1. Introduction

The QPG7015M comes with a Software Development Kit to unlock its functionality to the programmer. For more information on the the QPG7015M please see its Product Brief . Qorvo showcases its QPG7015M via its QPG7015M Development Kit (see figure 1.1). This programmer manual is intended for Software Developers and explains how to use the QPG7015M DK for functionality verification, how to port the SDK to you custom application processor and how to build custom Bluetooth LE, Zigbee, Matter and/or Thread applications

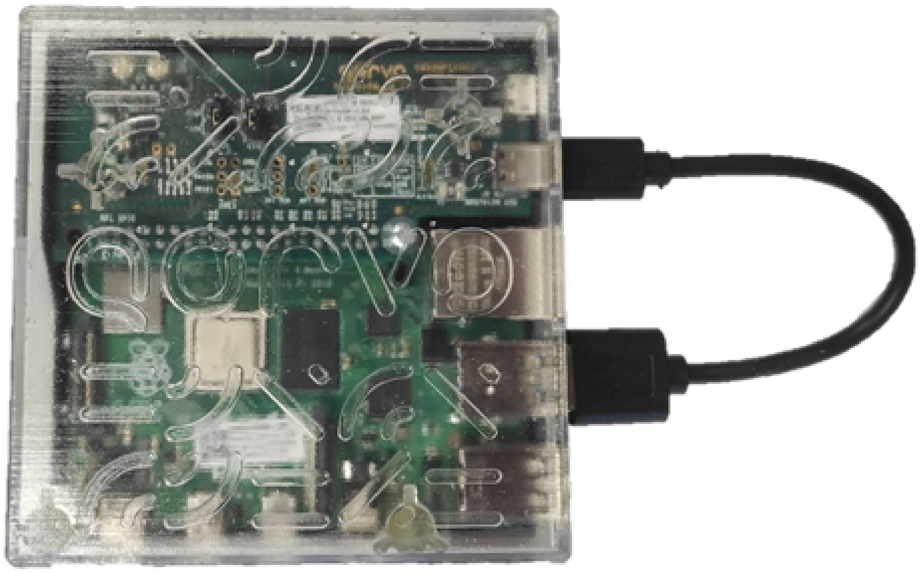


Figure 1.1: QPG7015M Development Kit

Contents

2. Getting started with the Development Kit

This chapter describes:

- The content of the QPG7015M DK
- A quickstart guide of the example applications
- How to access the source files and their documentation of the showcased applications

2.1 Development Kit setup and configuration

This section describes the content of the Development Kit box and how to get up and running with the QPG7015M DK.

2.1.1 QPG7015M Development Kit box content

When ordering the QPG7015M DK, the customer receives a box as shown in figure 2.1. Following items can be found inside:

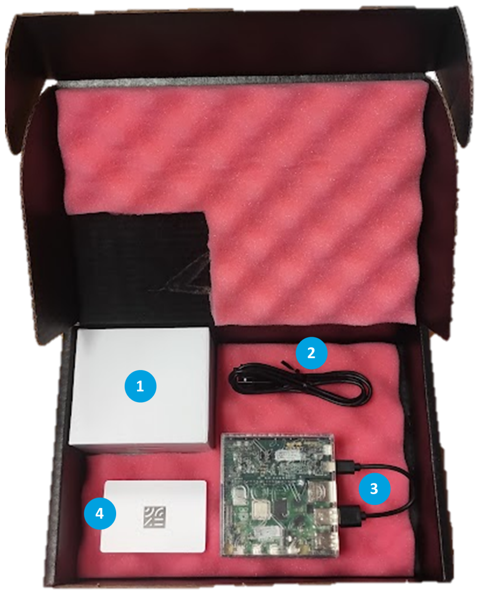


Figure 2.1: QPG7015M Development Kit box content

Table 2.1: QPG7015M DK box content

No.	Name	Description
1	Power supply	Raspberry Pi 4 Power supply, 5.1 V - 3 A
2	USB A ⇔ micro USB cable	Used to access the terminal of the RPi over a serial communication.
3	Documentation card	QR code to the QPG7015M DK Product Brief
4	QPG7015M Development Kit	QPG7015M Radio board mounted on a RPi4 host, running the QPG7015M SDK

2.1.2 Accessing the QPG7015M Development Kit's terminal

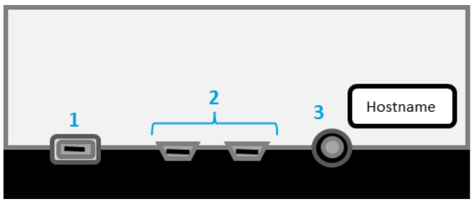


Figure 2.2: QPG7015M Development Kit front view

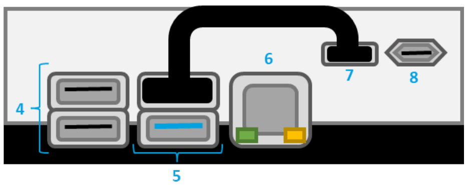


Figure 2.3: QPG7015M Development Kit right view

Access to the QPG7015M Development Kit terminal is required to configure, start and stop applications. There are three options to access the terminal of the QPG7015M Development Kit:

1. **Terminal over serial**: Connect the microUSB cable to port 8 shown in figure 2.3 and to your computer. Install putty for Windows or for Linux:

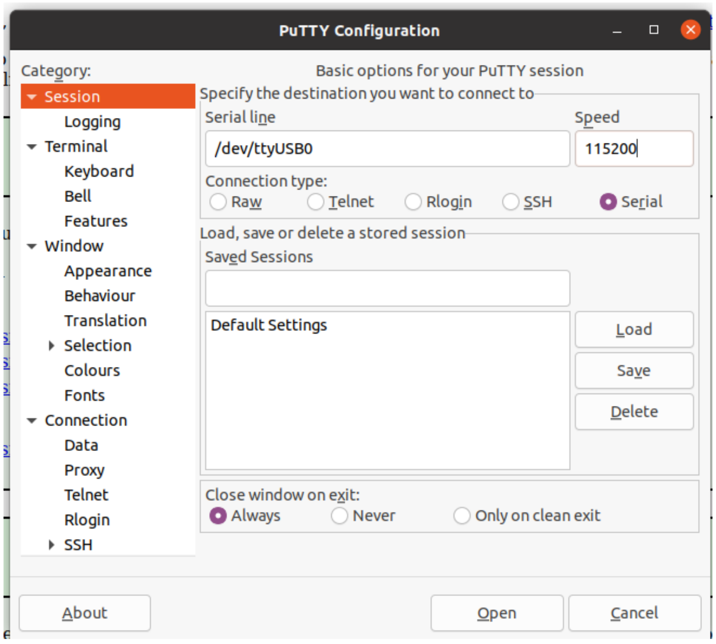
```
pi@[hostname]:~$ sudo apt-get install -y putty
```

   For Linux, open a terminal and find the port for the serial connection:

```
 pi@[hostname]:~$ dmesg
.
.
.
[111648.900661] usb 1-4.4:  FTDI USB Serial Device converter now attached to
ttyUSB0
```

   For Windows, press ⊞ + x and open the Device Manager. Open "Ports (COM& LPT)" and retrieve the serial USB COM port.

   Now open Putty and configure it as shown in picture 2.4. In the "Serial line", enter the serial port, e.g. COM6 or /dev/ttyUSB0. Next click Open. The username of the QPG7015M DK is *pi*, the password is *raspberry*. Please already typ the username followed by an ↲ when the serial window has opened. The terminal will ask for the password afterwards. The user now has access to the terminal



Specify the destination you want to connect to
Serial line: /dev/ttyUSB0
Speed: 115200
Connection type: Raw Telnet Rlogin SSH [✓ Serial]
Load, save or delete a stored session
Saved Sessions
Default Settings
Close window on exit: [✓ Always] Never Only on clean exit

Figure 2.4: Serial configuration on Putty

2. **Secure SHell (SSH)**: Connect the QPG7015M DK to your local area network using an ethernet cable on port 6, shown in figure 2.3 and apply power to port 1 shown in figure 2.2. From a computer that has an SSH client application installed the QPG7015M DK can be accessed over SSH by running from a command prompt (Windows) or terminal (Linux):

```
pi@[hostname]:~$ ssh pi@raspberrypi-<last 4 digits of the radio board its
serial number>
```

   The hostname can be found on the casing as depicted in figure 2.2. The username of the QPG7015M DK is *pi*, the password is *raspberry*. If for some reason the hostname cannot be accessed over ethernet, this is an option to retrieve the IP address running:

```
pi@[hostname]:~$ ifconfig
```

3. **Keyboard and screen**: Connect a USB keyboard to port 4 or 5 shown in figure 2.3 and attach a screen to a HDMI micro port 2 shown in figure 2.2. Please ensure the screen is connected before applying power to the QPG7015M DK. The user of the QPG7015M DK is *pi*, the password is *raspberry*.

