1. Introduction

The QPG7015M comes with a development kit to unlock its functionality to the programmer: **Qorvo® Iot Dev Kit Pro** . For more information on the QPG7015M please see its Product Brief. The QPG7015M gateway can be used to demonstrate the unique capabilities of the QPG7015M system (SW and HW) and serve as a platform for application development and prototyping (see figure 1.1). This programmer manual is intended for Software Developers and explains how to use the example applications included in the DK, how to port the software to you custom application processor and how to build custom Bluetooth LE, Zigbee, Matter™ and/or Thread applications

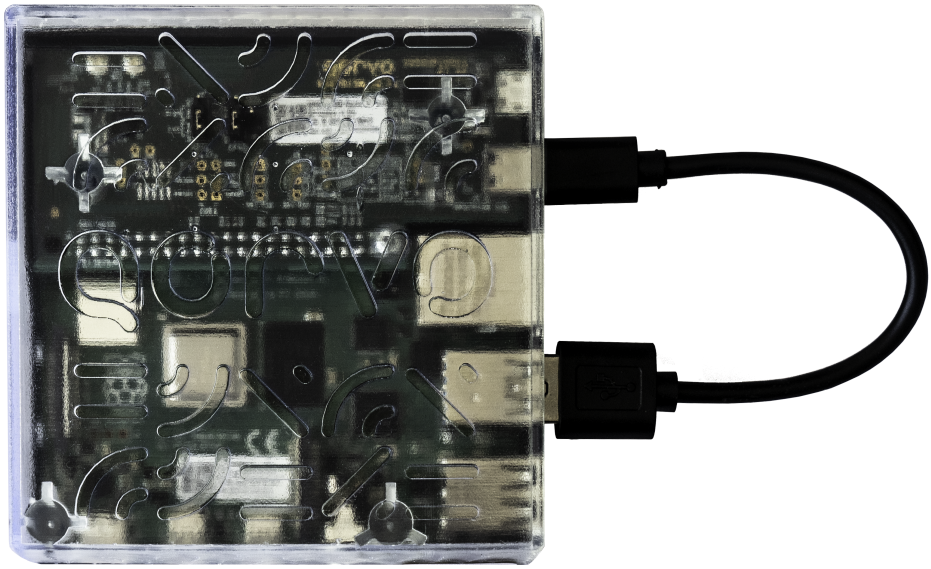


Figure 1.1: QPG7015M Gateway in Qorvo® Iot Dev Kit Pro

Contents

2. Getting started with the Development Kit

This chapter describes:

- The content of the Qorvo® Iot Dev Kit Pro.
- 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 Gateway in the Qorvo® Iot Dev Kit Pro.

2.1.1 Qorvo® Iot Dev Kit Pro box content

When ordering the Qorvo® Iot Dev Kit Pro, the customer receives a box as shown in figure 2.1. Following items can be found inside:

Figure 2.1: Qorvo® Iot Dev Kit Pro box content

Table 2.1: Qorvo® Iot Dev Kit Pro box content

No.	Name	Description
1	Power supply	Raspberry Pi 4 Power supply, 5.1 V - 3 A
2	Documentation card	QR code to the Qorvo® Iot Dev Kit Pro landing page
3	USB A ⇔ USB C cable	Used to power the Qorvo® IoT Dev Kit for QPG6105 and have serial access
4	QPG7015M Gateway	QPG7015M Radio board mounted on a RPi4 host, running the QPG7015M SDK
5	USB A ⇔ micro USB cable	Used to access the terminal of the QPG7015M Gateway over a serial communication.
6	Qorvo® IoT Dev Kit for QPG6105	QPG6105 based Development Board running the Matter Light application

2.1.2 Accessing the QPG7015M Gateway its terminal

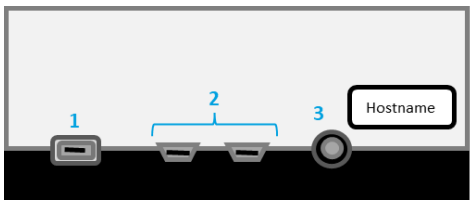


Figure 2.2: QPG7015M Gateway in Qorvo® Iot Dev Kit Pro front view

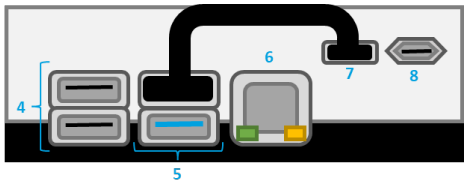


Figure 2.3: QPG7015M Gateway in Qorvo® Iot Dev Kit Pro right view

Table 2.2: QPG7015M Gateway ports

No.	Name
1	USB-C Power Supply input
2	2 Micro-HDMI inputs
3	Audio port
4	2 USB 2.0 ports
5	2 USB 3.0 ports
6	Ethernet port
7	QPG7015M RF Board USB-C port (USB communication)
8	QPG7015M RF Board microUSB port (serial access)

Access to the QPG7015M Gateway terminal is required to configure, start and stop applications. There are three options to access the terminal of the development kit. The recommended fastest approach to get started is "Terminal over Serial":

1. **Terminal over serial**: Connect the microUSB cable to port 8 shown in figure 2.3 and to your computer. Apply power to port 1 shown in figure 2.2. J8 en J9 should be configured as "RPi UART <> USB" as shown in figure 2.4. This should be already the case in the off-the-shelf QPG7015M Gateway in the Qorvo® Iot Dev Kit Pro.

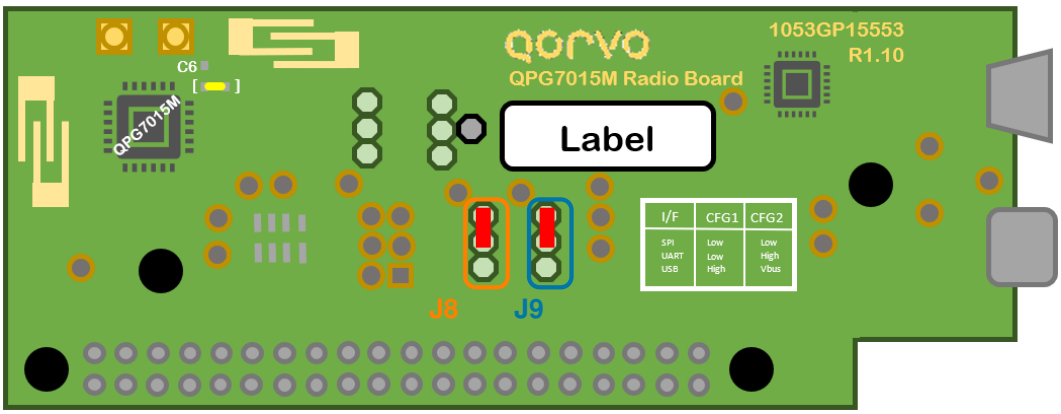


Figure 2.4: QPG7015M Terminal over Serial access

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 2.5. In the "Serial line", enter the serial port, e.g. COM6 or /dev/ttyUSB0. Next click Open. The username of the DK is *pi*, the password is *raspberry*. Please already type 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

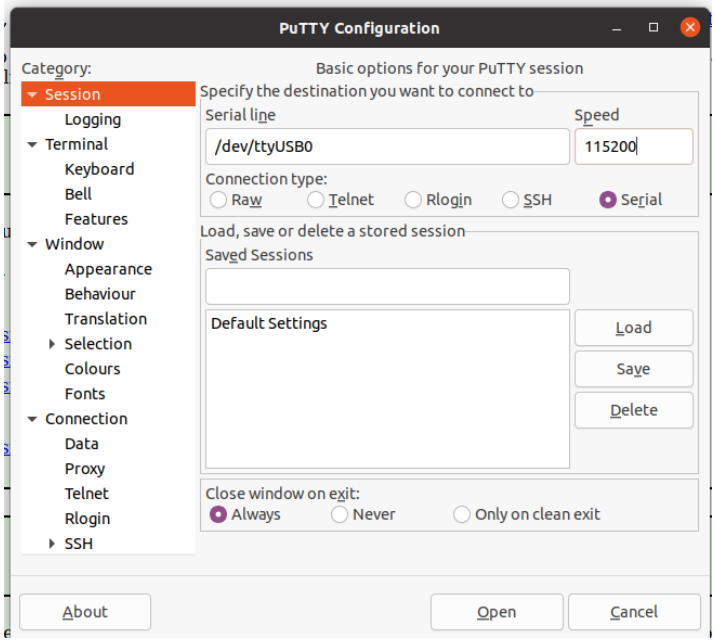


Figure 2.5: Serial configuration on Putty

2. **Secure SHell (SSH)**: Connect the QPG7015M Gateway 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 Gateway 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 DK is *pi*, the password is *raspberry*.

The alternative option is to access the QPG7015M Gateway using one of the other two options and retrieve the IP address by running following command from the DK's terminal:

```
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 port 1 of the DK. The user of the 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 and apply power to port 1, shown in figure 2.2. The ethernet connection gives the RPi connection to the internet and allows following advantages:

- The OpenThread Border Router its landing page will have icons in its lay out
- The user is enabled to download additional packages if interested

- The RPi is connected to the local network for possible SSH connections
- The WiFi functionality of the RPi is reserved by the OpenThread Border Router and cannot be used for Wireless internet access in this demo.

2.1.3 Use the QPG7015M Gateway

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

To use the QPG7015Mgateway in the Qorvo® Iot Dev Kit Pro the user must

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

To terminate the QPG7015Mgateway the user can

- Stop the QPG7015Mgateway. See page 10

OR

- Reset the QPG7015Mgateway. See page 11

Default configuration of the QPG7015Mgateway

The 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 2.3: QPG7015Mgateway 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
Production 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 Gateway automatically when it is powered.

Start the QPG7015M gateway

To start the QPG7015M gateway, 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 gateway

! This document describes in section 2.1.4 how to configure the QPG7015M Gateway using `qorvo_stack_config`. Since the scripting use `qorvo_stack_config` to start, stop and reset the gateway, it should not be editted before stopping or resetting the gateway. If not, it's possible that certain stacks are not stopped or reset.

To stop the 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...
```

Factory reset the QPG7015M gateway

! This document describes in section 2.1.4 how to configure the QPG7015M Gateway using `qorvo_stack_config`. Since the scripting use `qorvo_stack_config` to start, stop and reset the gateway, it should not be editted before stopping or resetting the gateway. If not, it's possible that certain stacks are not stopped or reset.

To stop and factory reset the 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:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
deleted:  sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
deleted:  sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
deleted:  sha256:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

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

Shutting down the Raspberry Pi

The RPi can be shut down using following command:

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

Wait for the green LED to stop blinking before removing the power as shown in figure 2.6.

Figure 2.6: QPG7015M Development Kit left view

2.1.4 Configure the QPG7015M Gateway

The QPG7015M Gateway 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 3.3. By default SPI is configured. Changing the interface protocol is done by modifying the jumper configuration on the DK AND modifying the qorvo_stack_config file.

Access jumper configuration

To open the QPG7015Mgateway, remove the 4 screws on the bottom of the DK with a Philips screwdriver as shown in picture 2.7. No need to remove the nut. Afterwards the top of the plastic casing can be removed.

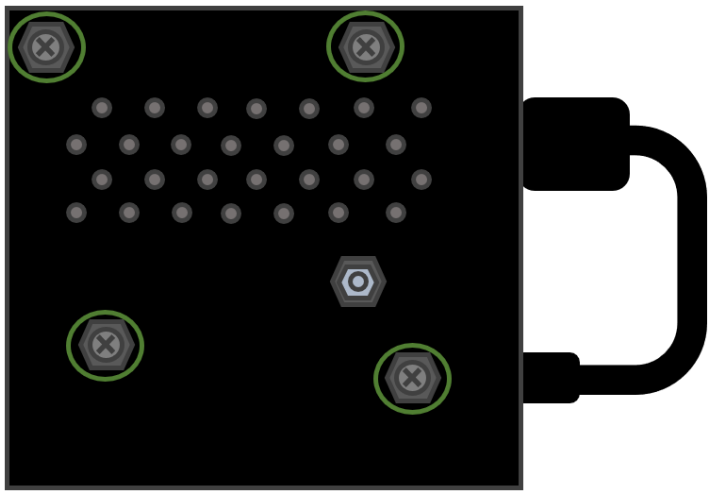


Figure 2.7: QPG7015M Gateway bottom

The jumpers CFG1 and CFG2 can be found by opening the case on the QPG7015M Radio Board. They're shown in figure 2.8

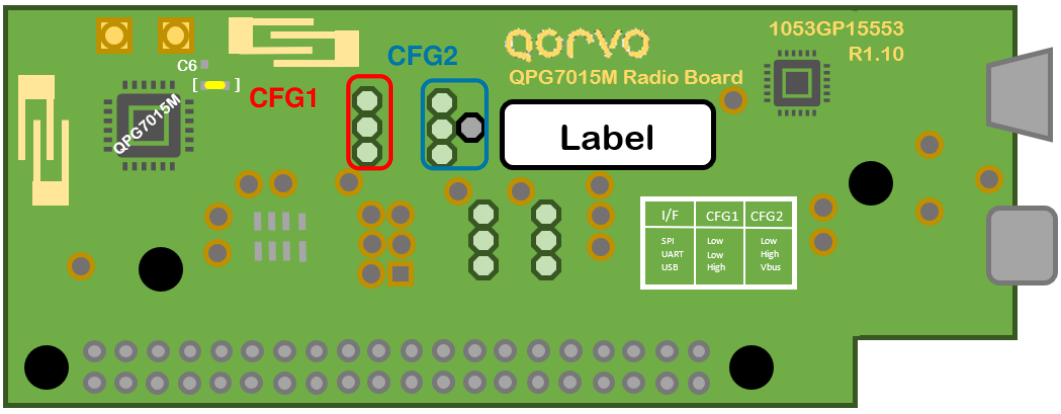
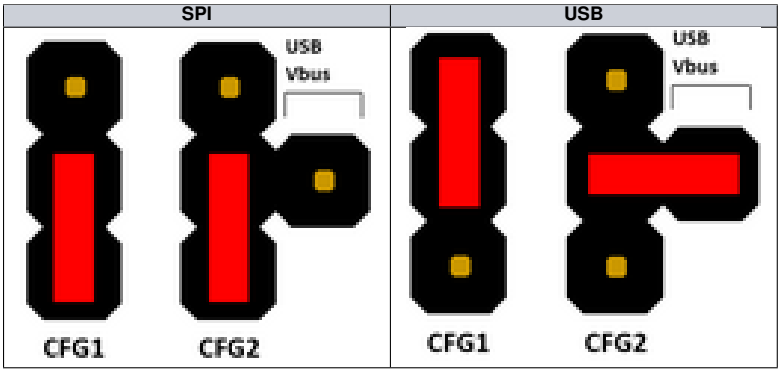


Figure 2.8: QPG7015M interface configuration

Possible jumper configurations are shown in table 2.4. For USB a USB A ⇔ USB C cable shoud be connected from port 5 to port 7 shown in figure 2.3.

Table 2.4: QPG7015M Gateway Jumper Settings for USB or SPI interface selection



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 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 Gateway, but can be modified when building a custom filesystem when porting the QPG7015M SDK 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, they 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 3.1.

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

Following stacks can be configured:

Table 2.5: Stack configurations

Configuration	Default	Description
QORVO_ZIGBEE	Disabled	• Verify our **certified** Zigbee 3.0 R22 functionality. • Enables interacting with the Zigbee 3.0 SmartHome Gateway Client Application.
QORVO_OT_CLI	Disabled	• Enables interacting with the OpenThread Command Line Interface. • Verify our Thread 1.3 functionality.
QORVO_OT_BR	Enabled	• Enables interacting with the OpenThread Border Router. • Set up a Matter network using our OpenThread Border Router.
QORVO_OT_BRBB_INTERFACE	eth0	• 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	Enabled	• 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	Enabled	• 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	Disabled	• Configure this to enable production test application. • Test the QPG7015M radio performance and functionality.
QORVO_CTC	Disabled	• Configure this to enable coexistence verification. • Test the QPG7015M PTA capability.

