UWB Qorvo Tools guide

Qorvo

Release 1.0.2

Contents

1 UWB Qorvo Tools

Welcome to the 'UWB Qorvo Tools' (UQT) Reference Guide. This guide is designed to help you navigate and utilize the various tools provided by Qorvo for operating Ultra-Wideband (UWB) devices through the UCI (UWB Communication Interface).

> Warning: These tools are delivered by Qorvo as UCI examples on an as-is basis and are not optimized nor meant to be used as an end product.

1.1 Root Folder Organization

Here is the core structure and key components of the UQT library. Less critical files and directories have been omitted.

```
uwb-qorvo-tools
    ├── README.md                          <-- UQT Introduction
    ├── pyproject.toml                     <-- UQT Project configuration file.
    ├── lib                                <-- Library folder.
    │   ├── uqt-utils                      <-- Helpers for main scripts.
    │   └── uwb-uci                        <-- UCI transport layer libraries.
    └── scripts                            <-- Script folder divided into "chapters".
        ├── <functionality_1>              <-- A "chapter" represents functionality 1.
        │   ├── <example_1>                <-- Example 1 for functionality 1.
        │   │   ├── <example_1_main.py>    <-- Entrance point for example 1 for functionality 1.
        │   │   ├── <example_1_helper.py>  <-- Helper script for example 1 for functionality 1.
        │   │   ├── ...                    <-- Other scripts and assets needed for for example 1␣
↪for functionality 1.
        │   │   └── README.md              <-- Documentation for example 1 for functionality 1.
        │   ├── <example_2>                <-- Example 2 for functionality 1.
        │   ├── ...                        <-- Other examples for functionality 1.
        │   └── README.md                  <-- Generic documentation for functionality 1.
        ├── <functionality_2>              <-- A "chapter" represents functionality 2.
        └── ...                            <-- Other "chapters" represent further␣
↪functionalities.
```

1.2 Prerequisites

- Installed python 3.10
- Installed pip 21.3 or greater

1.2.1 Python

You may download Python 3.10 from official web page or install it using an appropriate package manager.

In Linux you may install python using apt:

```
sudo apt-get install python3.10
```

In Windows PowerShell you may install python using chocolatey:

```
choco install python310 -y
```

1.2.2 PIP upgrade

You may upgrade pip to a required version by executing:

```
pip install pip>=21.3 --upgrade
```

1.2.3 Other specific set-ups

1.2.3.1 Linux

Some scripts relies on PySide library. It seems that on Ubuntu 22.04, the PySide environment installed by pip is missing some extra library. If you got the below run-time error:

```
qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in "" even though it was found"
```

You must install the library below:

```
sudo apt install libxcb-xinerama0
```

1.3 Quick Start

1.3.1 Install UQT

After you have installed the prerequisites mentioned above, you can proceed with the installation of UQT. There are two types of installations: editable and non-editable. The non-editable installation is the default option and is recommended for most users. However, if you need to make modifications to the UQT codebase, you should use the editable installation.

1.3.1.1 Non-Editable Installation

To perform a non-editable installation of UQT, follow these steps:

1.3.1.1.1 Linux

```
cd uwb-qorvo-tools
python -m venv .venv
source .venv/bin/activate
pip install .
```

1.3.1.1.2 Windows PowerShell

```
cd uwb-qorvo-tools
python -m venv .venv
.\.venv\Scripts\activate.ps1
pip install .
```

1.3.1.2 Editable Installation

To perform an editable installation of UQT, follow these steps:

1.3.1.2.1 Linux

```
cd uwb-qorvo-tools
python -m venv .venv
source .venv/bin/activate
pip install -e .
pip install -e lib/uqt-utils/
pip install -e lib/uwb-uci/
```

1.3.1.2.2 Windows PowerShell

```
cd uwb-qorvo-tools
python -m venv .venv
.\.venv\Scripts\activate.ps1
pip install -e .
pip install -e lib/uqt-utils/
pip install -e lib/uwb-uci/
```

> **Note:** The order of package installation matters!. When performing an editable installation, it is important to install the packages in the following order: UQT, uqt-utils, and uwb-uci. This ensures that any changes made in the dependencies are properly reflected in the main project. When installing the main project in editable mode using `pip install -e .`, it is crucial to manually install the dependencies in the correct order to ensure that any changes made in the dependencies are properly reflected in the main project.