! In general we recommend to always attach an ethernet cable to port 6, shown in figure 2.3.

2.1.3 Use the QPG7015M Development Kit

The terminal enables access to the QPG7015M DK, see subsection 2.1.2 how to achieve access to it.

To use the QPG7015M DK the user must

1. Use default configuration or configure the QPG7015M DK to enable the prefered functionality.
2. Start the QPG7015M DK, explained in section 2.1.3
3. Use the example applications, explained in section 2.2.

To terminate the QPG7015M DK the user can

- Stop the QPG7015M DK. See section 2.1.3

OR

- Reset the QPG7015M DK. See section 2.1.3

followed by

```
pi@[hostname]:~$ sudo halt
```

Wait for the green LED to stop blinking before removing the power.

Default configuration of the QPG7015M DK

The QPG7015M DK comes preinstalled with several communication stacks and example applications. These can each be enabled or disabled pending on the user's preference. Subsection 2.1.4 describes how to change the default configuration. By default the Development Kit is enabeled for following functionality:

Table 2.2: QPG7015M DK default configuration

Stack Configuration		
Name	**Configuration.**	**Info**
Zigbee	Disabled	Section 2.2.6
Thread CLI	Disabled	Section 2.2.7
Thread Border Router	Enabled	Section 2.2.5
Bluetooth LE	Enabled	Section 2.2.1
Performance Test Application	Disabled	Section 2.2.3
PTA Test Application	Disabled	Section 2.2.4
Software Configuration		
Name	**Configuration.**	**Info**
SPI	Enabled	Use SPI as Host⇔ Controller communication protocol. See subsection 2.1.4
USB	Disabled	Use USB as Host⇔ Controller communication protocol. See subsection 2.1.4
Debug	Disabled	Loads the debug version of the firmware with additional logging
DTM	Disabled	Enables the QPG7015M for Bluetooth LE Direct Test Mode. See subsection 2.2.2
Gateway at boot	Disabled	Starts the QPG7015M DK automatically when it is powered.

Start the QPG7015M DK

To start the QPG7015M DK, Qorvo provides a script called **start_gateway.sh**. After executing the user can get started with **the example application, explained in section 2.2**. When executing the start script, expect following logging (default configuration set):

```
pi@[hostname]:~$ ./start_gateway.sh
CONFIGURING GATEWAY SERVER...
Please don't stop the process, This can take a few minutes.
```

```
Run time configuration:
QORVO_CHIP=QPG7015M
QORVO_DRIVER_EXT=_RPi4
QORVO_ZB3=ZigBee3.0
QORVO_HOST_INTERFACE=SPI
QORVO_BLE=Bluetooth_LE
QORVO_FIRMWAREUPDATER=FirmwareUpdater
QORVO_FW_IMAGE=FirmwareUpdater/Firmware_QPG7015M.hex
QORVO_PTC=0
QORVO_CTC=0
QORVO_PTC_APP=Ptc
QORVO_CTC_APP=Ctc
QORVO_BOARDCONFIG_DIR=/home/pi/BoardConfigTool
QORVO_BOARDCONFIG_BIN=BoardConfigTool_RPi.elf
QORVO_DRIVERS=Drivers
QORVO_DRIVER_PATH=Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4
OSAL_DRIVER=OsalDriver_RPi4
OSALOEM_DRIVER=OsalOemDriver_QPG7015M_SPI_RPi4
APP_DRIVER=DrvComKernel_QPG7015M_SPI_RPi4
QORVO_ZIGBEE=0
QORVO_OT_CLI=0
QORVO_OT_BR=1
QORVO_OT_BRBB_INTERFACE=
QORVO_OPENTHREAD=OpenThread
OPENTHREAD_CLI=qpg7015m-ot-cli-ftd.elf
OPENTHREAD_RCP=qpg7015m-ot-rcp.elf
QORVO_COMDUMP=
QORVO_CONFIG_PATH=
QORVO_STARTUP_XML=
QORVO_SUDO=sudo
QORVO_DEBUG=0
QORVO_INTERFERENCE_THRESHOLD=192
QORVO_GW_AT_BOOT=0
QORVO_SOCKET_IPC=/dev/socket
QORVO_MAC_NVM_PATH=/etc/mac/macNvm.dat
APP_DRIVER_DEFAULT=DrvComKernel_QPG7015M_SPI_RPi4
STARTING GATEWAY SERVER...
PID TTY      STAT   TIME COMMAND
1034 ?        Ss     0:00 /lib/systemd/systemd -user
1035 ?        S      0:00 (sd-pam)
1617 tty1     S+     0:00 -bash
13427 ?        S      0:00 sshd: pi@pts/0
13428 pts/0    Ss     0:00 -bash
13670 pts/0    S+     0:00 /bin/sh ./start_gateway.sh
13673 pts/0    S+     0:00 /bin/sh ./qorvo_stack_init start
13755 pts/0    S+     0:00 /bin/sh ./qorvo_stack_init start
13756 pts/0    S+     0:00 sleep 1
13758 pts/0    R+     0:00 ps x
DrvComKernel_QPG7015M_SPI_RPi4 module is not loaded.
OsalOemDriver_QPG7015M_SPI_RPi4 module is not loaded.
OsalDriver_RPi4 module is not loaded.
```

```
Loading OsalDriver_RPi4 ...
Loading GreenPeak Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/ OsalDriver_RPi4.ko
for  (/dev/gp) kernel module...
No dev node created for module
Loading OsalOemDriver_QPG7015M_SPI_RPi4 ...
Loading GreenPeak Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/O
salOemDriver_QPG7015M_SPI_RPi4.ko for  (/dev/gp) kernel module...
No dev node created for module
Loading DrvComKernel_QPG7015M_SPI_RPi4 ...
Loading GreenPeak Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/
DrvComKernel_QPG7015M_SPI_RPi4.ko for  (/dev/gp) kernel module...
10:09:49:575 01 ============================
10:09:49:575 01 FirmwareUpdater
10:09:49:575 01 vX.X.X.X Change:XXXXXX
10:09:49:575 01 ============================
10:09:49:575 01 loading FirmwareUpdater/Firmware_QPG7015M.hex
10:09:49:724 01 INFO: Bootloader reports ready and valid
10:09:49:725 01 Bootloader is active
10:09:49:725 01 Bootloader Stage 2: vX.X.X.X Change:XXXXXX
10:09:49:725 01 Bootloader Stage1: version data unavailable (request not supported
by this stage2 bootloader)
10:09:49:725 01 ProductId: QPG7015
10:09:49:725 01 IEEE 15.4 MAC: 18:fc:26:00:00:4e:d7:16
10:09:49:726 01 BLE MAC: 18:fc:26:4e:d7:16
10:09:49:737 01 Verifying CRC...
10:09:50:341 01 image matches
10:09:50:341 01 Starting application...
10:09:55:369 01 Received cbDeviceReady from firmware
10:09:55:369 01 Firmware update SUCCEEDED vX.X.X.X Change:XXXXXX
Firmware update finished successfully (0) ...
Firmware STARTING ...
gateway not running - starting!
NOTICE: Enabling ConcurrentConnect mode
10:09:55:386 01 Settings applied
10:09:55:387 01 Enabling ConcurrentConnect mode
Loading ot-br docker image from file!
3a7cc06ad581: Loading layer [==================================================>]
48.07MB/48.07MB
57fa7918522d: Loading layer [==================================================>]
81.91MB/81.91MB
d626ab464e19: Loading layer [==================================================>]
197.7MB/197.7MB
Loaded image: connectedhomeip/otbr:teX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
* rsyslogd is running
* dbus is running
Avahi mDNS/DNS-SD Daemon is running
* otbr-agent is running
* otbr-web is running
Gateway does not start automatically at boot
```

```
----
WARNING! Enabling 802.15.4 (Zigbee or Thread) and Bluetooth LE scanning (central or
observer) requires additional configuration!!!
----
Combining 802.15.4 (Zigbee or Thread) and Bluetooth LE scanning (central or
observer) limits the 802.15.4 protocols to a single common channel.
See GP_P1053_UM_17952_Multi_Protocol_Configuration.pdf for more information.
If you are using only BLE peripheral functionality, the two 802.15.4 stacks can have
different channels if QORVO_BLUETOOTHLE_SCANNING=0 is added to qorvo_stack_config.
If you are combining Bluetooth LE scanning with 2 802.15.4 stacks, you may have to
update the individual stack channel configuration to fix the two 802.15.4 stacks to
a single channel.

You can retrieve the configured channel of a previously used stack this way:
* OpenThread: on ot-cli or ot-ctl issue the 'channel' command
* ZigBee: check the inventory>gateway XML node's channel attribute
in/etc/facilityd/inventory.xml (see GP_P332_AN_12712_Smart_Home_Gateway_XML_Files.pdf)
To change the channel the 802.15.4 stack is using, you can use:
* OpenThread: on ot-cli or ot-ctl issue the 'channel' command
* ZigBee: modify the channelmask in ZigBee3.0/start.xml and issue a factory reset
----
----

You can now start the BLE stack, by entering the following command:

cd Bluetooth_LE && ./Bluetooth_LE_Test_Qorvo_rpi.elf
----

Docker container otbr_eth is running and Up 22 seconds.

You can now start ot-ctl command-line utility, and get logs by entering the
following commands:

docker exec -it otbr_eth ot-ctl
docker logs otbr_eth [-follow]

----
GATEWAY STARTED AND READY TO USE!
```