Configurable 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.

Table 2.6: Additional software configurations

Configuration	Default	Description
QORVO_CONFIG_PATH	`/etc`	• Indicates where the Zigbee coordinator configuration file will be stored.
QORVO_STARTUP_XML	`start.xml`	• Defines the name of the Zigbee coordinator configuration file.
QORVO_DEBUG	Disabled	• This parameter shows additional logging of the firmware and the kernel drivers in the kernel logging accessible via `dmesg`.
QORVO_CRASHREPORT	Enabled	• This parameter generates a report when the DK crashes. These reports are stored at `/home/pi`.

QORVO_COMDUMP	Disabled	• 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 and must be built from its source package (see chapter 3.2).
		• 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	Disabled	• This parameter 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	Disabled	• 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. How to build and port the applications that the QPG7015M Gateway in the Qorvo® Iot Dev Kit Pro provides can be found in chapter 3.2.

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 architecture is shown in figure 2.9.

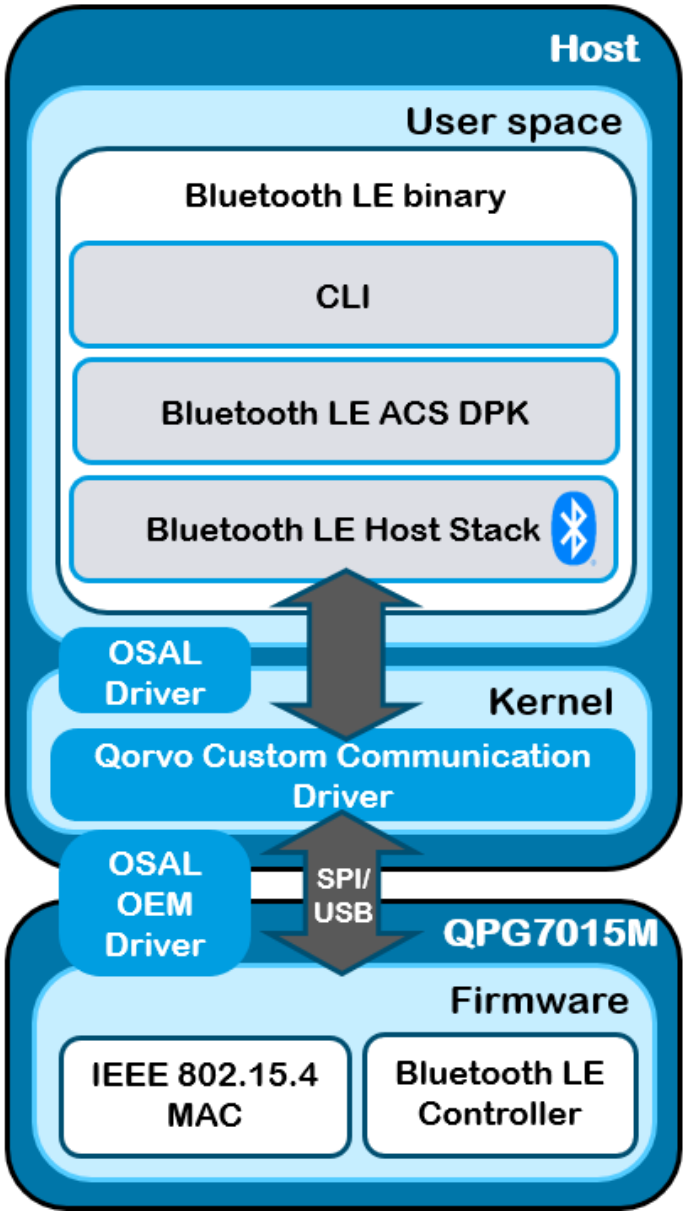


Figure 2.9: Bluetooth LE architecture

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. Start the QPG7015M Gateway 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 Google Play or in the Apple Appstore:

Figure 2.10: 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 2.11.

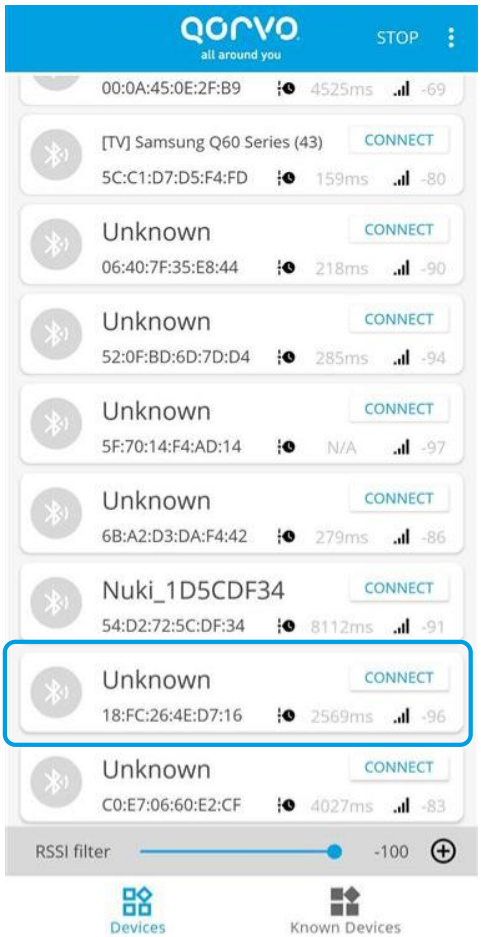


Figure 2.11: Find beacon device in Qorvo Connect

7. Tap on the device with the matching address and verify the advertising data. See figure 2.12.

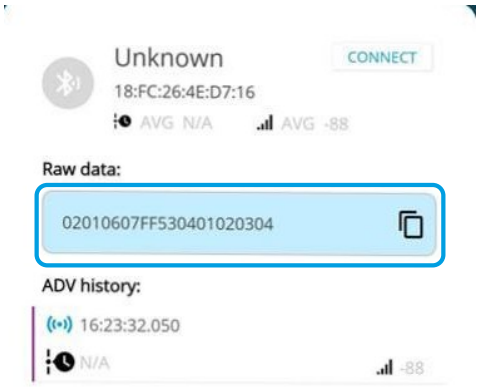


Figure 2.12: See Beacon data in Qorvo Connect

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

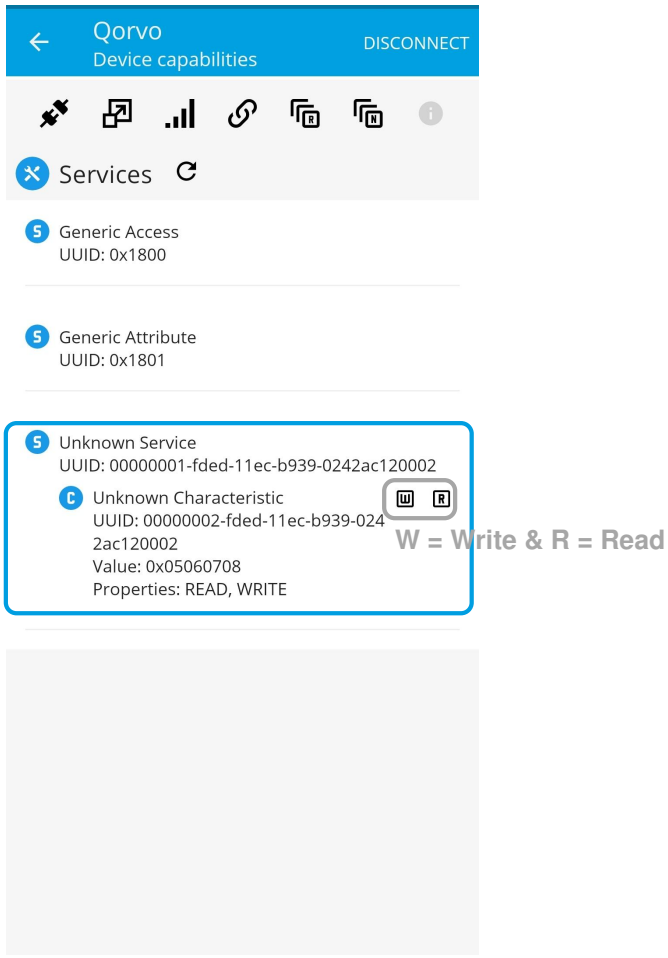


Figure 2.13: Find custom service in Qorvo Connect

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

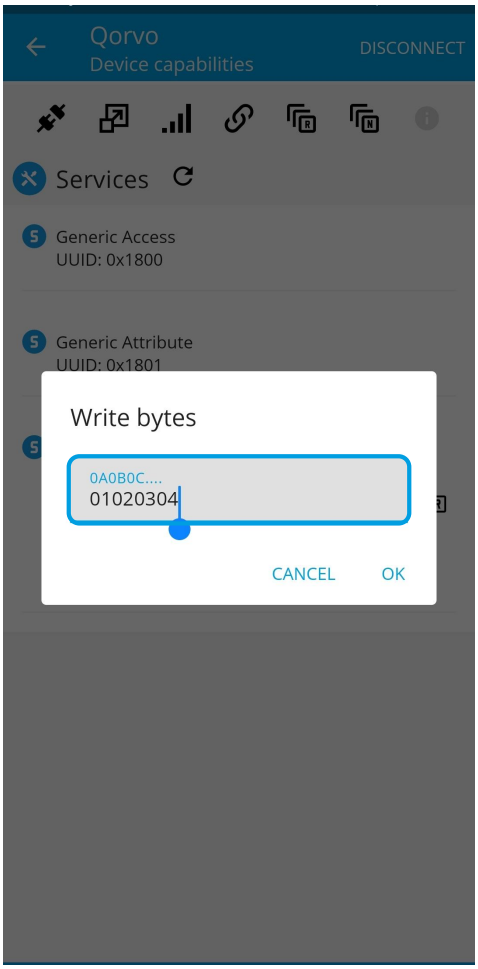


Figure 2.14: Write custom characteristic in Qorvo Connect

```
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.

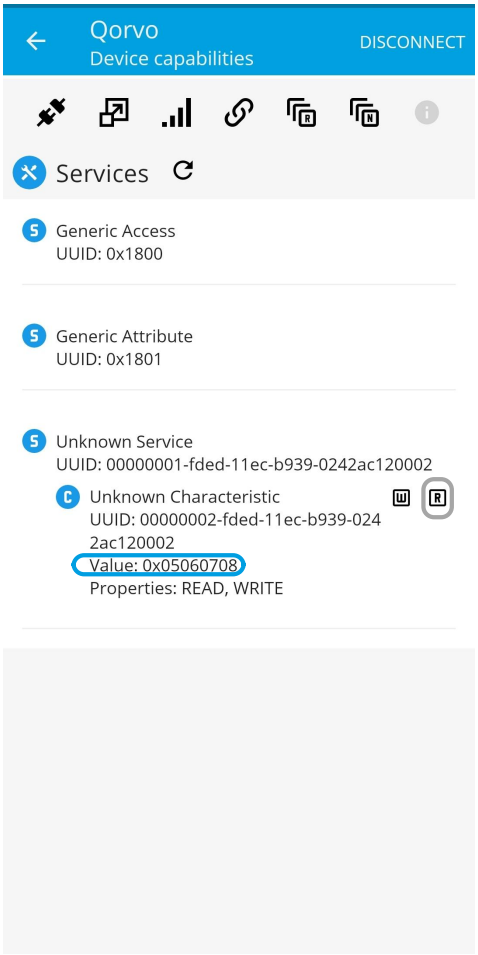


Figure 2.15: Read custom characteristic in Qorvo Connect

```
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 Gateway, 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. When the Central finds the Peripheral, stop the scan and connect to the Peripheral using the peer address reported in the scan results

   To stop scanning, type the scan 0 command. Note that while scanning is active, the input line is not visible.

```
scan 0
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

Scan Name Filtering Example

This example explains how to use filtering by the peripheral name during the scan procedure. The process of scanning resulting in a connection between the Central and the Peripheral is shown on the flowchart 2.16.

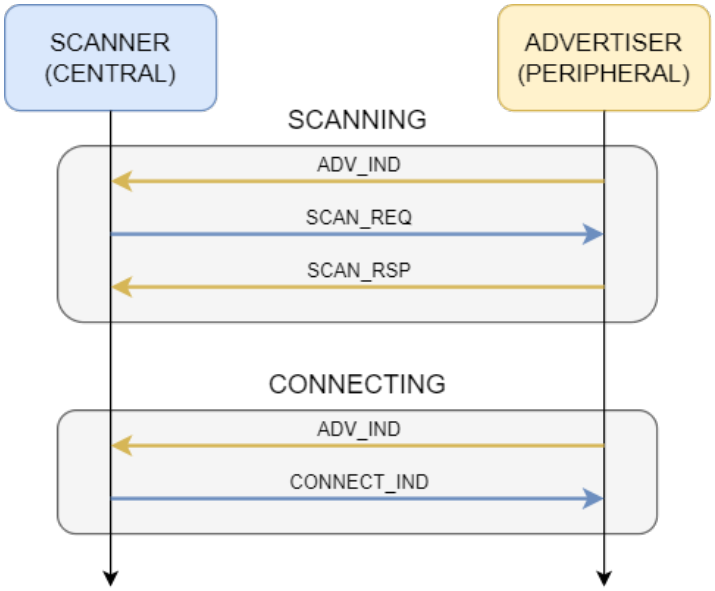


Figure 2.16: BLE connection process flowchart

1. Start the Bluetooth LE test application

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

2. On Peripheral, run

```
setdevprop 1 <custom_local_name>
setadvdata_app 0 0 0x0000 2 0
adv 0 1
```

3. On Central, run

```
setscanfilter 0 0 0 4 <custom_local_name>
setscanfiltparam 0 0 0 16 0 0 136 40 0 100 5 100 2
enablescanfilt 0 1
scan 1
```

4. When the Central finds the Peripheral, stop the scan and connect to the Peripheral using the peer address reported in the scan results

   To stop scanning, type the scan 0 command. Note that while scanning is active, the input line is not visible.