1.3.2 Executing programs

If UQT is installed within a virtual environment, you must activate the virtual environment before executing scripts. If UQT is installed globally, you can skip this step.

To activate the virtual environment, follow the instructions below:

Linux

```
cd uwb-qorvo-tools
source .venv/bin/activate
```

Windows PowerShell

```
cd uwb-qorvo-tools
.\.venv\Scripts\activate.ps1
```

> **Warning** On Windows recommened to use shell is PowerShell. Other shell like cmd, Git Bash, etc. may require activation of the virtual environment with other activation scripts.

After the virtual environment is activated you may use Python scripts as other regular Python scripts:

```
python /path/to/script1.py <arg1> <arg2> <...>
python /path/to/script2.py <arg1> <arg2> <...>
```

The second option is using entry points. In this case, an explicit call to `python` is not needed:

```
<entry_point1> <arg1> <arg2> <...>
<entry_point2> <arg1> <arg2> <...>
```

1.3.3 How to check all possible entry points

```
uqt_ls
-> <script_name_1>                          One line description of <script_name_1>
-> <script_name_2>                          One line description of <script_name_2>
-> <script_name_3>                          One line description of <script_name_3>
...
```

In order to get more information about a specific script run it with `-h` option or read `README.md` file in the appropriate subdirectory in `script` folder.

For example:

```
<script_name_1> -h
```

1.3.4 How to check current environment

```
uqt_info

    # Python3 version:
        ...
    # Customization (UQT_CUSTOM):
        ....
    # Wanted Extensions (UQT_ADDINS):
        addin_name_1
        addin_name_2
        addin_name_3
        ...
    # Loaded Extensions:
        ...
    # DUT 'Default':
        unknown port ('UQT_PORT' not defined.)
```

1.4 Manual: About communicating with a DUT

The DUT communication is done through the use of OS 'ports': `COMxx`, `/dev/ttyUSBxx`, `/dev/serial/xx`. The related transport protocol is UART.

1.4.1 Communication Ports

To find the appropriate port for your DUT, please:

Windows (PowerShell):

- Open the Device Manager.
- Expand `Ports (COM & LPT)`.
- Look for your DUT port (e.g. `USB Serial Port (COM5)`).
- Use the port number (e.g. `COM5`) to set the `UQT_PORT` variable.
- Set a default port:

```
$env:UQT_PORT="<port>"
```

- Verify the proper settings:

```
uqt_info
```

Linux:

- List the available ports using `ls /dev/serial/by-id/`.
- Look for the appropriate port (e.g. `/dev/serial/by-id/usb-Nordic_Semiconductor_nRF52_USB_Product_<XYZ>-if00`).
- Use the port name (e.g. `/dev/serial/by-id/usb-Nordic_Semiconductor_nRF52_USB_Product_<XYZ>-if00`) to set the UQT_PORT variable.
- Set a default port:

```
export UQT_PORT="<port>"
```

- Verify the proper settings:

```
uqt_info
```

Note: For UQT scripts that require specifying a communication port, the `--port` option is available. If this option is not explicitly provided, the scripts will by default use `UQT_PORT` environment variable as the communication port with the Device Under Test (DUT). The `--port` option can always be used to manually specify the port regardless of the `UQT_PORT` setting.

1.5 Manual: about configuring your DUT

1.5.1 How to get/set Device Configuration

```
# Get the list of available parameters:
    get_config -l
# Get the value of all available parameters:
    get_config all
# Set the default channel number to 5:
    set_config ChannelNumber 5
```

1.5.2 How to get/set Calibration Parameters

```
# Get the list of available parameters:
    get_cal -l
# Get the value of all cal parameters as binary stream:
    get_cal -b all
# Backup all your calibration values to a file:
    get_cal -fa all | tee my_calibration.txt
# Recover your calibration values from a file:
    set_cal -i my_calibration.txt
```

In order to get more information about a specific script and/or use case please read the subsequent chapter or `README.md` file in the appropriate subdirectory.