Stop the QPG7015M DK

! The user should be sure `qorvo_stack_config` was not editted before stopping the gateway.

To stop the QPG7015M DK, Qorvo provides a script called **stop_gateway.sh**. Running this script will **NOT** delete the info of bonded or joined devices. Expect following logging (default configuration was set):

```
pi@[hostname]:~$ ./stop_gateway.sh
STOPPING GATEWAY...
Unloading DrvComKernel_QPG7015M_SPI_RPi4 ...
```

```
found Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/DrvComKernel_QPG7015M_SPI_RPi4.ko
Unloading GreenPeak 'Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/
DrvComKernel_QPG7015M_SPI_RPi4.ko' kernel module...
Unloading OsalOemDriver_QPG7015M_SPI_RPi4 ...
found Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/OsalOemDriver_QPG7015M_SPI_RPi4.ko
Unloading GreenPeak 'Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/
OsalOemDriver_QPG7015M_SPI_RPi4.ko' kernel module...
Unloading OsalDriver_RPi4 ...
found Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/OsalDriver_RPi4.ko
Unloading GreenPeak 'Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/OsalDriver_RPi4.ko'
kernel module...
```

Reset the QPG7015M DK

! The user should be sure `qorvo_stack_config` was not editted before resetting the gateway.

To stop and factory reset the QPG7015M DK, Qorvo provides a script called **factory_reset_gateway.sh**. Running this script **WILL** delete the info of bonded or joined devices. Expect following logging (default configuration was set):

```
pi@[hostname]:~$ ./factory_reset_gateway.sh
FACTORY RESETTING GATEWAY...
STOPPING GATEWAY...
Unloading DrvComKernel_QPG7015M_SPI_RPi4 ...
found Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/DrvComKernel_QPG7015M_SPI_RPi4.ko
Unloading GreenPeak 'Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/
DrvComKernel_QPG7015M_SPI_RPi4.ko' kernel module...
Unloading OsalOemDriver_QPG7015M_SPI_RPi4 ...
found Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/OsalOemDriver_QPG7015M_SPI_RPi4.ko
Unloading GreenPeak 'Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/
OsalOemDriver_QPG7015M_SPI_RPi4.ko' kernel module...
Unloading OsalDriver_RPi4 ...
found Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/OsalDriver_RPi4.ko
Unloading GreenPeak 'Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4/OsalDriver_RPi4.ko'
kernel module...
Total reclaimed space:  0B
Deleted Images:
untagged:  connectedhomeip/otbr:teX
deleted:  sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
deleted:  sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
deleted:  sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
deleted:  sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Total reclaimed space:  XXX.XMB
GATEWAY IS FACTORY RESET!
```

2.1.4 Configure the QPG7015M Development Kit

The QPG7015M DK comes preinstalled with several communication stacks and example applications. These can each be enabled or disabled pending on the user preference using a configuration file: **qorvo_stack_config**.
The configuration file is located in the home director: `/home/pi`.
From a terminal, the user can modify the configuration file by uncommenting a setting and changing its value. To edit the file run:

```
pi@[hostname]:~$ sudo nano qorvo_stack_config
```

To save the file run Ctrl+o followed by Ctrl+x.
The file exists out of several sections:

- Communication Interface Configuration
- Application location on the file system
- Stack configuration
- Software configuration

Communication Interface Configuration

The QPG7015M supports two interface protocols: **USB** and **SPI**. For more detailed info see section 3.4. By default SPI is configured. Changing the interface protocol is done by modifying the jumper configuration on the QPG7015M DK AND modifying the qorvo_stack_config file.

Access jumper configuration

To open the QPG7015M DK, remove the 4 screws on the bottom of the QPG7015M DK with a cross tip screwdriver as shown in picture 2.5. No need to remove the nut. Afterwards the top of the plastic casing can be removed.

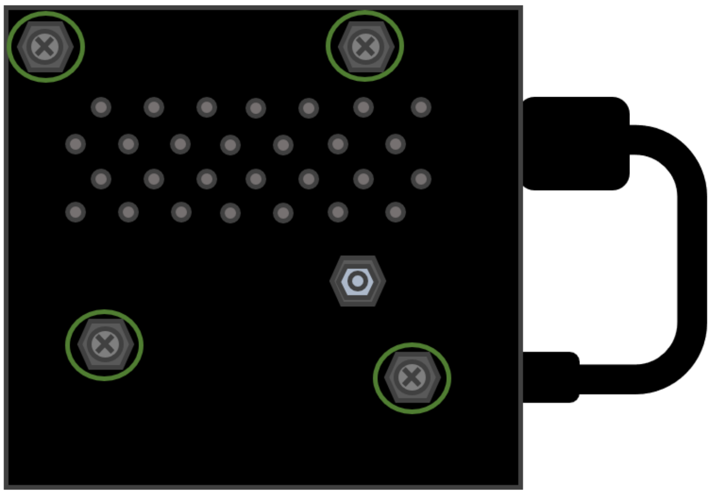


Figure 2.5: QPG7015M DK bottom

The jumpers CFG1 and CFG2 can be found by opening the case on the QPG7015M Radio Board. They're shown in figure 2.6

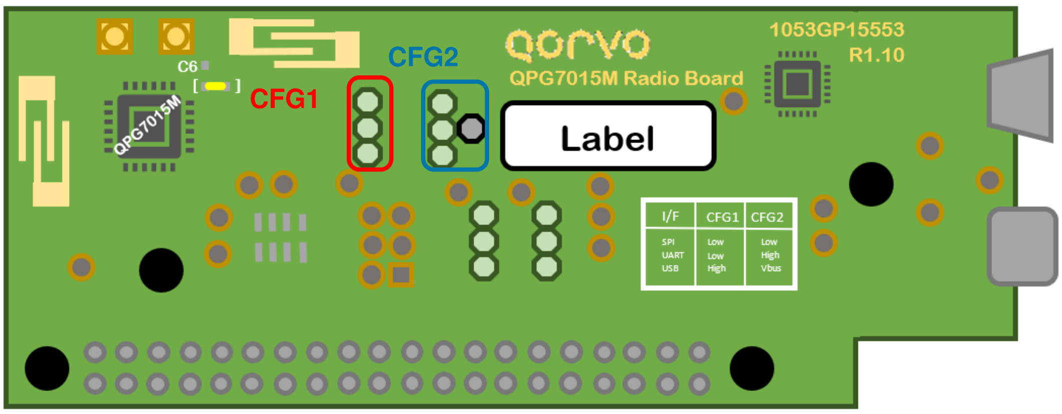
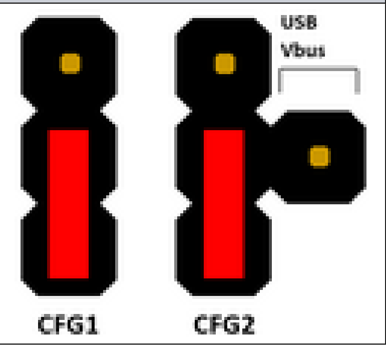
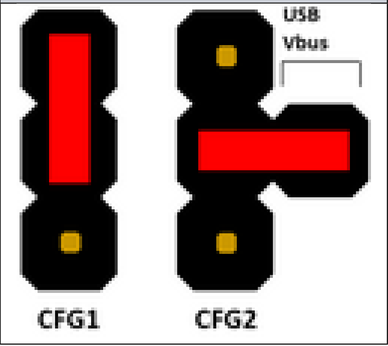