```
scan 0
connect 0 <peer_address> 1 2
```

Additional documentation

For more info on porting the Bluetooth LE example application to your own platform see section 3.2

Additional documentation can be found within GP_P1053_SW_17777_Bluetooth_LE_ACS_DPK.zip. For retrieval, see section 3.1.

- 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.

The architecture is shown in figure 2.17. The BoardConfigTool application is used to configure the QPG7015M to operate in DTM. This BoardConfigTool is called by the scripting when the gateway is started as show in section 2.2.2.

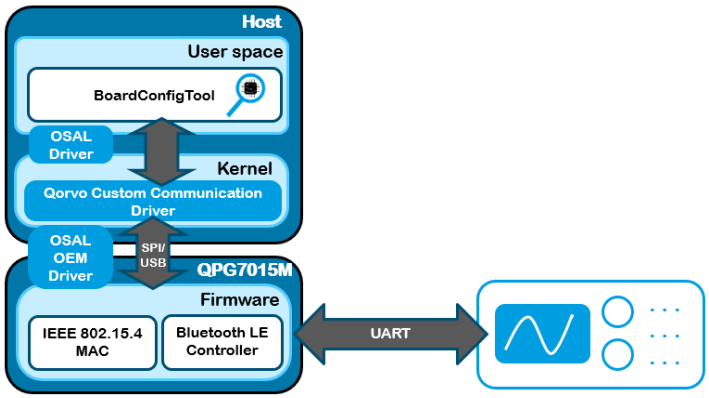


Figure 2.17: Bluetooth LE DTM architecture

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 2.18 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 2.18

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.

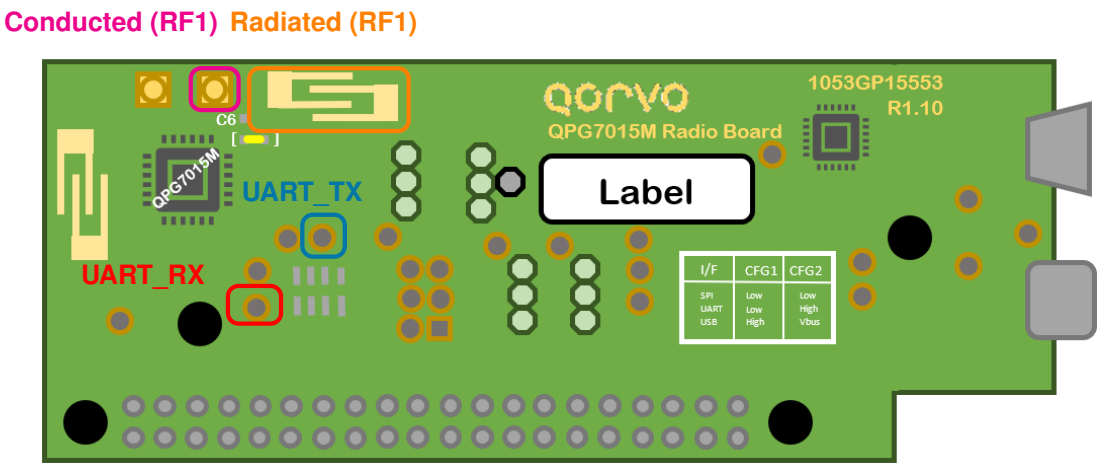


Figure 2.18: DTM Test points

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

2.2.3 RF Evaluation (PTC)

The QPG7015M Gateway provides tooling to verify the QPG7015M its RF functionality.

The architecture is shown in figure 2.19

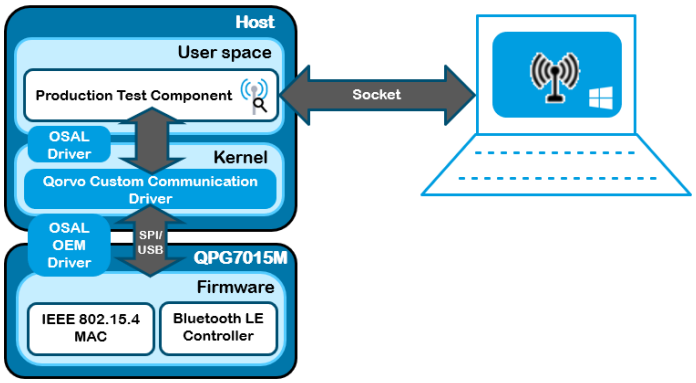


Figure 2.19: Production Test Component architecture

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

PTC Setup

The PTC functionality can be confirmed in three steps:

1. Configure the QPG7015M Gateway to open a socket.

   The user must reconfigure the `qorvo_stack_config`, after a stop or reset of the gateway (see section 2.1.3) with:

   - QORVO_PTC=1

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

2. Install the Radio Control Console on your Windows PC.

   To retrieve the required package for the installers see section 3.1.
   The installers can be found in the package its "RadioControlConsole" directory. After unpacking the package:

   - Unzip `GP_P864_SW_16383_RadioConsolePackage_r4.0.0.0.zip` in `RadioControlConsole`
   - Install RadioControlConsole_vX.X.X.X.msi
   - Install TestComponentsExtensions_vX.X.X.X.msi

```
GP_P1053_SW_17304_TestComponents_QPG7015M/
|
+-- Documents
|   +-- ApplicationNotes
|   |   +-- GP_P1053_AN_16476_Host_Production_Test_Component.pdf
|   |   +-- GP_P1053_AN_16490_Host_Coexistence_Test_Component.pdf
|   |   +-- GP_P1053_AN_16709_GNU_Debugger.pdf
|   +-- document_overview.txt
|   +-- ReleaseNotes
|   |   +-- ReleaseNotes.pdf
|   +-- UserManuals
|       +-- GP_P1053_UM_16828_Coexistence_Signaling_Validation_Guide_QPG7015M.pdf
|       +-- GP_P864_UM_16380_PTC_Overview.pdf
|       +-- GP_P864_UM_16482_QPG7015M_CTC_API.pdf
+-- End_User_License_Agreement.pdf
+-- GP_P1053_RN_17305_Host_Test_Components.pdf
+-- RadioControlConsole
|   +-- GP_P864_SW_16383_RadioConsolePackage_r4.0.0.0.zip
|   |   +-- RadioControlConsole_vX.X.X.X.msi
|   |   +-- GP_P864_RN_12252_RadioControlConsoleReleaseNotes.pdf
|   +-- Extensions
|       +-- TestComponentsExtensions_vX.X.X.X.msi
+-- README.txt
+-- Software
.    +-- build.sh
.    +-- package_defs.sh
.    +-- TestComponents_190959_vX.X.X.X.tgz
```

3. Now connect via RCC to the socket opened by the QPG7015M Gateway.

```
user@[hostname] C:\Program Files (x86)\Qorvo\RadioControlConsole\v4.0.0.0
> RadioControlConsole.exe -e [Hostname/IP address QPG7015MDK] -p 9192
Checking .NET framework version...
.NET Framework Version:  4.6.2 or later
== Quick Edit Mode enabled ==
TcpClient connection [IP address Windows PC]:62677 <-> [IP address
QPG7015MDK]:9192 established.
TcpClient connection [IP address Windows PC]:62678 <-> [IP address
QPG7015MDK]:9192 established.
Looking for TestComps DLL in C:\Program Files (x86)\Qorvo\RadioControlConsole
\Extensions
=> Using the following DLL to get target version
=> PTC_QPG7015M_RPi_vX.X.X.X.dll :  (X.X.X.X, PTC_QPG7015M_RPi)
Found target version :  X.X.X.X ; productname :  PTC_QPG7015M_RPi
Looking for compatible DLL with productname PTC_QPG7015M_RPi and version
X.X.X.X...
=> Selected the following DLL :
=> PTC_QPG7015M_RPi_vX.X.X.X.dll :  (X.X.X.X, PTC_QPG7015M_RPi)
Connected to Ethernet [IP address QPG7015MDK]:9192
console>
```

Info Example

There are two information chuncks to be received,

1. Generic chip info

```
console> INFO
Mac Address :  0x18FC2600004ED716
BLE Dev Address :  18:FC:26:4E:D7:16
PTC component Version :  X.X.X.X
Application Version :  X.X.X.X CL:XXXXXX
Chip :  0D00
Chip Product ID : QPG7015
Product Name :  PTC_QPG7015M_RPi
```

2. PTC configuration info

```
console> i

Attributes :
=============
CH = 11 (0xB)
AN = 1 (0x1)
AD = OFF (0x0)
SLP = RC (0x0)
```

```
CSMA = NOCCA (0x0)
MAXBE = 5 (0x5)
MINBE = 3 (0x3)
MC = 4 (0x4)
MR = 3 (0x3)
W = 20 (0x14)
SPAN = 51966 (0xCAFE)
SSA = 65535 (0xFFFF)
CW = M (0x0)
PCL = ON (0x1)
PACKETINTERVAL = 5 (0x5)
SCANINTERVAL = 7 (0x7)
PACKETLENGTH = 5 (0x5)
PACKETCOUNT = 10 (0xA)
SCANCOUNT = 3 (0x3)
PI = ON (0x1)
PM = ON (0x1)
SETTXDATA = ()
PHY = RF4CE (0x0)
BLETESTPACKET = 0 (0x0)
BLEDATARATE = 1M (0x1)
MAPCLKIOPIN = Not set ()
MAPCLKTYPE = Not set ()
MIDLEVELAGCATT = OFF (0x0)
RXMS = OFF (0x0)
RXHS = ON (0x1)
RXMC = OFF (0x0)

RX mode :  ON
SETCW mode :  OFF
```

Continuous Wave Example

This sets up an unmodulated continuous wave on a configured channel (e.g. channel 11) Run following commands:

1. Configure the QPG7015M MAC to IEEE802.15.4 mode

```
console> PHY RF4CE
Attrib changed :  PHY to RF4CE
```

2. Disable listening mode

```
console> RX OFF
RX set to OFF
```

3. Configure to channel 11

```
console> CH 11
Attrib changed :  CH to 11
```

4. Start an **unmodulated** continuous wave.

```
console> SETCW U ON
SETCW set to ON
```

Energy Detect Example

This gives an indication of the amount of energy measured on a configured channel (e.g. channel 11). Run following commands:

1. Configure the QPG7015M MAC to IEEE802.15.4 mode

```
console> PHY RF4CE
Attrib changed :  PHY to RF4CE
```

2. Configure the QPG7015M to operate in High Sensitivity listening mode

```
console> RXHS ON
Attrib changed :  RXHS to ON
```

3. Enable listening mode

```
console> RX ON
RX set to ON
```

4. Enable Energy Detect mode

```
console> ED ON
ED set to ON
```

5. Configure to channel 11

```
console> CH 11
Attrib changed :  CH to 11
```

6. Run **3 Energy Detect mode scans with 7 ms scan interval.**

```
console> ED 3 7
ED 0
ED 0
ED 0
ED Scan finished
```

Quit RCC

The RCC application can be terminated by running:

```
console> quit
```

Additional documentation

For more info on porting the PTC application to your own platform see section 3.1

Additional documentation can be found within GP_P1053_SW_17304_TestComponents_QPG7015M.zip. For retrieval, see section 3.1.

- GP_P1053_RN_17305_Host_Test_Components.pdf
  ⇒ Release notes PTC QPG7015M Gateway application.
- GP_P864_RN_12252_RadioControlConsoleReleaseNotes.pdf
  ⇒ Release notes RCC.
- GP_P1053_AN_16476_Host_Production_Test_Component.pdf
  ⇒ Application Note on building and QPG7015M Gateway configuration.
- GP_P864_UM_16380_PTC_Overview.pdf
  ⇒ Application Note with a full list of compatible RCC commands.

2.2.4 PTA Evaluation (CTC)

The QPG7015M supports Packet Traffic Arbitration. It can be used in products that have multiple Wireless interfaces (Wi-Fi, 802.15.4 and/or BT/BLE) to better avoid collision and improve overall performance. Thanks to multiple priority lines, the device supports multiple Coex slaves.
The architecture is shown in figure 2.20

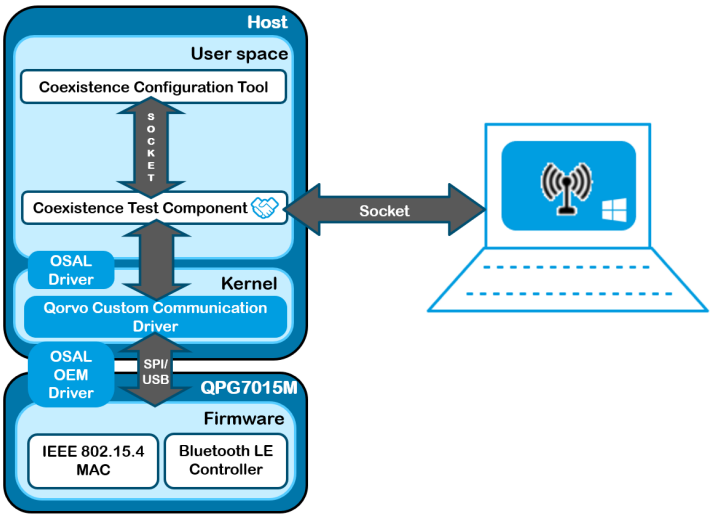


Figure 2.20: Coexistence Test Component architecture

Coexistence pinout

PTA defines 4 connections:

1. Ground
2. Grant
   ⇒ Indicates that the medium is free to be used.
3. Request
   ⇒ Requests Grant for the medium to the host chip.
4. Priority
   ⇒ Comes with a request, indicates the importance of the packet. ⇒ The QPG7015M supports two priority lines, Prio_0 and Prio_1 (optional) , indicating a binary number of the priority:
   - 00 ⇒ 0
   - 01 ⇒ 1
   - 10 ⇒ 2
   - 11 ⇒ 3

The testpoints for these pins are available on the QPG7015M RF board:

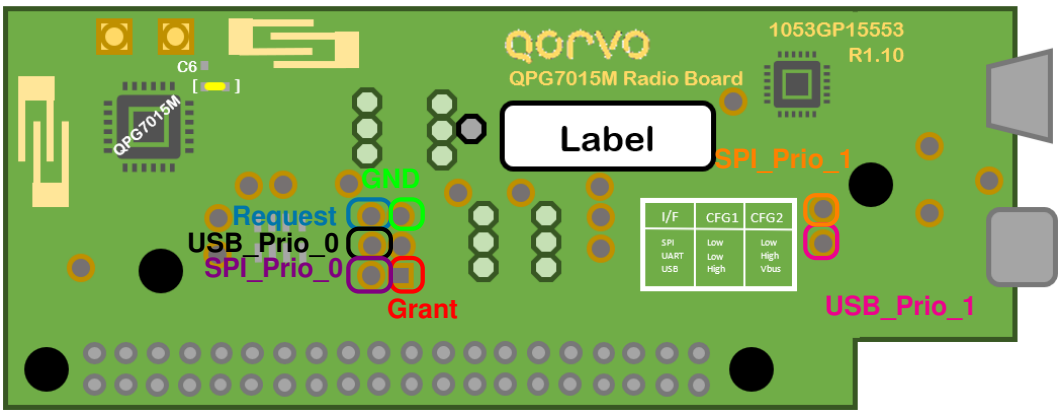


Figure 2.21: Coexistence Test points

The Grant line is active high. This means that it must be pulled low to have access to the medium when coexistence is enabled.

Additional documentation

For more info on porting the Coexistence example application to your own platform see section 3.2

Additional documentation can be found within GP_P1053_SW_16844_CoexConfigTool.zip and GP_P1053_SW_17304_TestComponents_QPG7015M.zip. For retrieval, see section 3.1.

GP_P1053_SW_16844_CoexConfigTool.zip

- GP_P1053_RN_17922_Coexistence_Config_Tool.pdf
  ⇒ Release notes
- gpCoex_API_Manual.pdf
  ⇒ Manual describing how to configure the PTA using the available API.
- GP_P1053_UM_19488_CoexConfigTool.pdf
  ⇒ Application note describing what configuration options are available for the on board Coexistence Configuration Tool via the socket.
- GP_P1053_UM_17788_Quick_Start_Guide_CoEx_Signaling_Validation_QPG7015M.xlsx
  ⇒ Quickstart guide explaining scenarios on how to configure the PTA using an on board or an external tool.

GP_P1053_SW_17304_TestComponents_QPG7015M.zip

- GP_P1053_AN_16490_Host_Coexistence_Test_Component.pdf
  ⇒ Explains how to build and run the CTC application that opens the socket.
- GP_P864_UM_16482_QPG7015M_CTC_API.pdf
  ⇒ Application note describing what configuration options are available for the external RCC application via the socket.
- GP_P1053_UM_16828_Coexistence_Signaling_Validation_Guide_QPG7015M.pdf
  ⇒ Shows how to validate the coexistence signaling between the QPG7015M and a third-party Wi-Fi (WLAN) chip or host processor

2.2.5 Matter

The Matter protocol is an industry-unifying standard for IoT. It defines a unified application layer and data model for devices operating with different IP protocols including Wi-Fi and Thread. The Matter stack is developed and maintained as an open-source project. The Matter protocol is built around four core principles:

- Simplicity
- Interoperability
- Reliability
- Security

For more information about the Matter standard, please refer to Matter repository on GitHub. The QPG7015M gateway supports the OpenThread Border Router role, which plays a crucial part in any Matter supporting gateway. This functionality combined with our ConcurrentConnect™ technology, makes the QPG7015M a suiting candidate to operate as a Matter controlling device.
The architecture is shown in figure 2.22

! The docker implementation used, is to facilitate our scripting to showcase this technology. It is optional and thus no requirement for the customer's own platform.

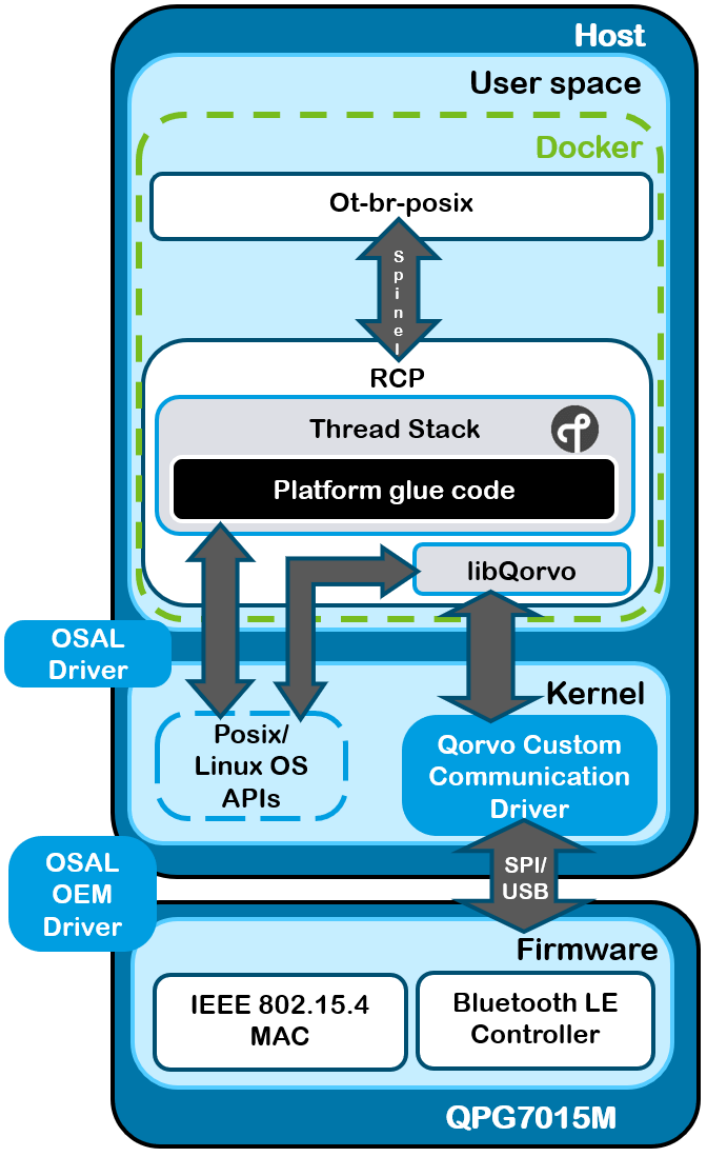


Figure 2.22: OpenThread Border Router architecture

5 interesting blocks can be seen:

1. **ot-br-posix** ⇒ OpenThread Border Router application, maintained by the OpenThread community. It communicates to the Thread Stack using the Spinel protocol.
2. **RCP** ⇒ Radio Co-Processor defines the architecture where the Thread Stack lives on the Host Processor side.
3. **Thread Stack** ⇒ Enables Thread functionality, maintained by the OpenThread community
4. **Platform Glue code** ⇒ Interface for OpenThread with platform specific functionalities such as alarms, storage, random number generator,
5. **libQorvo** ⇒ Implements platform specific functionality and hardware. It provides timers, random and radio APIs in a way the OpenThread platform glue code can use it. On the bottom side, it

uses a Posix API to interact with OS and Custom Communication Driver via IOCTL to interact with radio hardware.

Matter network example

This example explains how a Matter Light bulb can join the OpenThread Border Router and how that light can be toggled using a Matter controller. The steps are explained on Qorvo's Matter Github repository:
https://github.com/Qorvo/QMatter/blob/master/Documents/Guides/commissioning_android_chiptool.md

Additional documentation

For more info on porting the OpenThread Border Router to your own platform see section 3.2.2. Additional documentation can be found on our GitHub page: https://github.com/Qorvo/QGateway/ For more architectural info, check out the QPG7015M setup instructions.

2.2.6 Zigbee

The QPG7015M Gateway contains a Zigbee deliverable that enables the device to operate as a Zigbee coordinator. The QPG7015M is certified for Zigbee 3.0 R22. To showcase its functionality, the QPG7015M Gateway contains a Zigbee coordinator application named "SmartHomeGatewayClient" which exposes the Zigbee 3.0 R22 Coordinator functionality through a command line interface for easy testing and evaluation. Customers developing Zigbee 3.0 applications can use this reference application as an example implementation how to access and use the API.

The architecture is shown in figure 2.23.

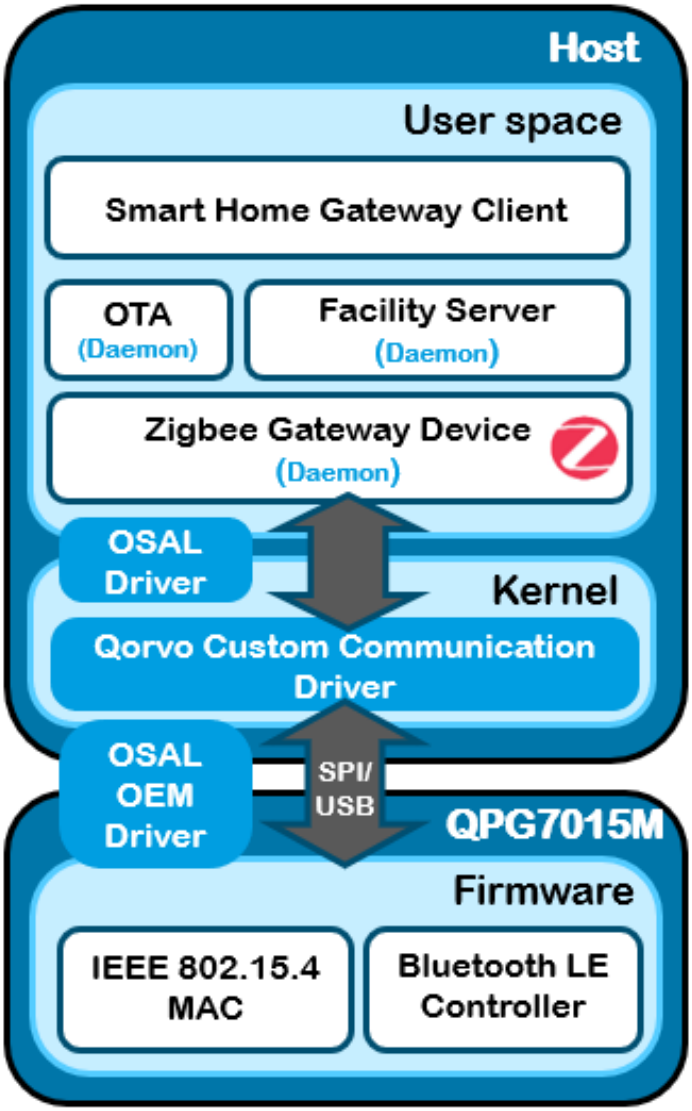


Figure 2.23: Zigbee 3.0 architecture

Within figure 2.23, four blocks are shown:

1. Smart Home Gateway Client
   - A Facility Client and OTA Client application that showcase the OTA and Facility API
2. Facility Server
   - Manages a facility, by keeping an inventory of all the joined devices
   - Interfaces with one or more Facility clients, such as Smart Home Gateway Client or a smartphone application.
   - Runs as a daemon, named "facilityd"
3. OTA Server
   - Enables OTA upgrades towards nodes in the network that support the Zigbee OTA cluster

- Runs as a daemon, named "otad"

4. Zigbee Gateway Device
   - Provides communication between the Zigbee Stack and paired TCPIP based host applications.
   - Enables Zigbee Cluster framework operations. E.g. readingwriting cluster attributes.
   - Enables Zigbee Device Object operations. E.g. opening networks, querying node addresses, querying node descriptors, managing binbdings and handling device anouncements.
   - Enables network forming and joining.
   - Enables dynamical endpoint management on local gateway device.
   - Enables network scanning
   - Enables Service discovery on all available device endpoints of a Zigbee node.
   - Runs as a daemon, named "zgdd"

! Prior to running the Smarthome Gateway Client application, two action items:

1. Configure Zigbee to be enabled in `qorvo_stack_config`: `QORVO_ZIGBEE=1` (See subsection 2.1.4.)
2. Start the QPG7015M Gateway as described in subsection 2.1.3.

Zigbee coordinator example with SmartHome Gateway Client running on RPi

This examples showcases the QPG7015M gateway its Zigbee coordinator functionality. A Sengled light bulb that operates as a Zigbee router will join the network formed by the DK. Afterwards the Zigbee coordinator can be used to operate the light.

1. Start the Smarthome Gateway Client application

```
pi@[hostname]:~$ cd ZigBee3.0 && LD_LIBRARY_PATH=/home/pi/ZigBee3.0/lib
./SmartHomeGatewayClient_RPi.elf connect --host=localhost --port=8888
```

2. Form a network

```
permit joining --enable
```

3. Toggle the light bulb manually 10 times on/off. Wait until the device has joined the network.

```
A device with ID 0 joined the network.
```

4. See the facility inventory

```
> show f

> ****************************************
S/N 999999, description = "[Description]", location = "[Location]"
Not enrolled (no credentials).
The following host addresses are available: [ "localhost:8888",
"raspberrypi-XXXX.local.:8888" ]
```

```
Currently connected to this facility.
Network closed for new devices.
Offset between clocks: 0 second(s).
Service started at Mon XXX XX XX:XX:XX XXXX BST (2464 seconds ago)

*** Device 0, 354e/b0ce1814030b9192 -
Manufacturer ID = 0x1160
Vendor = sengled
Model = E11-G23
Location =
Heartbeat interval: 363 seconds (approach 0, available: 0x01)
Device is alive (determinate).
Device connectivity state: active.
Last activity: 16611788919978876 (Mon XXX XX XX:XX:XX XXXX BST) (121 seconds
ago)
Applications: [ #1 ]
```

5. The facility inventory shows that this light has 1 endpoint #1, check out the endpoint

```
show e 0 1

> Application 0.#1 ("")
Profile ID = 0x0104, device ID = 0x0101, version = 0x00
Inbound (Server) Cluster 0x0000 "basic"
[inventory] Attribute 0x0000, type = unsigned8, value = 1
[inventory] Attribute 0x0001, type = unsigned8, value = 1
[inventory] Attribute 0x0002, type = unsigned8, value = 0
[inventory] Attribute 0x0003, type = unsigned8, value = 1
[inventory] Attribute 0x0004, type = string, value = { 07, 73, 65, 6e, 67, 6c,
65, 64 }
[inventory] Attribute 0x0005, type = string, value = { 07, 45, 31, 31, 2d, 47,
32, 33 }
[inventory] Attribute 0x0007, type = enum8, value = { 00 }
Inbound (Server) Cluster 0x0003 "identify"
Inbound (Server) Cluster 0x0004 "groups"
Inbound (Server) Cluster 0x0005 "scenes"
Inbound (Server) Cluster 0x0006 "on/off"
[__cache__] Attribute 0x0000, type = bool, value = { 00 }
Inbound (Server) Cluster 0x0008 "level control"
[__cache__] Attribute 0x0000, type = unsigned8, value = 255
[inventory] Attribute 0x0011, type = unsigned8, value = 255
Inbound (Server) Cluster 0x0702 "metering"
[inventory] Attribute 0x0200, type = bitmap8, value = { 00 }
[inventory] Attribute 0x0300, type = enum8, value = { 00 }
[inventory] Attribute 0x0301, type = unsigned24, value = 1
[inventory] Attribute 0x0302, type = unsigned24, value = 10000
[inventory] Attribute 0x0306, type = bitmap8, value = { 00 }
[__cache__] Attribute 0x0400, type = signed24, value = 0
Inbound (Server) Cluster 0x0b05
Outbound (Client) Cluster 0x0019 "over-the-air upgrade"
```

```
Binding to 18fc2600004ed716.EP#1 for clusters [ 0x0006, 0x0008, 0x0702
```

6. This endpoint contains the on/off Server cluster with identifier 0x0006. Section 3.8.2.2 of the Zigbee Cluster Library Specification shows it contains an OnOff attribute with identifier "0x00". A command can be send to toggle (0x02) the light as described in the Zigbee Cluster Library Specification Section 3.8.2.3.3

```
> zcl send 0 1 6 0x02

> ZCL send completed with status 0x0 and sequence number 0x0.
APS Callback triggered with status 0x0 and sequence number 0x0.
Short Destination Address:  0x0
Short Source Address:  0x354e
Profile:  0x104
Cluster:  0x6
Data:  0x08000b0200
```

7. The device can be removed from the network

```
> device remove 0

> Device removal completed with status 0x0.
```

8. It will not pop up in the facility inventory

```
> show f

> ********************************************
S/N 999999, description = "[Description]", location = "[Location]"
Not enrolled (no credentials).
The following host addresses are available:  [ "localhost:8888",
"raspberrypi-XXXX.local.:8888" ]
Currently connected to this facility.
Network closed for new devices.
Offset between clocks:  0 second(s).
Service started at Mon XXX XX XX:XX:XX XXXX BST (82006 seconds ago)
```

9. It will not react to new ZCL commands