2 scripts

2.1 Scripts Introduction

The **scripts** directory serves as the core functional hub of UQT library, housing a collection of essential scripts designed to provide a wide array of functionalities. Whether you are looking to manage devices, run UWB use cases, or handle specific scenarios, you'll find the tools you need here.

Each subdirectory within this folder is dedicated to a specific area of functionality, ensuring that the scripts are well-organized and easy to navigate. Within this directory, you'll encounter scripts that communicate with devices via UCI, alongside standalone "offline" tools and UQT supporting scripts.

For detailed information on how to use these scripts, please refer to the subsequent chapters or the README.md files located within the appropriate subdirectories. These resources will guide you through the specific use cases and help you make the most out of the available tools.

2.2 device

2.2.1 Device Scripts Introduction

The **device** directory introduces general functionality which is not related with any specific feature, but with a device itself. It means that in order to invoke these scripts you need a connected device. These scripts will communicate with a device using UCI interface.

Example scripts from this chapter are getting some information from a device, like statistics, firmware version or operations with calibration (get/set).

To learn more about detailed use cases, please read the subsequent chapters or README.md files in the appropriate subdirectories.

2.2.2 get_cal

This script retrieves from the device current calibration values.

2.2.2.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication intervace
-l / –list	list available param names and their spec
-v / –verbose	prints additional debug information
-f / –format	Choose param display format. Available: r(repr), x(hex), b(bytes), or a(android param file). Default: natural

2.2.2.2 Example

```
get_cal -p <port> ant_set0.tx_power_control
```

2.2.3 get_cap

This script retrieves DUT's (Device Under Test) capabilities through FiRa UCI commands CORE_GET_CAPS_INFO_CMD/CORE_GET_CAPS_INFO_RSP.

2.2.3.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-v / –verbose	prints additional debug information

2.2.3.2 Example

```
get_cap -p <port>

# Get Device Capability Parameter

# Get Caps Info:
        status:              Ok (0x0)
        Caps:

[...]
```

2.2.4 get_config

This script retrieves the device configuration; it can be used to display firmware debug traces configuration.

2.2.4.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-l / –list	list available param names and their spec
-v / –verbose	prints additional debug information
-b / –as-bytes	show the key value as a byte stream
-x / –as-hex	show the key value in hex format
-r / –as-repr	show the key value in format used to set it back

2.2.4.2 Example

```
get_config -p <port>
State                = DeviceState.Ready

[...]
```

2.2.5 get_device_info

This script displays information about the device based on information retrieved from the `CORE_GET_DEVICE_INFO_RSP`.

2.2.5.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-v / –verbose	prints additional debug information

2.2.5.2 Response Fields

Below are the details of each field in the response:

Field	Length	Description
Status	1 Octet	Indicates the status of the response.
UCI Version	2 Octets	Supported UCI version.
MAC Version	2 Octets	Suppoted MAC version.
PHY Version	2 Octets	Suppoted PHY version.
UCI Extension Version	2 Octets	Suppoted UCI Extension version.
Vendor Specific Info Len	1 Octet	The length of the vendor-specific information.
Vendor Specific Info	n Octets	Vendor-specific information such as chip version and chip variant.

Note: For more details refer to `CORE_GET_DEVICE_INFO_RSP` in `FIRA UCI Technical Specification`.

2.2.5.2.1 Vendor Specific Info Fields

These fields contain vendor-specific information such as chip version and chip variant.

Field	Description
QMF Version	Qorvo Mobile Firmware (QMF) version.
OEM Version	Original equipment manufacturer (OEM) version, possible to set by customer.
Build Job	Information about the build job.
SOC ID	System on Chip (SOC) identifier.
Device ID	Device identifier.
Packaging ID	Packaging type: ‘sip’ (System in Package) or ‘soc’ (System on Chip).

Note: The fields `packaging id` and `build job` are optional and may not be present for each firmware or device.

2.2.5.3 Example

```
get_device_info -p <port>

Pinging device at COM18:
# Get Device Info:
    status:              Ok (0x0)

[...]
```

2.2.6 reset_calibration

This script resets the DUT's calibration parameters to their default values.