Figure 2.6: QPG7015M interface configuration

Possible jumper configurations are shown in table 2.3. For USB a USB A ⇔ USB C cable shoud be connected from port 5 to port 7 shown in figure 2.3.

Table 2.3: QPG7015M DK default configuration

SPI	USB
	

Edit configuration file

The interface can be configured by uncommenting `QORVO_HOST_INTERFACE` and set `"SPI"` or `"USB"`. E.g. for USB configuration:

```
# HARDWARE CONFIGURATION
#---

# Qorvo chip
- ~~QORVO_CHIP="QPG7015M"~~
+ QORVO_CHIP="GP712"

# Qorvo host interface selection
#                                        "USB"
#                                        "SPI" (default)
- ~~# QORVO_HOST_INTERFACE="SPI"~~
+ QORVO_HOST_INTERFACE="USB"
```

Application location on the file system

These parameters are used to give input to the Qorvo startup, stop and reset scripts about the path of the application executables and kernel drivers on the file system.

! These locations should **NOT** be changed when using the QPG7015M DK, but can be modified when building a custom filesystem when porting the QPG7015M DK SW to a custom platform.

Configureable locations in `qorvo_stack_config`

```
# LOCATIONS
#---

# ZigBee 3.0 stack location: "ZigBee3.0" (default)
#                            "ZigBee3.0_OSHS"
# QORVO_ZB3="ZigBee3.0"

# OpenThread stack location:           "OpenThread" (default)
# QORVO_OPENTHREAD="OpenThread"

# Ble stack location:                   "Bluetooth_LE" (default)
# QORVO_BLE="Bluetooth_LE"

# FirmwareUpdater location:            "FirmwareUpdater" (default)
# QORVO_FIRMWAREUPDATER="FirmwareUpdater"

# Firmware image location:      "${QORVO_FIRMWAREUPDATER}/Firmware_${QORVO_CHIP}
${QORVO_DEBUG_POSTFIX}.hex" (default)
# QORVO_FW_IMAGE="FirmwareUpdater/Firmware_QPG7015M.hex"

# Qorvo drivers location:               "Drivers" (default)
# QORVO_DRIVERS="Drivers"

# Qorvo Driver Path string:             "${QORVO_DRIVERS}/${RUNTIME_PI_MODEL}/
${RUNTIME_KERNEL_VERSION}/${QORVO_CHIP}${QORVO_DRIVER_EXT}" (default)
# QORVO_DRIVER_PATH="Drivers/RPi4/5.10.17-v7l+-qorvo/QPG7015M_RPi4"

# Qorvo socket IPC location:            "/dev/socket" (default)
# QORVO_SOCKET_IPC="/dev/socket"

# PTC location:                   "Ptc" (default)
# QORVO_PTC_APP="Ptc"

# CTC location:                   "Ctc" (default)
# QORVO_CTC_APP="Ctc"

# Override Osal Driver kernel module name:     "OsalDriver${QORVO_DRIVER_EXT}"
(default)
# OSAL_DRIVER=""

# Override OsalOem Driver kernel module name:  "OsalOemDriver_${QORVO_CHIP}
${QORVO_FEM} ${QORVO_DRIVER_EXT}" (default)
# OSALOEM_DRIVER=""

# Override App Driver kernel module name:      "DrvComKernel_${QORVO_CHIP}${QORVO_FEM}
${QORVO_DRIVER_EXT}" (default)
# APP_DRIVER=""
```

Stack configuration

The stack configuration section is used to configure which communication stacks will be loaded at start up. Combinations of these protocols are possible. Since the OpenThread Border Router contains the OT-CLI functionality, both can't be configured at the same moment. By selecting the required protocols, the script will also configure the optimal listening mode. For more info see 3.9.

! These are the main configurations that should be set by the user before running the `start_gateway.sh` script!

Following stacks can be configured:

- QORVO_ZIGBEE
  - Verify our **certified** Zigbee 3.0 R22 functionality.
  - Enables interacting with the Zigbee 3.0 SmartHome Gateway Client Application.
- QORVO_OT_CLI
  - Enables interacting with the OpenThread Command Line Interface.
  - Verify our Thread 1.3 functionality.
- QORVO_OT_BR
  - Enables interacting with the OpenThread Border Router.
  - Set up a Matter network using our OpenThread Border Router.
- QORVO_OT_BRBB_INTERFACE
  - Configures the OpenThread Border Router network access.
  - Connect to the OT-BR over a local WiFi network or via a wired ethernet connection to internet.
- QORVO_BLUETOOTHLE
  - Our QPG7015M is **certified** for Bluetooth LE 5.2
  - Enables interacting with the Bluetooth LE ACS deliverable.
  - Supports all Bluetooth LE GAP and GATT roles.
- QORVO_BLUETOOTHLE_SCANNING
  - Configure this when scanning is required within Bluetooth LE.
  - Enabling Bluetooth LE scanning, e.g. Central use case or is scanning for beacons, while at least, one IEEE802.15.4 protocol is enabled will enable our ConcurrentConnect™ functionality. ConcurrentConnect™ enables concurrently listening to a Bluetooth LE advertising channel (i.e. for scanning) and an IEEE802.15.4 channel (i.e. for Zigbee or Thread).
- QORVO_PTC
  - Configure this to enable production test application.
  - Test the QPG7015M radio performance and functionality.
- QORVO_CTC
  - Configure this to enable coexistence verification.
  - Test the QPG7015M PTA capability.

Configureable stacks in `qorvo_stack_config`

```
# STACK CONFIGURATION
#---

# Enable ZigBee 3.0 stack:                  0 (default)
#                                           1
# QORVO_ZIGBEE=0

# Enable Thread stack (CLI):                0 (default)
#                                           1
# QORVO_OT_CLI=0

# Enable OpenThread Border Router:          0
#                                           1 (default)
# QORVO_OT_BR=1

# OpenThread Border Router Backbone Interface
#                                          eth0 (default)
#                                          wlan0
# QORVO_OT_BRBB_INTERFACE=eth0

# Enable Bluetooth LE (Host) stack:         0
#                                           1 (default)
# QORVO_BLUETOOTHLE=1

# When QORVO_BLUETOOTHLE and one of the 802.15.4 stacks (QORVO_ZIGBEE or
QORVO_OT_CLI or QORVO_OT_BR) is enabled, the QPG7015M will be configured in
# ConcurrentConnect mode. In ConcurrentConnect mode, the radio will be able to
concurrently do BLE scanning and listen to a single
# 802.15.4 channel.
# If no BLE scanning is required, disable the configuration below to configure the
QPG7015M in multi-channel mode, allowing the Thread
# and Zigbee stack to operate on separate channels.
See GP_P1053_UM_17952_Multi_Protocol_Configuration.pdf for more information.
#                                           0
#                                           1 (default)
# QORVO_BLUETOOTHLE_SCANNING=1

# Enable production test application        0 (default)
#                                           1
# QORVO_PTC=0

# Enable Coex test application              0 (default)
#                                           1
# QORVO_CTC=0
```

Software configuration

This section enables additional configuration for the software. The most interesting ones are highlighted.