```
> zcl send 0 1 6 0x02

ERROR: No such application.
```

Zigbee coordinator example with SmartHome Gateway Client running on smartphone

This example will showcase how an external Client application (an app on the smartphone) can access the facility daemon on the DK in the same ethernet network. A Sengled light bulb that operates as a Zigbee router will join the Zigbee network formed by the QPG7015M Gateway and triggered by a smartphone app. Afterwards the smartphone can be used to operate the light.

For this demo, your QPG7015M Gateway and smartphone should be connected to the same network.

1. Install Ubisys Smart Home on smartphone

   These examples are to be tested against a Smarthome Gateway Client application. **Ubisys Smart Home** can be installed in Google Play or in the Apple Appstore:

Figure 2.24: Install codes Ubisys Smart Home

2. Retrieve the IP address from the QPG7015M Gateway.

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

3. Launch the Ubisys Smart Home app, open the menu by clicking the "+" in the lower right corner and select "IP address"

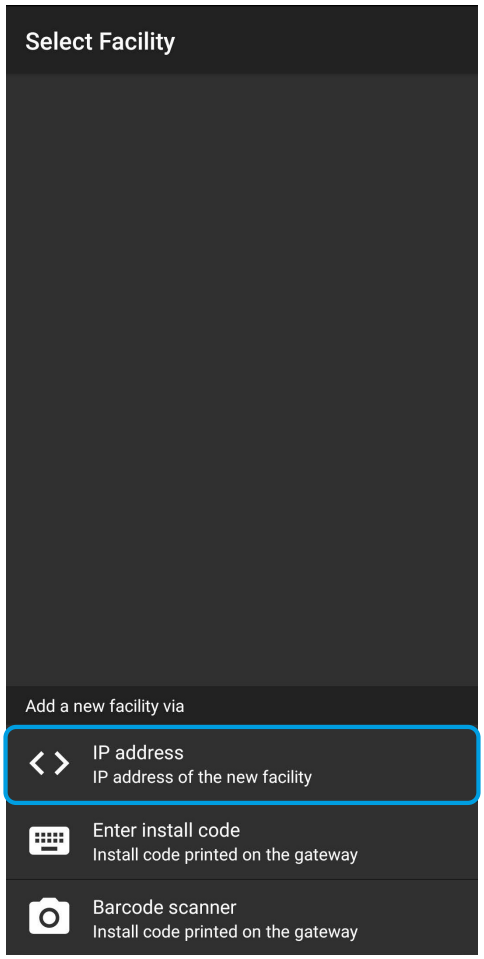


Figure 2.25: "Add Facility" menu

4. Enter the IP address of the QPG7015M gateway and add ":8888". Press "Ok" afterwards.

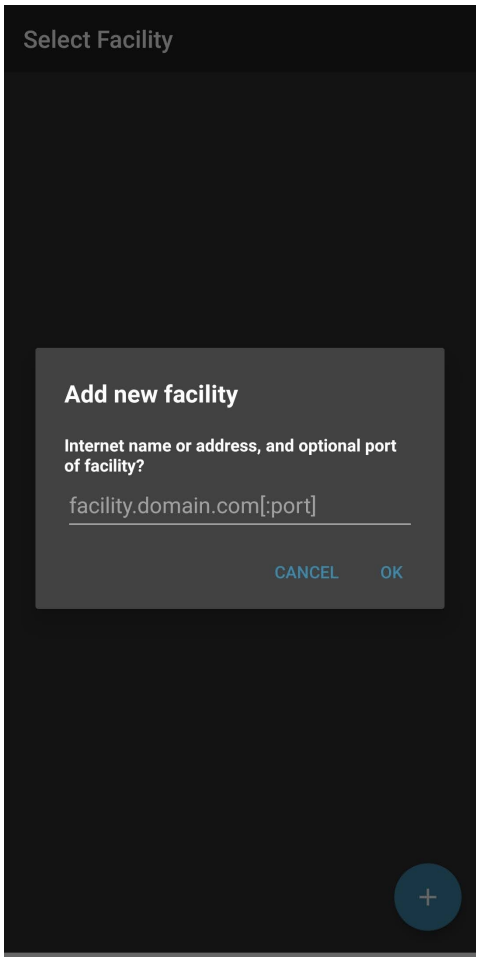


Figure 2.26: IP address menu

5. Enter the access code `1234`. Press "Ok" afterwards.

! After running the start gateway script, you only have 30 seconds to enter the access code.

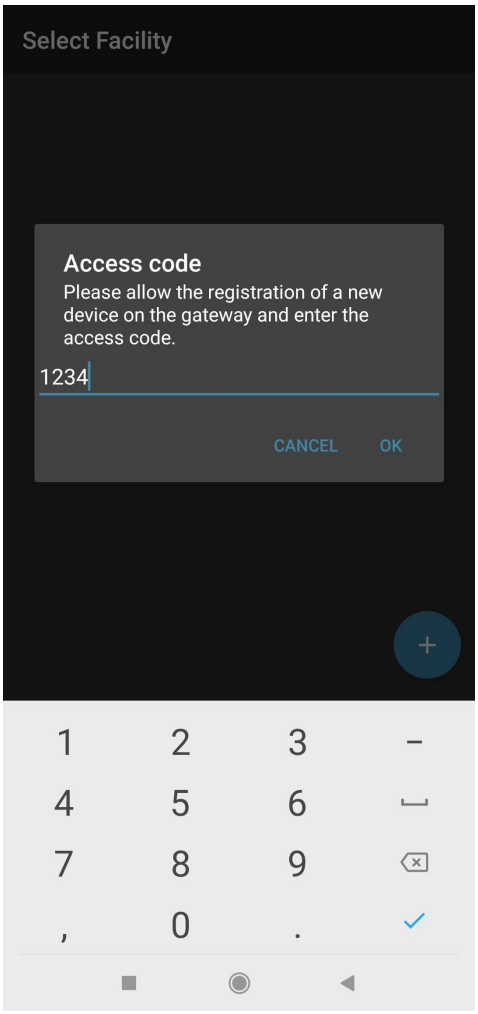


Figure 2.27: Access code

6. In the next menu move to **Configuration** ⇒ **Basic Configuration**
7. Tap **Open for new devices**, the lock should be unlocked on your screen.
8. Toggle the light bulb manually 10 times on/off. Wait until the device has joined the network. This is indicated by a [1] shown next to "Define Components".

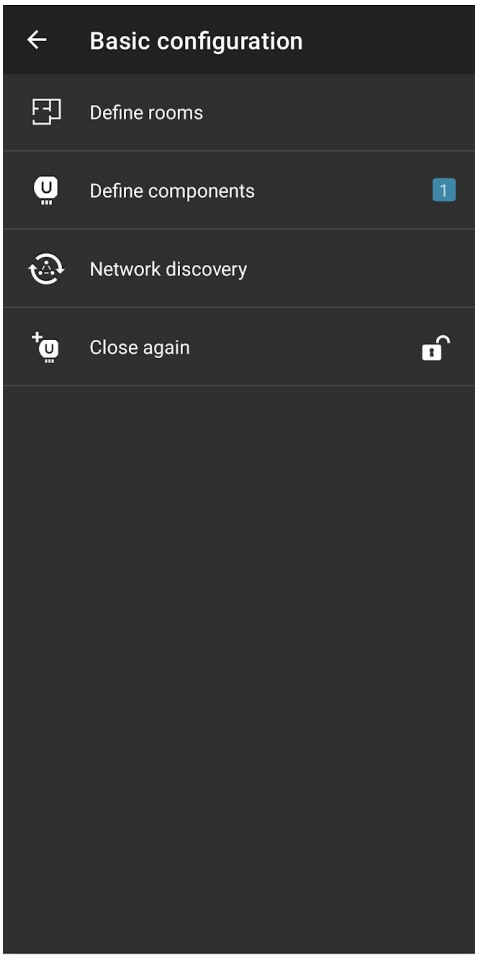


Figure 2.28: Join light bulb to Zigbee Coordinator

9. Tap close again, the lock should appear locked again
10. Move back to **Basic configuraton** ⇒ **Define rooms** .
11. Press the "+" in the upper right corner, add a room e.g. "test", press OK and pick a symbol. Your screen should look as like shown in Figure 2.29.

Figure 2.29: Test Room

12. Press the "+" next to the new room and add the light bulb to the room. Save by tapping ✓.

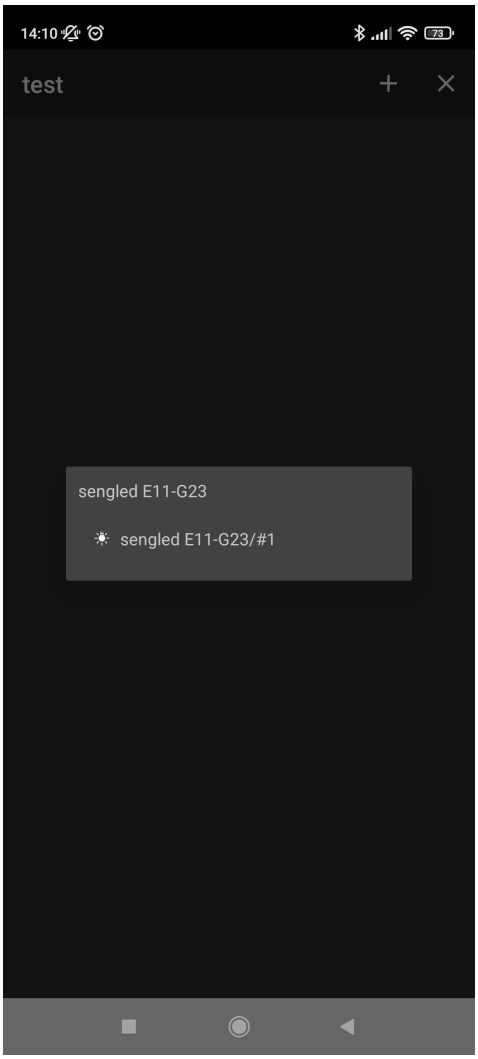


Figure 2.30: Add light bulb to room

13. Move back (4 times) to the main menu and open **(**Building Control).
14. Open the room that was created. The user now has the option to play with the On/Off switch or dim the light.

Figure 2.31: Control light

15. Move back to the main menu and open (Configuration) ⇒ **Basic Configuration** ⇒ **Define components**.
16. Keep the Sengled light pressed for a long time.
17. Tap remove in the upper right corner to remove the light bulb from the Zigbee network

Over-The-Air Upgrade example

This examples showcases the QPG7015M Gateway its Zigbee coordinator Over-The-Air Upgrade functionality. An end node which supports the OTA upgrade cluster will join the network and receive a firmware update.

1. Copy the new firmware to the correct directory

```
pi@[hostname]:~$ sudo cp test.zigbee /data/ota/firmware/
```

2. Start the Smarthome Gateway Client application

```
pi@[hostname]:~$ cd ZigBee3.0 && LD_LIBRARY_PATH=/home/pi/ZigBee3.0/lib
./SmartHomeGatewayClient_RPi.elf connect --host=localhost --port=8888
```

3. Form a network

```
permit joining --enable
```

4. Start commissioning the end node. Wait until the device has joined the network.

```
A device with ID 0 joined the network.
```

5. See the facility inventory

```
> show f

> ********************************************
S/N 999999, description = "[Description]", location = "[Location]"
Not enrolled (no credentials).
The following host addresses are available: [ "localhost:8888",
"raspberrypi-XXXX.local.:8888" ]
Currently connected to this facility.
Network closed for new devices.
Offset between clocks: 0 second(s).
Service started at Mon XXX XX XX:XX:XX XXXX BST (2464 seconds ago)

*** Device 0, 2a4d/00155f0400004983 -
Manufacturer ID = 0x10d0
Vendor = Qorvo
Model = Light
Location =
Heartbeat interval: 381 seconds (approach 0, available: 0x01)
Device is alive (determinate).
Device connectivity state: active.
Last activity: 16612614132736789 (Mon XXX XX XX:XX:XX XXXX BST) (22 seconds
ago)
Applications: [ #1, #2, #242 ]
```

6. The facility inventory shows that this light has 3 endpoints, endpoint #1 shows the OTA upgrade cluster.

```
> show e 0 1

> Application 1.#1 ("")
Profile ID = 0x0104, device ID = 0x0101, version = 0x01
Inbound (Server) Cluster 0x0000 "basic"
[inventory] Attribute 0x0000, type = unsigned8, value = 2
[inventory] Attribute 0x0001, type = unsigned8, value = 1
[inventory] Attribute 0x0002, type = unsigned8, value = 1
```

```
[inventory] Attribute 0x0003, type = unsigned8, value = 1
[inventory] Attribute 0x0004, type = string, value =  05, 51, 6f, 72, 76, 6f
[inventory] Attribute 0x0005, type = string, value =  05, 4c, 69, 67, 68, 74
[inventory] Attribute 0x0006, type = string, value =  0f, 32, 30, 31, 37, 30,
34, 30, 34, 2d, 44, 45, 2d, 46, 42, 30
[inventory] Attribute 0x0007, type = enum8, value =  01
[inventory] Attribute 0x0008, type = enum8, value =  ff
[inventory] Attribute 0x0009, type = enum8, value =  ff
[inventory] Attribute 0x0011, type = enum8, value =  00
[inventory] Attribute 0x4000, type = string, value =  07, 01, 30, 2e, 03, 30,
2e, 01
Inbound (Server) Cluster 0x0001 "power configuration"
[inventory] Attribute 0x0010, type = bitmap8, value =  00
Inbound (Server) Cluster 0x0003 "identify"
Inbound (Server) Cluster 0x0004 "groups"
Inbound (Server) Cluster 0x0005 "scenes"
Inbound (Server) Cluster 0x0006 "on/off"
[__cache__] Attribute 0x0000, type = bool, value =  00
[inventory] Attribute 0x4003, type = enum8, value =  ff
Inbound (Server) Cluster 0x0008 "level control"
[__cache__] Attribute 0x0000, type = unsigned8, value = 1
[inventory] Attribute 0x000f, type = bitmap8, value =  01
[inventory] Attribute 0x0010, type = unsigned16, value = 65535
[inventory] Attribute 0x0011, type = unsigned8, value = 255
[inventory] Attribute 0x4000, type = unsigned8, value = 255
Outbound (Client) Cluster 0x0003 "identify"
Outbound (Client) Cluster 0x0019 "over-the-air upgrade"
[inventory] Attribute 0x0000, type = ieee64, value =  ff, ff, ff, ff, ff, ff,
ff, ff
[inventory] Attribute 0x0001, type = unsigned32, value = 4294967295
[inventory] Attribute 0x0002, type = unsigned32, value = 16974082
[inventory] Attribute 0x0003, type = unsigned16, value = 2
[inventory] Attribute 0x0007, type = unsigned16, value = 4304
[inventory] Attribute 0x0008, type = unsigned16, value = 65535
Binding to 18fc2600004ed716.EP#1 for clusters [ 0x0006, 0x0008 ]
```

7. Enable OTA on the QPG7015M Gateway

```
ota enable

> OTA service is succesfully enabled.
```

8. Verify OTA is enabled

```
ota status

> OTA service is Enabled.
```

9. Open the OTA Update window

```
ota update window 0xff 0 1439

> OTA update window succesfully set.
```

10. Verify your End Node starts downloading the newer firmware.

Additional documentation

For more info on porting the Zigbee Smarthome Gateway example to your own platform see section 3.2

Additional documentation can be found within GP_P1053_SW_17645_Smart_Home_Gateway_Client.zip. For retrieval, see section 3.1.

- GP_P332_RN_11552_Smart_Home_Gateway_Reference_Application.pdf
  ⇒ Release notes
- GP_P332_AN_11551_Smart_Home_Gateway_SDK.pdf
  ⇒ An application note, describing full functionality and architecture of the Smarthome Gateway Client Application.
- GP_P332_AN_13115_Smart_Home_Gateway_ZCL_Clusters.pdf
  ⇒ An application note describing full functionality on how the Smarthome Gateway Client Application can be used to interact with Zigbee attributes. ⇒ Elaborates on matching API for this functionality.
- GP_P332_AN_13802_Smart_Home_Gateway_Device_Monitoring.pdf
  ⇒ An application note describing how to interpret the device monitoring, done by the facility.
- GP_P1053_AN_16709_GNU_Debugger.pdf
  ⇒ An application note explaining how the GNU debugger can be used to debug the application.
- gpGatewayClient_API_Manual.pdf
  ⇒ A reference manual with a full overview and info on the provided API for the Smarthome Gateway Client application.
- GP_P332_UM_12175_Facility_Service_Client_C_Technical_Reference.pdf
  ⇒ A reference manual with a full overview and info on the provided API to use the Facility daemon.
- GP_P332_UM_12176_Gateway_Service_Technical_Reference.pdf
  ⇒ A reference documentation giving more technical information on the Zigbee Gateway Device Service.

2.2.7 Thread

The QPG7015M gateway contains a Thread deliverable that enables the device to operate in/form a Thread network. The QPG7015M is certified for Thread 1.3. To showcase its functionality, the QPG7015M gateway contains a Thread CLI application named which exposes the Thread 1.3 functionality through a command line interface for easy testing and evaluation. Customers developing Thread applications can use this reference application as an example implementation how to access and use the API.
The architecture is shown in figure 2.22

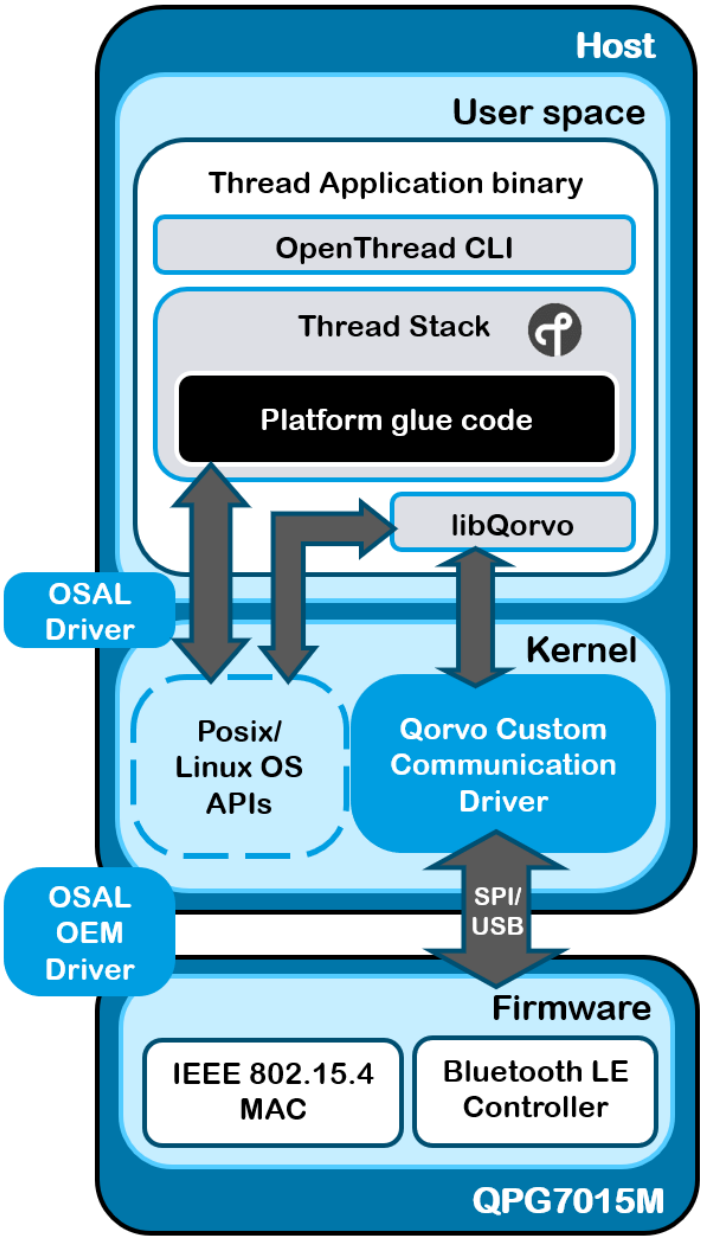


Figure 2.32: OpenThread CLI architecture

4 interesting blocks can be seen:

1. **OpenThread CLI** ⇒ A CLI application that showcases the OpenThread API functionality.
2. **Thread Stack** ⇒ Enables Thread functionality, maintained by the OpenThread community
3. **Platform Glue code** ⇒ Interface for OpenThread with platform specific functionalities such as alarms, storage, random number generator,
4. **libQorvo** ⇒ Implements platform specific functionality and hardware. It provides timers, random and radio APIs in a way the OpenThread platform glue code can use it. On the bottom side, it uses a Posix API to interact with OS and Custom Communication Driver via IOCTL to interact

with radio hardware.

Prior to running the Thread CLI application, two action items:

1. Configure Thread to be enabled in `qorvo_stack_config`: `QORVO_OT_CLI=1` (See subsection 2.1.4.)
2. Start the DK should as described in subsection 2.1.3.

Form a Thread network and ping a device

This example shows how to form a Thread network using the Thread CLI Application and ping a node in the network. The ping command can be used to generate network traffic and indicates packet loss during the duration. One DK will operate as a parent and one as a child.

1. Start the Thread Application on both QPG7015M Gateways