2.2.6.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-v / –verbose	prints additional debug information
–timeout	time in second until the script timeout

2.2.6.2 Example

```
reset_calibration -p <port> --timeout 6
```

2.2.7 reset_device

This script resets the DUT by sending FiRa UCI Command CORE_DEVICE_RESET_CMD.

2.2.7.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-v / –verbose	prints additional debug information

2.2.7.2 Example

```
reset_device -p <port>
```

2.2.8 set_cal

This script pushes a specific calibration value by adding the selected key and its value.

2.2.8.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-l / –list	list available param names and their spec
-v / –verbose	prints additional debug information
-f / –format	Choose param display format. Available: r(repr), x(hex), b(bytes), or a(android param file). Default: natural

2.2.8.2 Example

```
set_cal -p <port> ant_set0.tx_power_control 3

INFO:      setting ant_set0.tx_power_control
            ant_set0.tx_power_control = Int8(3)
                                      = 3
Ok (0)
```

2.2.9 set_config

This script sets a configuration parameter value; it can be used to activate firmware debug traces.

2.2.9.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-l / –list	list available param names and their spec
-v / –verbose	prints additional debug information

2.2.9.2 Example

```
set_config ChannelNumber 5 -p <port>
OK (0)
```

2.2.10 load_cal

2.2.10.1 calib_files

2.2.10.1.1 Calibration Files Introduction

JSON calibration files are stored in the directory `scripts/device/load_cal/calib_files`. These files enable the configuration of default calibration settings for various targets.

The calibration and configuration data are encapsulated within JSON files, which can be passed as arguments to the `load_cal` script. This script processes the JSON file and transmits the settings to the device via UCI for device configuration.

2.2.10.1.1.1 Directory Structure

```
calib_files
├── <target_1>                          <- Default calibrations for <target_1>
│   └── <antenna_1>.json                <- JSON calibration file for <target_1> and <antenna_1>
├── <target_2>                          <- Default calibrations for <target_2>
│   ├── <antenna_1>.json                <- JSON calibration file for <target_2> and <antenna_1>
│   └── <antenna_2>.json                <- JSON calibration file for <target_2> and <antenna_2>
└── ...
```

2.2.10.1.1.2 Customizing JSON Calibration Files

Default configuration JSON files serve as examples and can be customized to meet specific requirements. Calibration and configuration are managed through a Key/Value mechanism, reflected in the structure of these JSON files.

> **Note:** Detailed descriptions of available Calibration and Configuration keys are provided in a separate document.

Example JSON file:

```
{
  "LUT": {
    "PDOA_LUT_0_CH5" : [
      [-2.9021, -1.5708],
      ...
      [3.0487, 1.4661],
      [3.1142, 1.5708]
    ],

    "PDOA_LUT_1_CH9" : [
      [-3.3905, -1.5708],
      ...
      [2.8451, 1.4661],
      [2.8932, 1.5708]
    ]
  },

  "calibrations": {
    "pdoa_lut0.data": "PDOA_LUT_0_CH5",
    "pdoa_lut1.data": "PDOA_LUT_1_CH9",

    "ant0.transceiver": "0x00",
    "ant0.port": "0x01",
    "ant0.lna": "0",

    ...
  }
}
```

2.2.10.1.2 DWM3001CDK

This directory contains a default calibration and configuration JSON file for the DWM3001CDK kit. The development board integrates a UWB transceiver module based on the Qorvo DW3110 IC with an attached planar Dual-Hoe (WB007) antenna and nRF52833 MCU.

As the development kit does not support the AoA capabilities, only one default calibration file is available:

- **dual-hoe_non_aoa.json**: This file contains default calibration and configuration settings for the `DWM3001CDK` setup not supporting AoA.

2.2.10.1.3 QM33120WDK1

This directory contains default calibration and configuration JSON files for the `QM33120WDK1` kit.

Depending on the daughter board and antenna sets used, there are two files available:

- **jolie_aoa.json**: This file contains default calibration and configuration settings for the `QM33120WDK1` setup supporting AoA.

  Note: Compatible with the Nordic nRF52840 DK and a daughter board equipped with the QM33120W UWB transceiver featuring directional dual Jolie-AoA (JL359) antenna.

- **jolie_omni_non_aoa.json**: This file contains default calibration and configuration settings for the `QM33120WDK1` setup not supporting AoA.