- QORVO_CONFIG_PATH
  - Indicates where the Zigbee coordinator configuration file will be stored
- QORVO_STARTUP_XML
  - Defines the name of the Zigbee coordinator configuration file
- QORVO_DEBUG
  - This parameter shows additional logging of the firmware and the kernel drivers in the kernel logging accessible via `dmesg`.
- QORVO_CRASHREPORT
  - This parameter generates a report when the QPG7015M Gateway DK crashes. These reports are stored at `/home/pi`.
- QORVO_COMDUMP
  - This parameter captures all traffic between the host and controller and stores it in `~/Logs`. The required application is not by default present on the QPG7015M Gateway DK and must be build from its source package (see chapter 3). After building ComDumpTool_RPi.elf make sure it is in the correct folder:

```
pi@[hostname]:~$ mkdir ComDumpTool
pi@[hostname]:~$ mv ComDumpTool_RPi.elf ComDumpTool/
```

! ComDumpTool has an impact on the performance and should be used only during development for debug purposes.

- QORVO_DTM
  - These parameters enables Bluetooth LE Direct Test Mode. This mode is used for Bluetooth LE certifications and cannot be combined with other active stacks. For more info see subsection 2.2.2
- QORVO_INTERFERENCE_THRESHOLD="192"
  - This parameter configures the energy detect threshold used by the Zigbee stack to pick the correct channel. Its range is from 0 to 255 and is linear with the measures power. (0: -95 dBm, 254: -45 dBm, 255: threshold disabled). The recommended setting is 192. For more info see subsection 2.2.6

! Do not change this unless the gateway must be specifically tuned for noisy environments.

- QORVO_GW_AT_BOOT
  - This setting will automatically start the QPG7015M drivers and applications upon boot. The startup behavior described in subsection 2.1.3 will be automatically triggered.

Configureable software in `qorvo_stack_config`

```
# SOFTWARE CONFIGURATION
#---

# Qorvo Configuration File Path string:  "" (default) = /etc
# QORVO_CONFIG_PATH=""

# Qorvo Startup XML File string:        "" (default) = "start.xml"
# QORVO_STARTUP_XML=""

# Qorvo SUDO string:                    "sudo" (default)
#                                       "NONE" no sudo
# QORVO_SUDO="sudo"

# Enable debugging options:             0 (default)
#                                       1
# QORVO_DEBUG=0

# Enable crash dumps:                   1 (default)
#                                       0
# QORVO_CRASHREPORT=1


# Enable Direct Test Mode               0 (default)
#                                       1
# QORVO_DTM=0

# Configurable interference threshold for channel selection.
# The gateway scans RF energy levels at startup to select a channel.
# Channels with energy level above this threshold will not be considered.
#
# Range 0 to 255 (0: -95 dBm, 254: -45 dBm, 255: threshold disabled).
# The recommended setting is 192.
# Do not change this unless the gateway must be specifically tuned for noisy
environments.
# QORVO_INTERFERENCE_THRESHOLD="192"

# Start Gateway at boot:                0 (default)
#                                       1
# QORVO_GW_AT_BOOT=0

# Dump COM packets in file              0 (default)
#                                       1
# QORVO_COMDUMP=0
```

2.2 Using the example applications

This section covers how to get started with the example applications. An architectural overview of the QPG7015M DK together with how to build and port these applications can be found in chapter 3.

2.2.1 Bluetooth LE

The Bluetooth LE deliverable is built upon the Amazon Common Software Device Porting Kit, (read here for more info) and exposes the Bluetooth LE functionality through a command line interface for easy testing and evaluation. Customers developing Bluetooth LE applications can use this reference application as an example implementation how to access and use the ACS Bluetooth LE DPK API.

The Test application covers is compliant for several roles:

- Generic Access Profile
  - Broadcaster (Beacons)
  - Peripheral
  - Central
- Generic Attribute Profile
  - Client
  - Server

! Prior to running the Bluetooth LE test application, two action items:

1. Configure Bluetooth LE to be enabled in `qorvo_stack_config`: `QORVO_BLUETOOTHLE=1` and also for the Central use case `QORVO_BLUETOOTHLE_SCANNING=1` (See subsection 2.1.4.)
2. The QPG7015M DK should already be started as described in subsection 2.1.3.

Install Qorvo Connect on smartphone

These examples are to be tested against a Bluetooth LE application, created by Qorvo. **Qorvo Connect** can be installed in the Google Play or in the Apple Appstore:

Figure 2.7: Install codes Qorvo Connect

Broadcaster (Beacons) Example

This example explains how to send beacons with an manufacturing specific data e.g. "01020304".

1. Start the Bluetooth LE test application

```
pi@[hostname]:~$ cd Bluetooth_LE && ./Bluetooth_LE_Test_Qorvo_rpi.elf
```

2. Get the Bluetooth LE address of the device

```
getdevprop 2
12:53:37:687 1C acsBleHalBt_GetDeviceProperty
Get Device Property, status:  0
>BTDevicePropertiesCb:
Number of Properties:  1
Properties ~ Bluetooth Device Address:  :  (6) 18 fc 26 4e d7 16
Status:  0x00
```

3. Configure the advertising properties to be NON CONNECTABLE and to have an unlimited duration:

```
setadvdata_prop 0 0 2 0 0
```

4. Configure the advertising data

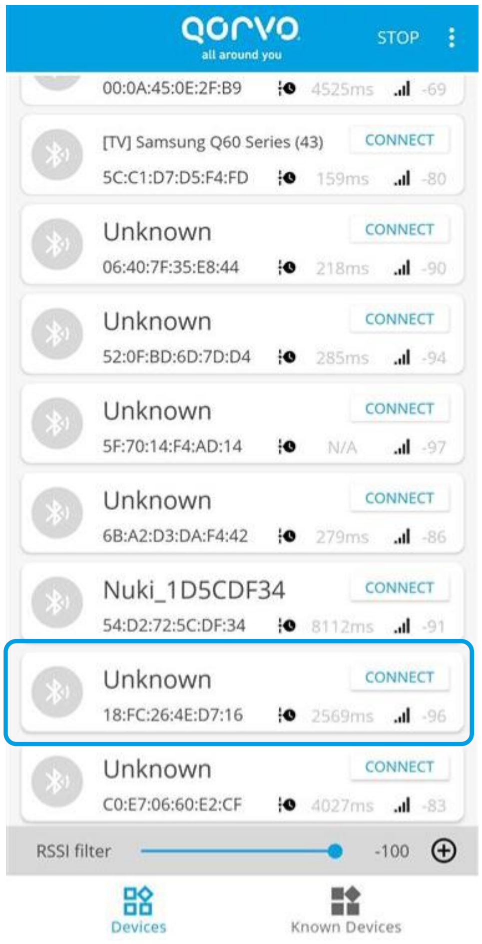
```
setadvrawdata 0 11 02010607FF530401020304
```

- 02 ⇒ Combined size of the advertising data type and its content
- 01 ⇒ Advertising data type [Flags]
- 06 ⇒ Flags [BR/EDR not Supported (b0100) + LE General Discoverable Mode (b0010)]
- 07 ⇒ Combined size of the advertising data type and its content
- FF ⇒ Advertising data type [Manufacturer Specific Data]
- 5304 ⇒ Manufacturer ID [Qorvo Utrecht B.V.]
- 01020304 ⇒ Manufacturer Specific Data [01020304]

5. Start advertising

```
adv 0 1
```

6. Open Qorvo Connect on your smartphone and look for the Bluetooth LE address of your device. See figure 2.8.

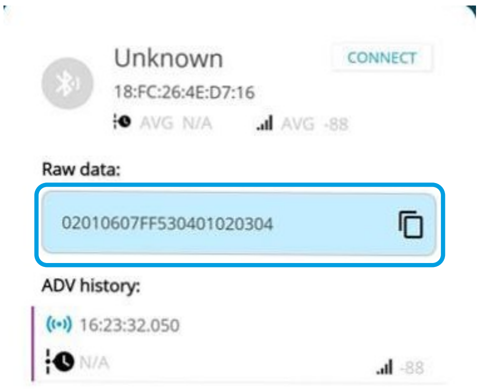


00:0A:45:0E:2F:B9	4525ms	-69	
[TV] Samsung Q60 Series (43) 5C:C1:D7:D5:F4:FD	159ms	-80	CONNECT
Unknown 06:40:7F:35:E8:44	218ms	-90	CONNECT
Unknown 52:0F:BD:6D:7D:D4	285ms	-94	CONNECT
Unknown 5F:70:14:F4:AD:14	N/A	-97	CONNECT
Unknown 6B:A2:D3:DA:F4:42	279ms	-86	CONNECT
Nuki_1D5CDF34 54:D2:72:5C:DF:34	8112ms	-91	CONNECT
Unknown 18:FC:26:4E:D7:16	2569ms	-96	CONNECT
Unknown C0:E7:06:60:E2:CF	4027ms	-83	CONNECT