```
pi@[hostname_parent]:~$ OpenThread/qpg7015m-ot-cli-ftd.elf
```

```
pi@[hostname_child]:~$ OpenThread/qpg7015m-ot-cli-ftd.elf
```

2. Configure a PAN ID for the network on both devices.

```
panid 0xb1eb
Done
>
```

```
panid 0xb1eb
Done
>
```

3. Bring up the IPv6 interface on both devices.

```
ifconfig up
Done
>
```

```
ifconfig up
Done
>
```

4. Enable the Thread protocol on the parent so the device can form its own network.

```
thread start
Done
>
```

5. Start commissioning on the parent.

```
commissioner start
Done
>
```

6. Add a joiner to your network on your parent.

```
commissioner joiner add * J01NME
Done
>
```

7. Start joining on the child.

```
joiner start J01NME
Done
>
```

8. Verify the child had joined the parent's network.

```
> join success
```

```
> ~ Discovery Request from XXXXXXXXXXXXXXXX: version=X,joiner=X
Commissioner:  Joiner start XXXXXXXXXXXXXXXX
Commissioner:  Joiner connect XXXXXXXXXXXXXXXX
Commissioner:  Joiner finalize XXXXXXXXXXXXXXXX
Commissioner:  Joiner end XXXXXXXXXXXXXXXX
```

9. Start the Thread stack on the child.

```
thread start
Done
>
```

10. Retrieve the IP address from the child.

```
ipaddr
fdde:ad00:beef:0:0:ff:fe00:401
fdde:ad00:beef:0:ff74:b2bc:1f98:9a27
fe80:0:0:0:5c5c:b4ab:c490:ea02
Done
>
```

Several IPv6 addresses are printed:

- fdde:ad00:beef:0:0:ff:fe00:401 ⇒ Routing Locator
- fdde:ad00:beef:0:ff74:b2bc:1f98:9a27 ⇒ Mesh-Local Endpoint Identifiers
- fe80:0:0:0:5c5c:b4ab:c490:ea02 ⇒ Link-Local Address

11. Ping the child/router with the parent.
    - 32 ⇒ Payload
    - 10 ⇒ Number of packets
    - 0.1 ⇒ Interval

```
ping fdde:ad00:beef:0:0:ff:fe00:401 32 10 0.1 > 40 bytes from
fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=1 hlim=64 time=25ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=2 hlim=64 time=16ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=3 hlim=64 time=17ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=4 hlim=64 time=17ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=5 hlim=64 time=20ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=6 hlim=64 time=15ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=7 hlim=64 time=14ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=8 hlim=64 time=16ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=9 hlim=64 time=16ms
40 bytes from fdde:ad00:beef:0:0:ff:fe00:401:  icmp_seq=10 hlim=64 time=15ms
10 packets transmitted, 10 packets received.  Packet loss = 0.0%.  Round-trip
min/avg/max = 14/17.100/25 ms.
Done
```

12. Both devices can be reset.

```
factoryreset
Done
>
```

```
factoryreset
Done
>
```

13. Terminate application

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

Additional documentation

For more info on porting the OpenThread solution to your own platform see section 3.2. Additional documentation can be found on our GitHub page: https://github.com/Qorvo/QGateway/ For more architectural info, check out the QPG7015M setup instructions. More info on the OpenThread CLI application can be found on the OpenThread GIT Repo.

3. Porting the SDK to a custom application processor

The QPG7015M gateway can be ported to several Linux platforms. The main flow to get started:

1. Access the latest software and documentation.
2. Verify the toolchain is compliant to the QPG7015M SDK using the ToolchainTest.
3. Port the drivers to the platform.
4. Port the FirmwareUpdater to the platform.
5. Port the TestCom application to the platform.
6. Port the BoardConfigTool application to the platform.
7. Load the drivers and flash the firmware.
8. Verify Driver functionality.
9. Enable prefered listening mode.
10. Optional: Request cross compiled Zigbee libraries to Qorvo.
11. Port prefered IoT Application.
12. Start verification.

3.1 Access the latest software and documentation

To get access to the QPG7015M SDK, send a mail to lpw.support@qorvo.com, stating the mail addresses of the people that require access to the SDK. Upon approval, these receive a confirmation mail to the latest version of the source code and documentation of the deliverables.

When opening the link select the folder vX.X.X.X [X.X.X.X being the current latest version] and click the download button.

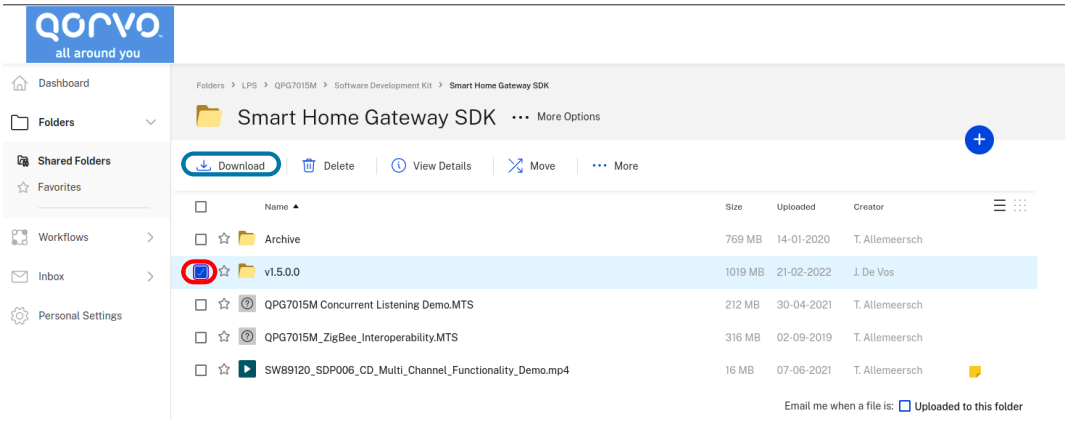


Figure 3.1: Sharefile access

Extract the zipped SDK, e.g. for v1.5.1.0:

```
pi@[hostname]:~/Downloads$ unzip v1.5.1.0.zip -d WORK_DIR && chmod 775 -R WORK_DIR
```

Resulting in following work directory that's created:

```
pi@[hostname]:~/Downloads$ tree WORK_DIR
WORK_DIR
+-- Bluetooth_LE
|   +-- GP_P1053_SW_17777_Bluetooth_LE_ACS_DPK.zip
+-- Drivers
|   +-- GP_P1053_SW_15249_Smart_Home_Gateway_Firmware_Updater.zip
|   +-- GP_P1053_SW_15657_Smart_Home_Gateway_Firmware_QPG7015M.zip
|   +-- GP_P1053_SW_17697_ComDriver_QPG7015M_RPi4.zip
+-- Misc
|   +-- GP_P1053_SW_15660_Smart_Home_Gateway_Reference_Scripts_QPG7015M.zip
|   +-- GP_P1053_SW_15849_ToolchainTest.zip
|   +-- GP_P1053_SW_15973_ComDumpTool.zip
|   +-- GP_P1053_SW_16292_HostnameGen.zip
|   +-- GP_P1053_SW_16799_BoardConfigTool.zip
|   +-- GP_P1053_SW_16844_CoexConfigTool.zip
+-- OpenThread
|   +-- README.txt
+-- Production_Test_and_RF_Validation_Tools
|   +-- GP_P1053_SW_17304_TestComponents_QPG7015M.zip
+-- README.txt
+-- SW08583_FirmwareUpdater_v1_5_1_0_RCP_patch.rar
+-- Smart_Home_Gateway_SDK_v1.5.1.0.img.7z
+-- ZigBee_3.0
.    +-- GP_P1053_SW_15247_Smart_Home_Gateway_Client_Libs.zip
.    +-- GP_P1053_SW_15641_Smart_Home_Gateway_ZB3.0.zip
.    +-- GP_P1053_SW_15643_Smart_Home_Gateway_ZB3.0_Libs.zip
.    +-- GP_P1053_SW_17645_Smart_Home_Gateway_Client.zip
```

```
6 directories, 19 files
```

The subfolders contain packages to enable the following functionality:

- **Bluetooth_LE** ⇒ Source code and documentation to develop and enable Bluetooth LE applications on the QPG7015M.
- **Drivers** ⇒ Kernel Drivers and Firmware tooling to enable communication between the QPG7015M and the host platform.
- **Misc** ⇒ Contains source code and documentation for configuration, scripting and software verification.
- **OpenThread** ⇒ Points to the QPG7015M its Thread and Matter protocol starting point GIT repository.
- **Production_Test_and_RF_Validation_Tools** ⇒ Source code and documentation for RF and PTA validation.
- **ZigBee_3.0** ⇒ Libraries, source code and documentation to develop and enable Zigbee applications on the QPG7015M.
- **Smart_Home_Gateway_SDK_v1.5.1.0.img.7z** ⇒ The zipped image used in the Qorvo® Iot Dev Kit Pr its QPG7015M gateway.

3.2 Port applications to the platform

The Host applications and kernel drivers can be ported to several Linux based platforms. This enables the customer to use the QPG7015M and its SDK on its own platform. There is a generic flow to port these applications with some additional steps to port the Zigbee and Thread applications.

3.2.1 Generic

The applications are built using the `make` command combined with a provided Makefile. By modifying this Makefile, the customer is able to build the application against its own toolchain.

Prerequisites

Before getting started, the following tools are required. Please refer to the manual of the used build environment to install the packages:

- git
- make

The Qorvo SDK is by default built using the arm-bcm2708hardfp-linux-gnueabi toolchain. To test with this toolchain, retrieve it from GIT:

```
user@[hostname]:~/WORK_DIR$ git clone https://github.com/raspberrypi/tools.git
Cloning into 'tools'...
remote:  Enumerating objects:  25415, done.
remote:  Counting objects:  100% (41/41), done.
remote:  Total 25415 (delta 23), reused 22 (delta 14), pack-reused 25374
Receiving objects:  100% (25415/25415), 610.89 MiB | 5.23 MiB/s, done.
```

```
Resolving deltas:  100% (14904/14904), done.
Updating files:  100% (19060/19060), done.
```

Cross compilation

To verify that a toolchain is compatible with our SDK, Qorvo provides a Toolchain Test application. This also includes the standardized flow used to cross compile an application.

1. Unzip the Toolchain Test

```
user@[hostname]:~/WORK_DIR$ cd Misc/
user@[hostname]:~/WORK_DIR/Misc$ unzip GP_P1053_SW_15849_ToolchainTest.zip -d
GP_P1053_SW_15849_ToolchainTest && chmod 775 -R GP_P1053_SW_15849_ToolchainTest

Archive:  GP_P1053_SW_15849_ToolchainTest.zip
creating:  GP_P1053_SW_15849_ToolchainTest/Documents/
creating:  GP_P1053_SW_15849_ToolchainTest/Documents/APIDocumentation/
inflating:  GP_P1053_SW_15849_ToolchainTest/Documents/APIDocumentation/
ReleaseNotes.pdf
creating:  GP_P1053_SW_15849_ToolchainTest/Documents/ApplicationNotes/
inflating:  GP_P1053_SW_15849_ToolchainTest/Documents/ApplicationNotes/
GP_P1053_AN_16709_GNU_Debugger.pdf
creating:  GP_P1053_SW_15849_ToolchainTest/Documents/ReleaseNotes/
inflating:  GP_P1053_SW_15849_ToolchainTest/Documents/ReleaseNotes/
ReleaseNotes.pdf
inflating:  GP_P1053_SW_15849_ToolchainTest/Documents/document_overview.txt
inflating:  GP_P1053_SW_15849_ToolchainTest/End_User_License_Agreement.pdf
inflating:  GP_P1053_SW_15849_ToolchainTest/GP_P866_RN_16017_ToolchainTest.pdf
inflating:  GP_P1053_SW_15849_ToolchainTest/README.txt
creating:  GP_P1053_SW_15849_ToolchainTest/Software/
extracting:  GP_P1053_SW_15849_ToolchainTest/Software/
ToolchainTest_198093_v1.5.1.0.tgz
inflating:  GP_P1053_SW_15849_ToolchainTest/Software/package_defs.sh
inflating:  GP_P1053_SW_15849_ToolchainTest/Software/build.sh

user@[hostname]:~/WORK_DIR/Misc$ cd GP_P1053_SW_15849_ToolchainTest/Software/
user@[hostname]:~/WORK_DIR/Misc/GP_P1053_SW_15849_ToolchainTest/Software$ tar
xvf ToolchainTest_198093_v1.5.1.0.tgz

ToolchainTest/
ToolchainTest/code/
ToolchainTest/code/Applications/
ToolchainTest/code/Applications/P866_Ubisys_ZPRO/
ToolchainTest/code/Applications/P866_Ubisys_ZPRO/apps/
ToolchainTest/code/Applications/P866_Ubisys_ZPRO/apps/ToolchainTest/
ToolchainTest/code/Applications/P866_Ubisys_ZPRO/apps/ToolchainTest/
check_toolversions.mk
ToolchainTest/code/Applications/P866_Ubisys_ZPRO/apps/ToolchainTest/src/
ToolchainTest/code/Applications/P866_Ubisys_ZPRO/apps/ToolchainTest/src/Main.cpp
ToolchainTest/code/Work/
ToolchainTest/code/Work/Applications/
```

