop bits: 1
- Start bits: 1
- Data Bits: 8
- Parity: None
- Flow Control: None

The OS-IoT BG96 library has been tested with the Eclipse OM2M V1.3.0 CSE. This may be easily downloaded and hosted (e.g. on an Amazon Web Services instance) for testing.

Note that you may need access to the AT Command interface on the BG96 to configure options like RAT selection and power saving mode. If QFLOG is enabled on the BG96 then the USB AT command interface is not available. On the EVB the physical MAIN COM port may be used to send AT commands via a USB to serial adaptor.

Building the ATIS OS-IoT BG96 Demo Application

The scripts provided may be used to build the demo application once the SDK has been installed (see above). All the scripts require customization for the specific installation. Prior to building edit the scripts and customize:

- Directories for the project files and SDK components to match your installation
- The COM port number used for QFLOG operation

The following command will build the application and copy the http_dam.bin file to the BG96 using QFLOG:

```
build_dam_sample_app.bat
```

The option “-i" may be added to also copy the oem_app_path.ini file.

Documentation for the ATIS OS-IoT BG96 Library

The ATIS OS-IoT BG96 library source files contain comments in doxygen format the describe the functions provided by the library. This documentation will also be made available on the “Documentation” page of the OS-IoT web site.

Testing Power Saving Mode (PSM)

If the demo application is built with USE_PSM defined then the application will include simple support for BG96 PSM. In order for PSM to be invoked the following conditions must also be met:

- The active RAT must support PSM according to the current configuration on the BG96.
- PSM must be active on the BG96.

To activate PSM use the following AT command:

```
AT+QPSMS=1
```

See section 6.9 of the "BG96 AT Commands Manual" for more information and configurable options.

The PSM support in the demo application is very basic and is not intended to be fully optimized for any specific application. For example, when waking up from PSM the application will attempt to recreate an AE and Container resource on the CSE even if these have already been created during a previous interaction.

Appendix A – Configuring RAT Selection and Checking the Active RAT

If you wish to check operation on a particular RAT then this section may be useful. It is recommended that AT commands are used to optimize the search behaviour of the BG96 as RAT selection can be extremely slow if all RATs are enabled.

For more information refer to the Quectel “BG96 AT Commands Manual”.

RAT Selection Configuration

If your testing will use one particular type of RAT only it is recommended to use the following commands:

For NB-IoT	**For CAT M1**
AT+QCFG="nwscanmode",3 AT+QCFG="nwscanseq",3 AT+QCFG="iotopmode",1 AT+QCFG="nbsibscramble",0 Notes: 1) NB-IoT is deactivated unless the “iotopmode” configuration is set. 2) The nbsimscramble may not be required on all firmware versions.	AT+QCFG="nwscanmode",3 AT+QCFG="nwscanseq",2 AT+QCFG="iotopmode",0

Band Selection

Network selection may be optimized by specifying the bands to be used with the command AT+QCFG="band". See section 4.2.5 of the "BG96 AT Commands Manual".

Operator Selection

The operator selection may be controlled using the command AT+COPS. See section 6.2 of the "BG96 AT Commands Manual".

For example: manually select operator number “23591”:

```
AT+COPS=1,2,"23591"
```

Check Active RAT

The active RAT may be checked using the command:

```
AT+QCSQ
```

See section 6.18 of the "BG96 AT Commands Manual"

This will return "GSM", "CAT-M1" or "CAT-NB1".

Appendix B – Building Example Applications other than OS-IoT BG96

Building Quectel Example Applications

Source for Quectel examples are in the “quectel/example” directory in the SDK, but the build script is at the root directory of the SDK.

To build a Quectel example application:

- Edit the script “build_quectel_demo_app.bat” to set the paths as described in the “Quectel BG96 QuecOpen Application Note”.
- Build the Quectel example of your choice by following the instructions in the “Quectel BG96 QuecOpen Application Note”. e.g.:

  ```
  C:\quecteldev\sdk>build_quectel_demo_app.bat uart
  ```

NOTE: If you get a link failure due to duplicated symbol definitions you need to remove old “.o” files from previous builds by emptying the “sdk/example/build” directory. This can be done by running “`build_quectel_demo_app.bat -c`”.

Following successful build an “oem_app_path.ini” and “.bin” file will be placed in the “sdk/bin” directory. These may be copied to the EVB as described above.

The UART example will send data to the DEBUG COM port which can be monitored using the QCOM application.

Building Qualcomm Example Applications

The Qualcomm example applications are contained in a directory “Sample_Applications” in the SDK. Each subdirectory contains its own build script. This script should be customized with the file path information for your installation.

Build output is put in the “bin” directory in the example’s subdirectory. Note that the “oem_app_path.ini” is not created by the Qualcomm scripts. This should be manually added before copying the files to the EVB.

The Qualcomm examples use the QFLOG logging mechanism and therefore QFLOG should be enabled. For example, the “hello_world” example only sends information to the QFLOG log stream. This may be viewed with a command similar to:

```
python QFLOG.py -p COM13 VIEW_LOGS
```

(Where COM13 is the COM port number of the AT PORT)