RSSI filter -100

Figure 2.8: Find beacon device in Qorvo Connect

7. Tap on the device with the matching address and verify the advertising data. See figure 2.9.



Unknown
18:FC:26:4E:D7:16
AVG N/A AVG -88

Raw data:

02010607FF530401020304

ADV history:

16:23:32.050
N/A -88

Figure 2.9: See Beacon data in Qorvo Connect

8. Terminate the application via Ctrl + c

Peripheral Example

This example explains how to act as a peripheral with a custom readable and writeable service.

1. Start the Bluetooth LE test application

```
pi@[hostname]:~$ cd Bluetooth_LE && ./Bluetooth_LE_Test_Qorvo_rpi.elf
```

2. Define your own service
   (a) Generate a 128-bit UUID using a generator.
   E.g. DD017130-FDED-11EC-B939-0242AC120002 .
   (b) This can be reduced to the following base-UUID 0000XXXX-FDED-11EC-B939-0242AC120002
   (c) Replace XXXX with your chosen 16-bit service UUID.
   E.g. for service UUID 0001: 00000001-FDED-11EC-B939-0242AC120002
3. Add that service

```
addservicegs 2 0 0 00000001fded11ecb9390242ac120002 3
```

4. Add a characteristic that's writeable and readable to that service. E.g. for characteristic UUID 0002: 00000002-FDED-11EC-B939-0242AC120002

```
addchargs 2 26 00000002fded11ecb9390242ac120002 10 17
```

5. Enable that service

```
startservicegs 2 26 2
```

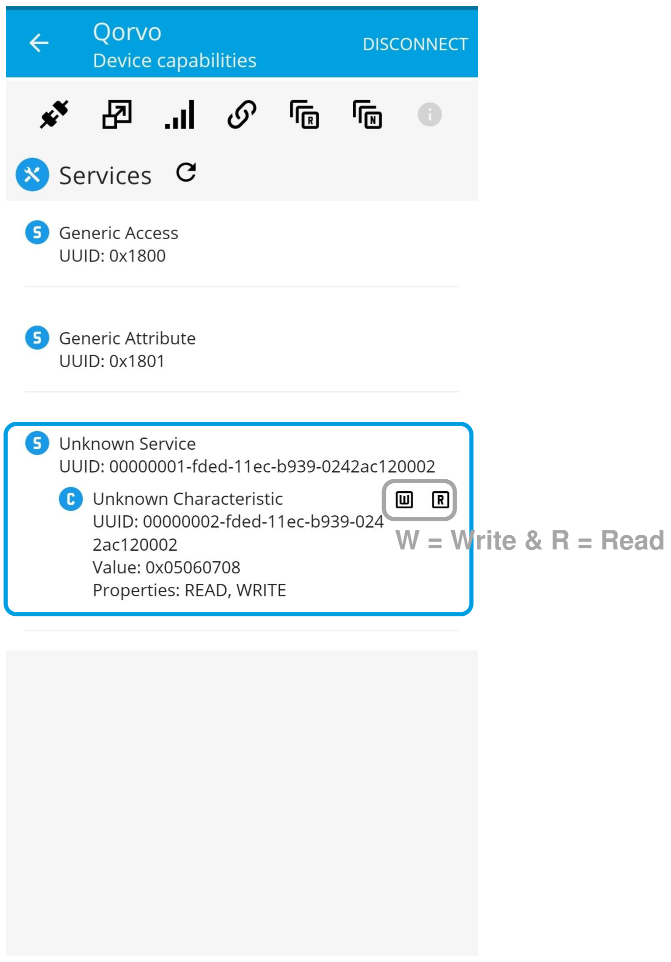
6. Configure the advertising to have an unlimited duration

```
setadvdata_prop 0 0 0 0 0
```

7. Start advertising

```
adv 0 1
```

8. Open the Qorvo Connect app, look for a device named Amazon Qorvo and press connect.
9. Open the connection tab and verify the service with a writeable and readable characteristic is present



Generic Access
UUID: 0x1800

Generic Attribute
UUID: 0x1801

Unknown Service
UUID: 00000001-fded-11ec-b939-0242ac120002

Unknown Characteristic
UUID: 00000002-fded-11ec-b939-024
2ac120002
Value: 0x05060708
Properties: READ, WRITE

Figure 2.10: Find custom service in Qorvo Connect

10. Write a value to the characteristic via your smartphone. e.g. 01020304.

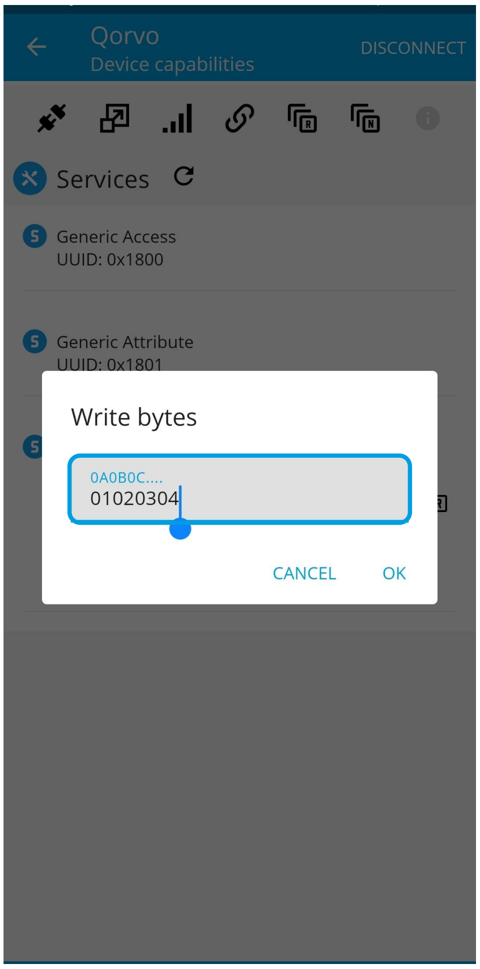


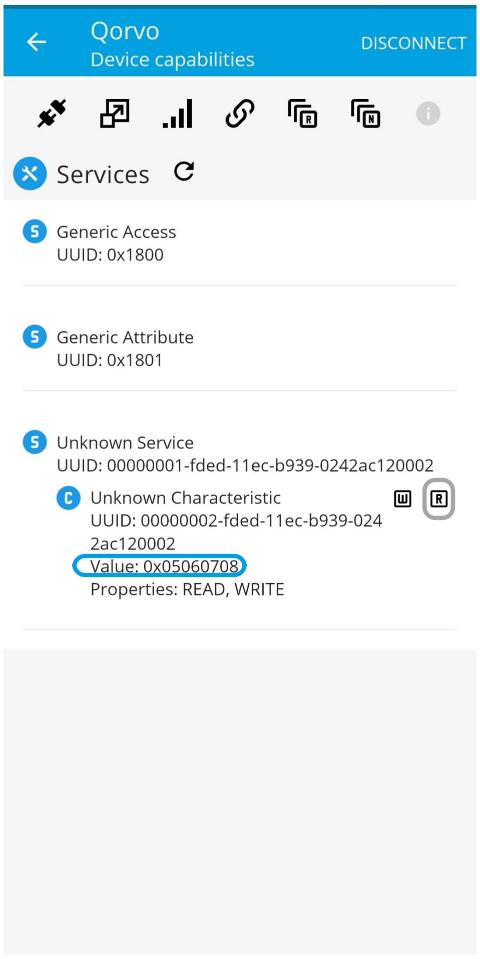
Figure 2.11: Write custom characteristic in Qorvo Connect

```
pxRequestWriteCb:
ConnId:  1
TransId:  0
BLE address (Little Endian):  :  (6) 78 90 e5 39 51 da
AttrHandle:  28
Offset:  0
Length:  4
NeedResponse:  true
IsPrepare:  false
Value (Little Endian):  :  (4) 01 02 03 04
15:47:43:795 1C acsBleHalGS_setVal
pxSetValCallbackCb:
Status:  0x00
usAttrHandle:  28
15:47:43:796 1C acsBleHalGS_sendResponse
pxResponseConfirmationCb:
Status:  0x00
usHandle:  28
```