```
ToolchainTest/code/Work/Applications/ToolchainTest_RPi/
ToolchainTest/code/Work/Applications/ToolchainTest_RPi/gen/
ToolchainTest/code/Work/Applications/ToolchainTest_RPi/gen/qorvo_config.h
ToolchainTest/code/Work/Applications/ToolchainTest_RPi/gen/qorvo_internals.h
ToolchainTest/overview.ToolchainTest_RPi.txt
ToolchainTest/make/
ToolchainTest/make/gpcommon.mk
ToolchainTest/make/compilers/
ToolchainTest/make/compilers/armrpi_gnueabihf/
ToolchainTest/make/compilers/armrpi_gnueabihf/compiler_defines.mk
ToolchainTest/make/compilers/bcmcommon/
ToolchainTest/make/compilers/bcmcommon/compiler_defines.mk
ToolchainTest/Makefile.ToolchainTest_RPi
user@[hostname]:~/WORK_DIR/Misc/GP_P1053_SW_15849_ToolchainTest/Software$ cd
ToolchainTest
user@[hostname]:~/PATH_TO/ToolchainTest$
```

2. Define the following parameters
   - **COMPILER** ⇒ Name of the compiler, defines the path to the Compiler specific make file.
   - **TOOLCHAIN** ⇒ Path to your toolchain
   - **ARCH** ⇒ Defines the architecture of your platform
   - **CROSS_COMPILE** ⇒ Defines what compiler to pick from the toolchain binary directory
   - **SYSTEMROOT** ⇒ Directory that contains the platform its headers and libraries to compile/link against.

   E.g. for arm-bcm2708hardfp-linux-gnueabi:

```
user@[hostname]:~/PATH_TO/ToolchainTest$ export COMPILER=rpi_bcm2708
user@[hostname]:~/PATH_TO/ToolchainTest$ export TOOLCHAIN=
/home/$USER/WORK_DIR/tools/arm-bcm2708/arm-bcm2708hardfp-linux-gnueabi
user@[hostname]:~/PATH_TO/ToolchainTest$ export ARCH=arm
user@[hostname]:~/PATH_TO/ToolchainTest$
export CROSS_COMPILE=arm-bcm2708hardfp-linux-gnueabi-
user@[hostname]:~/PATH_TO/ToolchainTest$ export SYSTEMROOT=
/home/$USER/WORK_DIR/tools/arm-bcm2708/arm-bcm2708hardfp-linux-gnueabi/
arm-bcm2708hardfp-linux-gnueabi/sysroot
```

3. Create an application Makefile based on the Makefile provided by Qorvo

```
user@[hostname]:~/PATH_TO/ToolchainTest$ cp Makefile.ToolchainTest_RPi
Makefile.ToolchainTest_$COMPILER
```

4. (OPTIONAL) Edit your Application Makefile: Makefile.ToolchainTest_$COMPILER (e.g. the application name)
5. Create a compiler specific Makefile based on the Makefile provided by Qorvo

```
user@[hostname]:~/PATH_TO/ToolchainTest$ cp -R make/compilers/armrpi_gnueabihf
make/compilers/$COMPILER
```

6. (OPTIONAL) Edit your Compiler specific Makefile in make/compilers/$COMPILER
7. Build the binary using the `make` command

```
user@[hostname]:~/PATH_TO/ToolchainTest$ make -f Makefile.ToolchainTest_$COMPILER
PATH_TO/ToolchainTest/make/compilers/rpi_bcm2708/compiler_defines.mk:46:
Filtering out compiler flags:  -march=armv6j
PATH_TO/ToolchainTest/make/compilers/rpi_bcm2708/compiler_defines.mk:46:
Filtering out compiler flags:  -march=armv6j
make[1]:  warning:  -j4 forced in makefile:  resetting jobserver mode.
Building Main.o
Linking ToolchainTest_RPi.elf
```

8. Copy the binary to your platform

```
user@[hostname]:~/PATH_TO/ToolchainTest/$
scp Work/ToolchainTest_RPi/ToolchainTest_RPi.elf user@host_platform:.
```

9. Verify it passes on your platform.

```
user@[hostname]:~$ ./ToolchainTest_RPi.elf
All tests passed.
```

3.2.2 Thread

A GIT repository is created for the Thread deliverables. Following link explains how to build the Thread applications:
https://github.com/Qorvo/QGateway/blob/vlatest/Documents/Guides/user_manual_ot_qpg7015M.md

! For Makefile operations, the flow is the same as described in section 3.2.1.

3.2.3 Zigbee

The generic QPG7015M gateway contains all Zigbee deliverables for the Raspberry Pi 4 based configuration. Since the Zigbee stack is provided as a library the customer must request a cross compiled deliverable to Qorvo so it can be ported to the customer's platform. When the ToolchainTest is successfull, see section , the customer should reach out to lpw.support@qorvo.com with the request for the Zigbee deliverables and should forwards its toolchain. For the Zigbee deliverable, 4 custom packets will be provided:

- GP_P1053_SW_17645_Smart_Home_Gateway_Client.zip
  ⇒ The source code and Makefile to build the Smarthome Gateway Client application
- GP_P1053_SW_15247_Smart_Home_Gateway_Client_Libs.zip
  ⇒ Contains static libraries, required to build the SmartHome Gateway Client application, these should be added in to the Smarthome Gateway Client Application package.

- GP_P1053_SW_15641_Smart_Home_Gateway_ZB3.0.zip
  ⇒ Contains the daemons accessed by the SmartHome Gateway Client application and standalone applications.
- GP_P1053_SW_15643_Smart_Home_Gateway_ZB3.0_Libs.zip
  ⇒ Contains dynamic libraries accessed by the binaries. The directory to these libraries should be stored in the environment variable `LD_LIBRARY_PATH`.

Building the Smarthome Gateway Client application

After receiving the ported version of the four Zigbee deliverables:

1. Extract GP_P1053_SW_17645_Smart_Home_Gateway_Client.zip and GP_P1053_SW_15247_Smart_Home_Gateway_Client_Libs.zip

```
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
unzip GP_P1053_SW_17645_Smart_Home_Gateway_Client.zip -d
GP_P1053_SW_17645_Smart_Home_Gateway_Client && chmod 775 -R
GP_P1053_SW_17645_Smart_Home_Gateway_Client
...
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
cd GP_P1053_SW_17645_Smart_Home_Gateway_Client/Software/
user@[hostname]:~/PATH_TO/Software$
tar xvf SmartHomeGatewayClient_198093_v1.5.1.0.tgz
...
user@[hostname]:~/PATH_TO/Software$ cd /home/$USER/WORK_DIR/ZigBee_3.0
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
unzip GP_P1053_SW_15247_Smart_Home_Gateway_Client_Libs.zip -d
GP_P1053_SW_15247_Smart_Home_Gateway_Client_Libs && chmod 775 -R
GP_P1053_SW_15247_Smart_Home_Gateway_Client_Libs
...
```

2. Copy the custom libraries to the Smarthome Gateway Client application

```
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
cp -a GP_P1053_SW_15247_Smart_Home_Gateway_Client_Libs/Software/.
/home/$USER/WORK_DIR/ZigBee_3.0/GP_P1053_SW_17645_Smart_Home_Gateway_Client/
Software/SmartHomeGatewayClient/code/Components/ThirdParty/
```

3. Build the Zigbee Smart Home Gateway Client Application, the same way it was done for the Toolchain Test section 3.2.1.

! For the toolchain mentioned in section 3.2.1 a flag needed to be added:

```
+ FLAGS+=-DHAVE_NO_STRTOLL
```

Linking will fail when the default libraries are used with a custom toolchain.

Running the Smarthome Gateway Client application

! Two remarks:

- To enable Zigbee traffic, the Host must be connected to the QPG7015M. This would require

a hardware development to onboard the QPG7015M. Reach out to lpw.support@qorvo.com for more information.

- Be sure you ran through steps 1-9 of the porting guide before continuing.

Figure 3.2 shows that multiple binaries are involved to get the Zigbee application up and running.

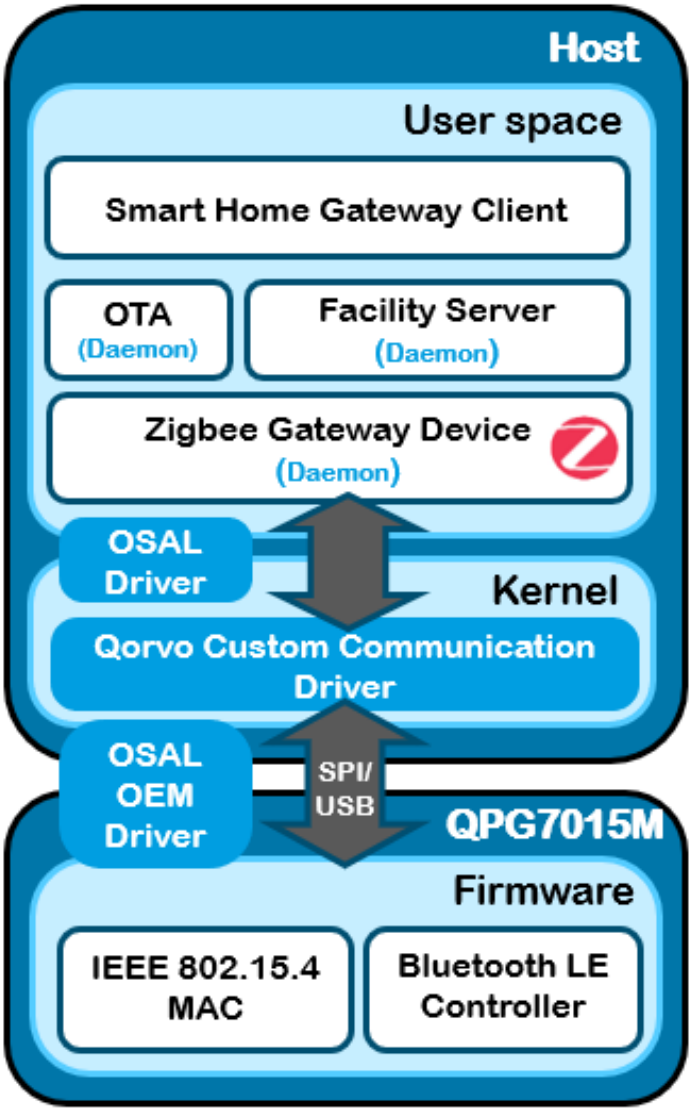


Figure 3.2: Zigbee 3.0 architecture

Following steps must be taken to run all the required processes:

1. Extract the dynamic libraries:

```
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
unzip GP_P1053_SW_15643_Smart_Home_Gateway_ZB3.0_Libs.zip -d
GP_P1053_SW_15643_Smart_Home_Gateway_ZB3.0_Libs && chmod 775 -R
GP_P1053_SW_15643_Smart_Home_Gateway_ZB3.0_Libs
```

```
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
cd GP_P1053_SW_15643_Smart_Home_Gateway_ZB3.0_Libs/Software && tar xvf
ZigBee3.0_Libs_c61823e_2022-02-08_v1.3.8.0_armv6-rpi-linux-gnueabihf-4.9.3.tar.gz
```

2. Copy the ZigBee3.0 directory to your platform:

```
user@[hostname]:~/PATH_TO/Software$ scp -r ZigBee3.0 user@hostname:.
```

3. Extract the daemons:

```
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
unzip GP_P1053_SW_15641_Smart_Home_Gateway_ZB3.0.zip -d
GP_P1053_SW_15641_Smart_Home_Gateway_ZB3.0 && chmod 775 -R
GP_P1053_SW_15641_Smart_Home_Gateway_ZB3.0
user@[hostname]:~/WORK_DIR/ZigBee_3.0$
cd GP_P1053_SW_15641_Smart_Home_Gateway_ZB3.0/Software && tar xvf
ZigBee3.0_c61823e_2022-02-08_v1.3.8.0_armv6-rpi-linux-gnueabihf-4.9.3.tar.gz
```

4. Copy the ZigBee3.0 directory to your platform, it will merge the content to the directory of the dynamic libraries:

```
user@[hostname]:~/PATH_TO/Software$ scp -r ZigBee3.0 user@hostname:.
```

5. Copy the SmartHome Gateway Client application, cross compiled in section 3.2.3 to the ZigBee3.0 directory on your platform.

```
user@[hostname]:~/PATH_TO/SmartHomeGatewayClient/Work/SmartHomeGatewayClient_RPi$
scp user@host:./ZigBee3.0/SmartHomeGatewayClient_RPi.elf
```

6. On your platform, set the linked libraries

```
user@[hostname_platform]:~$ export LD_LIBRARY_PATH=/home/$USER/ZigBee3.0/lib
```

7. Create directories for binaries

```
user@[hostname_platform]:~$ sudo mkdir -p   /etc/otad/
user@[hostname_platform]:~$ sudo chmod a+rw /etc/otad/
user@[hostname_platform]:~$ sudo mkdir -p   /etc/facilityd/
user@[hostname_platform]:~$ sudo chmod a+rw /etc/facilityd/
user@[hostname_platform]:~$ sudo mkdir -p   /etc/zgdd/
user@[hostname_platform]:~$ sudo chmod a+rw /etc/zgdd/
user@[hostname_platform]:~$ sudo mkdir -p   /etc/mac/
user@[hostname_platform]:~$ sudo chmod a+rw /etc/mac/
user@[hostname_platform]:~$ sudo mkdir -p   /dev/socket/
user@[hostname_platform]:~$ sudo chmod a+rw /dev/socket/
user@[hostname_platform]:~$ sudo mkdir -p   /etc/data/ota/firmware
user@[hostname_platform]:~$ sudo chmod a+rw /etc/data/ota/firmware
```

```
user@[hostname_platform]:~$ sudo mkdir -p   /etc/data/ota/tmp
user@[hostname_platform]:~$ sudo chmod a+rw /etc/data/ota/tmp
user@[hostname_platform]:~$ sudo mkdir -p   /system/etc/facilityd
user@[hostname_platform]:~$ sudo chmod a+rw /system/etc/facilityd
```

8. Enable IPv6 in the kernel

```
user@[hostname_platform]:~$ modprobe ipv6
```

9. Copy the default delivered settings

```
user@[hostname_platform]:~$ cp ZigBee3.0/etc/facilityd/settings_ubi.xml
/system/etc/facilityd/settings.xml
```

10. Copy the factory reset version of the Zigbee stack binary

```
user@[hostname_platform]:~$ cp ZigBee3.0/etc/facilityd/c7b.psb.factory.bin
/etc/facilityd/c7b.psb.bin
user@[hostname_platform]:~$ cp ZigBee3.0/c7b.psb.bin /etc/zgdd/
```

11. Make a FIFO special file to communicate with the Zigbee Gateway Device Daemon

```
user@[hostname_platform]:~$ mkfifo /tmp/zgdd_input
```

12. Start the Zigbee Gateway Device Daemon with its arguments and map it to the FIFO special file.