  Note: Compatible with the Nordic nRF52840 DK and a daughter board equipped with the QM33110W UWB transceiver featuring omnidirectional single Jolie-Omni (JL159) antenna.

2.2.10.1.4 Type2AB EVB

This directory contains the default calibration and configuration JSON file for the Murata Type2AB EVB kit (Rev4.0). The evaluation board integrates a Murata Type2AB module based on the nRF52840 MCU and Qorvo QM33120W transceiver and utilizes two patch antennas.

As the development kit supports AoA capabilities, only one default calibration file is available:

- **patch_aoa.json**: This file contains the default calibration and configuration settings for the `Type2AB` setup supporting AoA.

  Warning: The provided JSON configuration file is compatible only with the latest Type2AB EVB revision - Rev4.0 (PCB version: JS-1055). **Revision 3.0 (PCB version: JS-0989) is not supported.**

2.3 fira

2.3.1 FiRa Introduction

The **fira** directory introduces device ranging capabilities according to FiRa standard.

In order to know detailed description of each tool please read appropriate subchapters below or README.md files in appropriate subfolders.

2.3.2 run_fira_test_per_rx

Script **run_fira_test_per_rx** is provided to demonstrate the FiRa PER RX Test. In the Packet/bit Error Rate RX Test mode, the UWBS continues to look for UWB packets in a timely fashion (T_GAP) until a configured number of packets (NUM_PACKETS) have elapsed. The timing of the packet starts only after the successful reception of the first packet, otherwise UWBS looks for the first packet indefinitely. This test mode can be used to measure the receiver sensitivity of the UWB device.

2.3.2.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-t / –time	Set the duration of the ranging session (in second); `-1` - range forever
-p / –port	Set communication port
-c / –channel	Refer to CHANNEL_NUMBER in `FIRA UCI Technical Specification`
-i / –input-file	Recover test configuration from a test profile file
-v / –verbose	Print additional debug information
–randomized-psdu	Refer to RANDOMIZE_PSDU in `FIRA UCI Extension for Testing` (default 0)
–preamble-code-index	Refer to PREAMBLE_CODE_INDEX in `FIRA UCI Technical Specification`
–sfd-id	Refer to SFD_ID in `FIRA UCI Technical Specification`
–rframe-config	Refer to RFRAME_CONFIG in `FIRA UCI Technical Specification` (default 0)
–psdu-data-rate	Refer to PSDU_DATA_RATE in `FIRA UCI Technical Specification`
–phr-data-rate	Refer to BPRF_PHR_DATA_RATE in `FIRA UCI Technical Specification`
–preamble-duration	Refer to PREAMBLE_DURATION in `FIRA UCI Technical Specification`
–sts-length	Refer to STS_LENGTH in `FIRA UCI Technical Specification`
–nb-sts-segments	Refer to NUMBER_OF_STS_SEGMENTS in `FIRA UCI Technical Specification`
–psdu	':' or '.' separated list of bytes
–num-packets	Refer to NUM_PACKETS in `FIRA UCI Extension for Testing` (default = 1000)
–t-gap	Refer to T_GAP in `FIRA UCI Extension for Testing` (default = 2000)
–timeout	Set timeout to receive the first package (in seconds). (default: 3)
–en-diag	Set the Qorvo ENABLE_DIAGNOSTIC parameter to 1
–antenna-set-id	Set the antenna set to use for the session. (default: 0)

2.3.2.2 Application

Refer to “Application Configuration Parameters” in `FIRA UCI Technical Specification` for the possible values on APP configuration parameters.

Parameter Name	Len (octet)	ID	Description	De-fault
NUM_PACKE	4	0x00	No. of packets	1000
T_GAP	4	0x01	Gap between start of one packet to the next in us T_GAP >> packet length	2000
T_START	4	0x02	Max. time from the start of T_GAP to SFD found state in us	450
T_WIN	4	0x03	Max. time for which RX is looking for a packet from the start of T_GAP in us T_WIN > T_START	750
RANDOM-IZE_PSDU	1	0x04	0 - No randomization 1 - Take first byte of data supplied by command and it shall be used as a seed for randomizing PSDU	0

The PER RX Test shall be triggered by using TEST_PER_RX_CMD command.