11. Write a value to the characteristic using your peripheral. e.g. 05060708.

```
setvalgs 28 0 05060708
15:50:14:994 1C acsBleHalGS_setVal
Set Value request, status:  0
>pxSetValCallbackCb:
Status:  0x00
usAttrHandle:  28
```

12. Verify you can read the characteristic using your phone.



Generic Access
UUID: 0x1800

Generic Attribute
UUID: 0x1801

Unknown Service
UUID: 00000001-fded-11ec-b939-0242ac120002

Unknown Characteristic
UUID: 00000002-fded-11ec-b939-0242ac120002
Value: 0x05060708
Properties: READ, WRITE

Figure 2.12: Read custom characteristic in Qorvo Connect

```
pxRequestReadCb:
ConnId:  1
TransId:  4
BLE address (Little Endian):  :  (6) 78 90 e5 39 51 da
AttrHandle:  28
Offset:  0
```

```
15:50:25:465 1C acsBleHalQorvo_getVal
15:50:25:466 1C acsBleHalGS_sendResponse
pxResponseConfirmationCb:
Status:  0x00
usHandle:  28
```

13. Terminate the application via Ctrl + c

Central Example

This example explains how to start scanning and write to a service. It was tested against a second QPG7015M DK, running the peripheral example. It is expected to work with every Bluetooth LE device with a readable and writeable service with no encryption requirement.

1. Start the Bluetooth LE test application

```
pi@[hostname]:~$ cd Bluetooth_LE && ./Bluetooth_LE_Test_Qorvo_rpi.elf
```

2. Start scanning and find your peripheral

```
scan 1
pxScanResultCb:
Peer address:  16d74e26fc18
RSSI: -49
adv data:   :  (62) 02 01 06 02 01 06 06 08 51 6f 72 76 6f 00 00 8c 48 00 78 00
4d 00 00 00 f8 0d 40 b6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

3. Connect to the device

```
connectgc 1 16d74e26fc18 1 2
```

4. Discover all services on the device

```
searchservicegc 1 0
```

5. Show all discovered services

```
getgattdbgc 1
07:20:01:668 1C acsBleHalGC_getGattDb
Client Get GATT Database Request, status:0

>pxGetGattDbCb:
Client Database:
########################################################
#                    Primary Service                   #
########################################################

Start Handle: 0x0001
End Handle:   0x0007
```

```
Id:          0
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x1800
Type:        0
Attribute Handle ~ : 0x0001
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                    Characteristic                     ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   2
Id:          1
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2a00
Type:        4
Attribute Handle ~ : 0x0003
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                    Characteristic                     ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   2
Id:          2
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2a01
Type:        4
Attribute Handle ~ : 0x0005
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                    Characteristic                     ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   2
Id:          3
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2aa6
Type:        4
Attribute Handle ~ : 0x0007
#######################################################
#                    Primary Service                  #
#######################################################

Start Handle: 0x0010
End Handle:   0x0019
Id:          4
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x1801
Type:        0
Attribute Handle ~ : 0x0010
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                    Characteristic                     ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   32
```

```
Id:          5
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2a05
Type:        4
Attribute Handle ~ : 0x0012
.......................................................
.                     Descriptor                      .
.......................................................

Id:          6
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2902
Type:        5
Attribute Handle ~ : 0x0013
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                    Characteristic                    ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   10
Id:          7
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2b29
Type:        4
Attribute Handle ~ : 0x0015
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                    Characteristic                    ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   2
Id:          8
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2b2a
Type:        4
Attribute Handle ~ : 0x0017
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                    Characteristic                    ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   2
Id:          9
Uuid ~ Type: 16 bit
Uuid ~ Uuid (Big Endian): 0x2b3a
Type:        4
Attribute Handle ~ : 0x0019
#######################################################
#                   Primary Service                   #
#######################################################

Start Handle: 0x001a
End Handle:   0xffff
Id:          10
```

```
Uuid ~ Type: 128 bit
Uuid ~ Uuid (Little Endian) : 0x : (16) 02 00 12 ac 42 02 39 b9 ec 11 ed fd 01
00 00 00
Type:        0
Attribute Handle ~ : 0x001a
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~                   Characteristic                   ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Properties:   10
Id:          11
Uuid ~ Type: 128 bit
Uuid ~ Uuid (Little Endian) : 0x : (16) 02 00 12 ac 42 02 39 b9 ec 11 ed fd 02
00 00 00
Type:        4
Attribute Handle ~ : 0x001c
Count: 12
```

6. Write data to the readable/writable characteristic

```
writechargc 1 0x001c 2 4 0 01020304
```

7. Verify the written data by reading data to from that readable/writable characteristic

```
readchargc 1 0x001c 0
07:48:51:875 1C acsBleHalGC_readCharacteristic
Client Read Characteristic Request, status:0

>pxReadCharacteristicCb:
ConnId: 0x0001
Status: 0x00
Data ~ Handle: 0x001c
Data ~ Value:
0x01
0x02
0x03
0x04

Data ~ Value length: 0x0004
Data ~ Value Type: 0x0002
Data ~ Status: 0x00
```

8. Disconnect from device