```
user@[hostname_platform]:~$ cd ZigBee3.0 && ./zgdd --etc=/etc/zgdd
--c7bgp-filter=0 --c7b-factory=./ --c7b-working=/etc/zgdd
--c15dot4-scan-remap=0#131 --c7b-acceptable-energy-level=192 <>/tmp/zgdd_input
&
```

Possible arguments:

- **--etc=<path>** ⇒ Path to factory block and NVM
- **--c7bgp-filter=<0/1>** ⇒ Enable GreenPower filter
- **--c7b-factory=<path>** ⇒ Factory fresh NVM
- **--c7b-working=<path>** ⇒ Currently used NVM
- **--c15dot4-scan-remap=<A#B>** ⇒ Remap a scan-type A between [0, 9] to a custom scan-type value B. Custom types:
  - 128 ⇒ (Wifi Aware) RSSI as a measure of the energy.
  - 129 ⇒ (Wifi Aware) RSSI Masked as a measure of the energy.)
  - 130 ⇒ Absolute Interference as a measure of the energy
  - 131 ⇒ Absolute Interference Masked as a measure of the energy [**RECOMMENDED**]

- **--c7b-acceptable-energy-level** ⇒ Threshold for channel selection during energy detection scan.

Wait for the ZGDD to be up and running. By default we take a **10s sleep time** before running the next command.

13. Copy the start configuration to maintain the original settings

```
user@[hostname_platform]:~$ cp start.xml tmp_start.xml
```

This start.xml file contains a channel mask, that can be editted if a different channel pick is preferred for the Zigbee protocol:

```
user@[hostname_platform]:~/PATH_TO/ZigBee3.0$ cat start.xml
<?xml version="1.0" encoding="utf-8"?>
<commissioning>
<!-- Keep current device setting - Selecting channels 11, 15, 20, 25 -->
<startup device-type="0x01" channel-mask="0x02108800"/>
</commissioning>
```

E.g: 0x02108800 ⇒ 00100001000010001000000000000
So following channels are picked by default: 25, 20, 15, 11.

14. Run the statup application to configure the ZGD daemon and enable peer connections over a given port.
Possible arguments:

```
user@[hostname_platform]:~$ cd ZigBee3.0 && ./zstartup --port=17755
tmp_start.xml
```

- **--port=<port>** ⇒ Port number for accepting peer connection request [default: 17755]

Wait for the start up application to be up and running. By default we take a **15s sleep time** before running the next command.

15. Start the facility daemon

```
user@[hostname_platform]:~/PATH_TO/ZigBee3.0$ ./facilityd --etc=/etc/facilityd
--settings=/system/etc/facilityd/settings.xml &
```

- **--etc=<path>** ⇒ Path to factory block and NVM
- **--settings=<path>** ⇒ Path to settings.xml, this is a file that enables analysing the attribute values.

16. Start the OTA daemon

```
user@[hostname_platform]:~/PATH_TO/ZigBee3.0$ ./otad -etc=/etc/otad/
-images=/etc/data/ota/firmware &
```

- **--etc=<path>** ⇒ Path to factory block and NVM

- **--images=<path>** ⇒ Path to the new firmware image

17. Open a 30s window with custom PIN code for the facility client to join the facility server.

```
user@[hostname_platform]:~/PATH_TO/ZigBee3.0$ ./facility-manage enable 1234 300
```

18. Start the Zigbee Smarthome Gateway Client application

```
user@[hostname_platform]:~/PATH_TO/ZigBee3.0$./SmartHomeGatewayClient_RPi.elf
connect --host=localhost --port=8888
```

3.3 Port the drivers to the platform

The QPG7015M supports two possible protocols to enable communication between the Host and the QPG7015M:

- SPI
- USB

To do so, the kernel driver must be ported to the platform using their configured Device Tree. Three Kernel drivers are defined:

- **DrvComKernel_QPG7015M_{SPI/USB}_RPi4.ko**

  ⇒ Communication Driver
  ⇒ Allows data relay and routing between userspace applications and the QPG7015M.

  ! The Operating System Abstraction Layer (OSAL) drivers are under GPL license.

- **OsalDriver_RPi4.ko**

  ⇒ Operating System Abstraction Layer Driver
  ⇒ Platform independent drivers
  ⇒ Foresees common functionality across platforms, OS specific, such as threads, events, logging,...

- **OsalOemDriver_QPG7015M_{SPI/USB}_RPi4.ko**

  ⇒ Operating System Abstraction Layer Original Equipment Manufacturer Driver
  ⇒ USB/SPI/GPIO Driver for the QPG7015M.
  ⇒ Platform specific because of the pin mapping dependency.

Following steps explain how to port the kernel driver to the RPi4 its used kernel. These steps can be applied to the customer its own kernel.

3.3.1 Generic

The QPG7015M requires at least 4 GPIO lines that must be configured in the Device Tree, independent of the chosen communication protocol:

1. MCU_INTOUTN
   ⇒ Interrupt to the host processor

2. RESETN
   ⇒ Resets the QPG7015M
3. PROG_EN
   ⇒ Programming Enable pin ⇒ Used during production and RFA. Advised to connect to TP for RFA, otherwise do not connect.
4. WKUP
   ⇒ Optional when SPI is used ⇒ Do not connect if WKUP signal is not used.

These must be configured in the Device Tree Source files of the kernel for the SoC. The Device Tree deliverables can be found in the Drivers directory:

```
user@[hostname]:~/WORK_DIR/linux$ cd ../Drivers/
```

```
user@[hostname]:~/WORK_DIR/Drivers$
unzip GP_P1053_SW_17697_ComDriver_QPG7015M_RPi4.zip
-d GP_P1053_SW_17697_ComDriver_QPG7015M_RPi4 && chmod 775 -R
GP_P1053_SW_17697_ComDriver_QPG7015M_RPi4
```

Additional documentation for porting the Device Tree deliverables can be found in:

GP_P1053_SW_17697_ComDriver_QPG7015M_RPi4

- GP_P1053_RN_15744_Communication_Driver.pdf
  ⇒ Release notes
- GP_P1053_AN_16579_Communication_Kernel_Driver_SPI.pdf
  ⇒ Porting the Communication driver for SPI ⇒ Porting the OsalDrivers without using the Device Tree
- GP_P1053_AN_16579_Communication_Kernel_Driver_USB.pdf
  ⇒ Porting the Communication driver for USB ⇒ Porting the OsalDrivers without using the Device Tree

3.4 Verification on the custom platform

When the porting is compleet the user can verify the functionality

The section "Preparation" in 3.4.1 should always be executed at boot of the system.

The section "ConcurrentConnnect™ Technology configuration" in 3.4.3 should only be executed IoT functionality is wanted.

3.4.1 Preparation

Following steps should always be executed at boot:

1. Load all the cross compiled kernel drivers

```
pi@[hostname]:~$ sudo insmod OsalDriver_RPi4.ko
```

```
pi@[hostname]:~$ sudo insmod OsalOemDriver_QPG7015M_SPI/USB_RPi4.ko
```

```
pi@[hostname]:~$ sudo insmod DrvComKernel_QPG7015M_SPI/USB_RPI4.ko
```

2. Load the Firmwware, using the FirmwareUpdater

```
pi@[hostname]:~$ ./FirmwareUpdater_RPi.elf Firmware_QPG7015m.hex
15:47:40:273 01 ============================
15:47:40:273 01 FirmwareUpdater
15:47:40:273 01 vX.X.X.X Change:XXXXXX
15:47:40:273 01 ============================
15:47:40:273 01 loading FirmwareUpdater/Firmware_QPG7015M.hex
15:47:40:424 01 INFO: Bootloader reports ready and valid
15:47:40:425 01 Bootloader is active
15:47:40:425 01 Bootloader Stage 2:  vX.X.X.X Change:XXXXXX
15:47:40:426 01 Bootloader Stage1:  version data unavailable (request not
supported by this stage2 bootloader)
15:47:40:426 01 ProductId:  QPG7015
15:47:40:426 01 IEEE 15.4 MAC: XX:XX:XX:XX:XX:XX
15:47:40:426 01 BLE MAC: XX:XX:XX:XX:XX:XX
15:47:40:437 01 Verifying CRC...
15:47:41:041 01 image matches
15:47:41:041 01 Starting application...
15:47:46:069 01 Received cbDeviceReady from firmware
15:47:46:070 01 Firmware update SUCCEEDED vX.X.X.X Change:XXXXXX
Firmware update finished successfully (0) ...
```

3.4.2 Verification of kernel drivers

Using the TestCom application, the customer its communication roundtrip and throughput between the host and the QPG7015M is verified.

Following steps are required to verify:

1. Cross compile the TestCom application from the ComDriver package. Cross compilation is described in section 3.2.1.
2. Load the drivers and the firmware, as explained in section 3.4.1.
3. Copy TestCom.elf to the platform
4. Execute TestCom.elf

```
pi@[hostname]:~$ ./TestCom.elf
14:02:50:109 01 ============================
14:02:50:109 01 TestCom
14:02:50:109 01 vX.X.X.X Change:XXXXXX
14:02:50:109 01 ============================
14:02:50:110 01 Test criteria summary:
14:02:50:110 01 Number of iterations (N): 10000
```

```
14:02:50:110 01 Minimum throughput roundtrip N*(req+ack+ind) 0 B/s
14:02:50:110 01 Minimum throughput burst RCP -> host (req+ack+ N*ind) 0 B/s
14:02:50:110 01 Test payload range:  [234 bytes -> 234 bytes]
14:02:50:110 01 Application_teststep_roundtrip_start:  start, 234 bytes
payload...
14:03:20:820 01 PASS: roundtrip bytes per second:  164953 (min 0) (average
including time between requests:  155566) length 234
14:03:20:820 01 Application_teststep_int2ext_start:  start, 234 bytes
payload...
14:03:33:170 01 PASS: int2ext throughput bytes per second:  191894 (min 0)
length:  234
14:03:33:170 01 Success Counter:  10000
14:03:33:171 01 Test FINISHED »»» Pass
```

3.4.3 ConcurrentConnnect™ Technology configuration

The QPG7015M its ConcurrentConnect™ Technology makes it a key differentiator in IoT sollutions. Qorvo defines four technologies to maximize its throughput:

1. **ConcurrentConnect™ Multi-Radio capability**
   ⇒ Single physical radio solution for conncurrent Thread/Matter protocol + Zigbee + Bluetooth LE use cases.
   ⇒ E.g. Bluetooth LE Central device combined with Matter Router and Zigbee Coordinator.
   ⇒ All IEEE802.15.4 Stacks (Matter protocol Thread and Zigbee) must operate on the same channel!

2. **ConcurrentConnect™ Multi-Channel capability**
   ⇒ Single physical radio solution for conncurrent Thread/Matter protocol + Zigbee use cases.
   ⇒ E.g. Zigbee Coordinator combined with Matter Router.
   ⇒ All IEEE802.15.4 Stacks (Matter protocol, Thread and Zigbee) can operate on different channels!
   ⇒ Can only be used when no Bluetooth LE Scanning is required concurrently.

3. **ConcurrentConnect™ Antenna Diversity**
   ⇒ Real time antenna selection, based on the best communication link for every packet received.

4. **ConcurrentConnect™ Coexistence**
   ⇒ Improved PTA functionality compared to industry standards
   ⇒ Preamble detects and MAC layer filtering allows faster medium requests
   ⇒ Validation is described in section 2.2.4

Following matrix defines the best combinations:

Zigbee	Matter	Bluetooth LE		
		Off	On	
			Scan On	Scan Off
Off	Off	Off	Normal	Normal
Off	On	Normal	Multi-Radio capability	Normal
On	Off	Normal	Multi-Radio capability	Normal
On	On	Multi-Channel capability	Multi-Radio capability ⚠	Multi-Channel capability

⚠ Zigbee and Matter MUST operate on the same channel in this case

Figure 3.3: Recommended Concurrent Connect Technology

	Antenna Diversity	Coexistence
Multi-Radio capability	Not Compatible	Compatible
Multi-Channel capability	Compatible	Compatible
Normal	Compatible	Compatible

ⓘ Antenna Diversity and Coexistence can be combined

Figure 3.4: Concurrent Connect Technology Compatibility

! Make sure the Drivers are loaded and the FirmwareUpdater ran before configuring the ConcurrentConnect™ Technology!

Configure ConcurrentConnect™ Multi-Radio capability

```
pi@[hostname]:~$ ./BoardConfigTool_RPi.elf --enable-concurrentconnect 12:32:12:966
01 Settings applied 12:32:12:967 01 Enabling ConcurrentConnect mode
```

Configure ConcurrentConnect™ Mutli-Channel capability

```
pi@[hostname]:~$ ./BoardConfigTool_RPi.elf --enable-multichannel 12:32:12:966 01
Settings applied 12:32:12:967 01 Enabling multichannel mode
```

Configure normal listening mode

```
pi@[hostname]:~$ ./BoardConfigTool_RPi.elf --enable-normal 12:32:12:966 01 Settings
applied 12:32:12:967 01 Enabling normal mode
```

Configure ConcurrentConnect™ Antenna Diversity

Several options are possible:

1. Configure Antenna 0 as the default antenna

```
pi@[hostname]:~$ ./BoardConfigTool_RPi.elf --set-antenna 0
12:26:02:429 01 Settings applied
12:26:02:432 01 Be sure the IEEE802.15.4 stacks are started before configuring
the antenna
12:26:02:433 01 Antenna configured succesfully to setting 0
```

2. Configure Antenna 1 as the default antenna

```
pi@[hostname]:~$ ./BoardConfigTool_RPi.elf --set-antenna 1
12:26:02:429 01 Settings applied
12:26:02:432 01 Be sure the IEEE802.15.4 stacks are started before configuring
the antenna
12:26:02:433 01 Antenna configured succesfully to setting 1
```

3. Configure Antenna Diversity

```
pi@[hostname]:~$ ./BoardConfigTool_RPi.elf --set-antenna 2
12:26:02:429 01 Settings applied
12:26:02:432 01 Be sure the IEEE802.15.4 stacks are started before configuring
the antenna
12:26:02:433 01 Antenna configured succesfully to setting 2
```

Configure ConcurrentConnect™ Coexistence

More information on Coexistence can be found in section 2.2.4, which explains the pin mapping, what application to run for configuration and what additional documentation to read first.

3.4.4 Verification of user applications

After the preparations from section 3.4.1, the customer is able to use the user application as described in section 2.2.

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
FCC	Federal Communications Commission
FIFO	First In First Out
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GPIO	General-Purpose Input/Output
GPL	GNU General Public License
HCI	Host Controller Interface
HDMI	High-Definition Multimedia Interface
HW	Hardware
IOCTL	I/O Control
IoT	Internet-of-Things
IPHA	TCP/IP based host applications
L2CAP	Logical Link Control and Adaptation Protocol
LED	Light Emitting Diode
NVM	None volatile Memory
OEM	Original Equipment Manufacturer
OS	Operating System
OSAL	Operating System Abstraction Layer
OT-BR	OpenThread Border Router
OTA	Over-The-Air
PAN	Personal Area Network
PC	Personal Computer
PIN	Personal Identification Number
PTA	Packet Traffic Arbitration
PTC	Production Test Component
RCC	Radio Control Console
RCP	Radio Co-Processor
RF	Radio Frequency
RFA	Return Field Analysis
RPi	Raspberry Pi
SDK	Software Development Kit
SMP	Secure Manager Protocol
SPI	Serial Peripheral Interface
SSH	Secure SHell
SW	Software
TP	Test Point
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
UUID	Universally Unique IDentifier
WSF	Wireless Software Foundation
ZCL	Zigbee Cluster Library
ZDO	Zigbee Gateway Device
ZGD	Zigbee Gateway Device

Regulatory Information

FCC notice

This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product. This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter.

Important Notices

The information contained herein is believed to be reliable; however, Qorvo makes no warranties regarding the information contained herein and assumes no responsibility or liability whatsoever for the use of the information contained herein. All information contained herein is subject to change without notice. Customers should obtain and verify the latest relevant information before placing orders for Qorvo products. The information contained herein or any use of such information does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other intellectual property rights, whether with regard to such information itself or anything described by such information. THIS INFORMATION DOES NOT CONSTITUTE A WARRANTY WITH RESPECT TO THE PRODUCTS DESCRIBED HEREIN, AND QORVO HEREBY DISCLAIMS ANY AND ALL WARRANTIES WITH RESPECT TO SUCH PRODUCTS WHETHER EXPRESS OR IMPLIED BY LAW, COURSE OF DEALING, COURSE OF PERFORMANCE, USAGE OF TRADE OR OTHERWISE, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
Without limiting the generality of the foregoing, Qorvo products are not warranted or authorized for use as critical components in medical, life-saving, or life-sustaining applications, or other applications where a failure would reasonably be expected to cause severe personal injury or death.

Document Revision Information

Version	Date	Changes
1.40	15 September 2023	Add instructions to stop scanning before connecting
1.30	10 July 2023	Examples aligned with v1.5.1.0
1.20	13 March 2023	Add Scanner Filtering example
1.10	27 February 2023	Add FCC notice
1.00	20 June 2022	First draft