```
disconnectgc 1 16d74e26fc18 1
>Client - pxCloseCb:
ConnId:  0x0001
Status:  0x00
ucClientIf:  1
Peer address (Little Endian):  :  (6) 18 fc 26 4e d7 16
```

9. Terminate the application via Ctrl + c

Additional documentation

Additional documentation can be found within GP_P1053_SW_17777_Bluetooth_LE_ACS_DPK.zip. For retrieval, see section 3.2.

- GP_P1053_RN_17778_Bluetooth_LE_ACS_DPK.pdf
  ⇒ Release notes
- GP_P1053_UM_17780_Quick_Start_Guide_Bluetooth_LE_ACS_DPK.pdf
  ⇒ Additional Quickstart examples
- GP_P1053_DD_17037_Architecture_BleAdapter.pdf
  ⇒ Architectural overview of the DPK implementation
- GP_P1053_AN_18056_Bluetooth_LE_Test_with_PTA.pdf
  ⇒ Application Note describing the BleTest application and the API functionality
- GP_P1053_AN_17824_Bluetooth_LE_ACS_API_List.pdf
  ⇒ Application Note with a list of all covered ACS API in our DPK deliverable.
- Additional info on the API of the Bluetooth LE Host Stack layer used below the ACS DPK API
  - att-api.pdf
    ⇒ Attribute protocol client and server API
  - dm-api.pdf
    ⇒ Device Manager subsystem API
  - hci-api.pdf
    ⇒ Host Controller Interface subsystem API
  - l2c-api.pdf
    ⇒ Logical Link Control and Adaptation Protocol subsystem API
  - smp-api.pdf
    ⇒ Secure Manager Protocol subsystem API
  - wsf.pdf
    ⇒ Wireless Software Foundation

2.2.2 Bluetooth LE Direct Test Mode

For Bluetooth LE Qualification purposes, the QPG7015M can be instructed to enter a UART-based HCI Bluetooth Direct Test Mode (DTM) via the host interface. The DTM and enables testing the radio operation at the PHY level with certified Bluetooth testers.

Radiated and conducted measurements

The user has the possibility to measure radiated or conducted:

1. Radiated [default] ⇒ Enabled when capacitor C6 is placed horizontal (-)
2. Conducted ⇒ Enabled when capacitor C6 is placed vertical (|)

See figure 2.13 to find the antennas and capacitor C6 on the board.

QPG7015M configuration

To enable the QPG7015M to boot up in direct test mode instead of normal operating mode, the user must reconfigure the qorvo_stack_config, after a stop or reset of the gateway (see section 2.1.3).

- QORVO_ZIGBEE=0
- QORVO_OT_CLI=0
- QORVO_OT_BR=0
- QORVO_BLUETOOTHLE=0
- QORVO_BLUETOOTHLE_SCANNING=0
- QORVO_PTC=0
- QORVO_CTC=0
- QORVO_DTM=1

Now run the start gateway script again (see section 2.1.3).

Following characteristics apply to the UART-based DTM interface:

- Baud rate: 57600Bd
- 8 data bits, 1 stop bit, no parity
- No hardware flow control

Following QPG7015M pins are assigned for Bluetooth LE Direct Test Mode, and shown in figure 2.13

1. Pin 11 (SWV / TDO / UART2_RX): UART_RX for Bluetooth Direct Test Mode
2. Pin 12 (TDI / UART2_TX): UART_TX for Bluetooth Direct Test Mode
3. Antenna 1, since DTM runs on antenna 1 only.

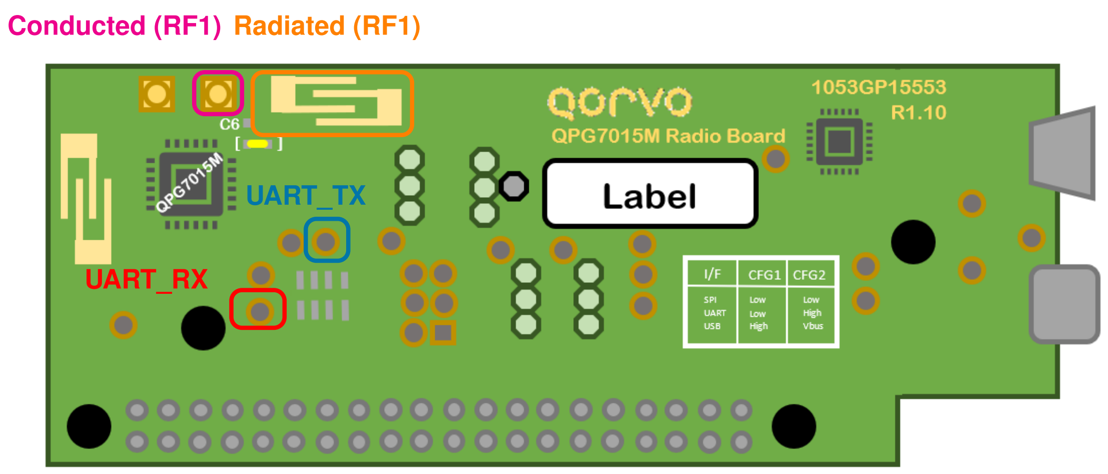


Figure 2.13: DTM Test points

For additional information on DTM, see the Bluetooth Specification 5.3, Volume 6, Part F.

2.2.3 RF Evaluation (PTC)

Radiated and conducted measurements

The user has the possibility to measure radiated or conducted. See subsection 2.2.2.

QPG7015M configuration

To enable PTC on QPG7015M to boot up in direct test mode instead of normal operating mode, the user must reconfigure the qorvo_stack_config, after a stop or reset of the gateway (see section 2.1.3).

- QORVO_ZIGBEE=0
- QORVO_OT_CLI=0
- QORVO_OT_BR=0
- QORVO_BLUETOOTHLE=0
- QORVO_BLUETOOTHLE_SCANNING=0
- QORVO_PTC=0
- QORVO_CTC=0
- QORVO_DTM=1

Now run the start gateway script again (see section 2.1.3).

2.2.4 PTA Evaluation (CTC)

2.2.5 Matter

2.2.6 Zigbee

Command Line Interface

Smartphone Control

2.2.7 Thread

Command Line Interface

3. Porting the example application to a custom application processor

3.1 Software architecture overview

```
pi@[hostname]:~$ tree
.
+- 99-qpg7015m.rules
+- Bluetooth_LE
|   +- Bluetooth_LE_Test_Qorvo_rpi.elf
+- BoardConfigTool
|   +- BoardConfigTool_RPi.elf
+- BUILDINFO
+- CoexConfigTool
|   +- CoexConfigTool_QPG7015M_RPi.elf
+- crash_report.sh
+- Ctc
|   +- CTC_QPG7015M_RPi.elf
+- docker
|   +- otbr_RPi4.docker
+- Drivers
|   +- load
|   +- RPi4
|   |   +- 5.10.17-v7l+-qorvo
|   |       +- QPG7015M_RPi4
|   |           +- DrvComKernel_QPG7015M_SPI_RPi4.ko
|   |           +- DrvComKernel_QPG7015M_USB_RPi4.ko
|   |           +- OsalDriver_RPi4.ko
|   |           +- OsalOemDriver_QPG7015M_SPI_RPi4.ko
|   |           +- OsalOemDriver_QPG7015M_USB_RPi4.ko
|   |           +- qorvo.service
|   +- TestCom.elf
+- factory_reset_gateway.sh
+- FirmwareUpdater
|   +- Firmware_QPG7015M-dbg.hex
|   +- Firmware_QPG7015M.hex
|   +- FirmwareUpdater_RPi.elf
+- HostnameGen
|   +- HostnameGen_RPi.elf
+- OpenThread
```

```
|   +- qpg7015m-ot-cli-ftd.elf
|   +- qpg7015m-ot-rcp.elf
+- Ptc
|   +- PTC_QPG7015M_RPi.elf
+- qorvo_stack_config
+- qorvo_stack_init
+- start_gateway.sh
+- stop_gateway.sh
+- ZigBee3.0
+- c7b.fb.bin
+- c7b.psb.bin
+- c7b.psb.factory.bin
+- etc
|   +- facilityd
|       +- c7b.psb.bin
|       +- c7b.psb.factory.bin
|       +- settings_ubi.xml
|       +- settings.xml
+- facilityd
+- facilityd-dbg
+- facility-manage
+- facility-manage-dbg
+- lib
|   +- libev.so -> libev.so.4.0.0
|   +- libev.so.4 -> libev.so.4.0.0
|   +- libev.so.4.0.0
|   +- libubisyssysdeps.so
|   +- libuuid.so -> libuuid.so.1.0.0
|   +- libuuid.so.1 -> libuuid.so.1.0.0
|   +- libuuid.so.1.0.0
+- otad
+- otad-dbg
+- SmartHomeGatewayClient_RPi-dbg.elf
+- SmartHomeGatewayClient_RPi.elf
+- start.xml
+- zcpd
+- zgdd
+- zgdd-dbg
+- zgsd
+- zgsd-dbg
+- zstartup
+- zstartup-dbg
```

3.2 How to get access to the latest software and documentation

3.3 Scripting

3.4 Kernel driver

3.4.1 USB

3.4.2 SPI

3.4.3 Verification of kernel driver porting

3.5 Matter

3.6 Bluetooth LE

3.7 Zigbee

3.8 Thread

3.9 Board Specific Package configuration

4. Building a custom application

4.1 Matter

4.2 Bluetooth LE

4.3 Zigbee

4.4 Thread

5. Enabling QPG7015M ConcurrentConnect™

6. IoT/Wi-Fi coexistence

7. Hardware reference design and other considerations

List of Abbreviations

ACS	Amazon Common Software
API	Application Programming Interface
ATT	Attribute protocol
CLI	Command Line Interface
CTC	Coexistence Test Component
DK	Development Kit
DM	Device Manager
DPK	Device Porting Kit
DTM	Direct Test Mode
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HCI	Host Controller Interface
HDMI	High-Definition Multimedia Interface
IoT	Internet-of-Things
L2CAP	Logical Link Control and Adaptation Protocol
LED	Light Emitting Diode
OT-BR	OpenThread Border Router
PTA	Packet Traffic Arbitration
PTC	Performance Test Component
RPi	Raspberry Pi
SDK	Software Development Kit
SMP	Secure Manager Protocol
SPI	Serial Peripheral Interface
SSH	Secure SHell
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
UUID	Universally Unique IDentifier
WSF	Wireless Software Foundation

Important Notices

Document Revision Information

Version	Date	Changes
1.00	20 June 2022	First draft