The TEST_PER_RX_CMD command shall be issued only after applying all required Configuration parameters and the “Device Test Mode” session SHALL be in SESSION_STATE_IDLE Session State, otherwise UWBS will respond TEST_PER_RX_RSP with Status of STATUS_ERROR_SESSION_NOT_CONFIGURED indicating that session is not configured.

The UWBS shall respond TEST_PER_RX_RSP with the status of STATUS_OK and start PER Test. The number of packets to be received over UWB is configured by the NUM_PACKETS APP Configuration Parameter. UWBS shall notify TEST_PER_RX_NTF notification after completing the PER Rx test.

Payload Field(s)	Size (octet)	Description
PSDU Data	N Octets	PSDU Data[0:N] bytes. 0 <= N <= 127 for BPRF, 0 <= N <= 4095 for HPRF.

Payload Field(s)	Size (octet)	Description
Status	1	Status code as per [FIRA_UCI_SPEC].

Payload Field(s)	Size (octet)	Description
Status	1	Notify host after receiving NUM_PACKETS. Refer generic status codes
ATTEMPTS	4	No. of RX attempts
ACQ_DETECT	4	No. of times signal was detected
ACQ_REJECT	4	No. of times signal was rejected
RX_FAIL	4	No. of times RX did not go beyond ACQ stage
SYNC_CIR_READY	4	No. of times sync CIR ready event was received
SFD_FAIL	4	No. of time RX was stuck at either ACQ detect or sync CIR ready
SFD_FOUND	4	No. of times SFD was found
PHR_DEC_ERROR	4	No. of times PHR decode failed
PHR_BIT_ERROR	4	No. of times PHR bits in error
PSDU_DEC_ERROR	4	No. of times payload decode failed
PSDU_BIT_ERROR	4	No. of times payload bits in error
STS_FOUND	4	No. of times STS detection was successful
EOF	4	No. of times end of frame event was triggered

2.3.3 run_fira_test_periodic_tx

Script **run_fira_test_periodic_tx** is provided to demonstrate FiRa Periodic TX Test. In the Periodic TX Test mode, the UWBS continues to send UWB packets until a configured number of packets (NUM_PACKETS) have been sent. This test mode may be used to measure characteristics like signal power, bandwidth and IEEE spectral mask of the transmitted signal.

2.3.3.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-t / –time	Set the duration of the ranging session (in seconds); `-1` - range forever
-p / –port	Specify communication interface
-c / –channel	Refer to CHANNEL_NUMBER in `FIRA UCI Technical Specification`
-i / –input-file	Recover test configuration from a test profile file
-v / –verbose	Prints additional debug information
–randomized-psdu	Refer to RANDOMIZE_PSDU in `FIRA UCI Extension for Testing` (default 0)
–preamble-code-index	Refer to PREAMBLE_CODE_INDEX in `FIRA UCI Technical Specification`
–sfd-id	Refer to SFD_ID in `FIRA UCI Technical Specification`
–rframe-config	Refer to RFRAME_CONFIG in `FIRA UCI Technical Specification` (default 0)
–psdu-data-rate	Refer to PSDU_DATA_RATE in `FIRA UCI Technical Specification`
–phr-data-rate	Refer to BPRF_PHR_DATA_RATE in `FIRA UCI Technical Specification`
–preamble-duration	Refer to PREAMBLE_DURATION in `FIRA UCI Technical Specification`
–nb-sts-segments	Refer to NUMBER_OF_STS_SEGMENTS in `FIRA UCI Technical Specification`
–sts-length	Refer to STS_LENGTH in `FIRA UCI Technical Specification`
–psdu	':' or '.' separated list of bytes
–num-packets	Refer to NUM_PACKETS in `FIRA UCI Extension for Testing` (default = 1000)
–t-gap	Refer to T_GAP in `FIRA UCI Extension for Testing` (default = 2000)
–timeout	Set timeout to receive the first package (in seconds). (default: 3)
–en-diag	Set the Qorvo ENABLE_DIAGNOSTIC parameter to 1
–antenna-set-id	Set the antenna set to use for the session. (default: 0)

2.3.3.2 Application

Refer to "Application Configuration Parameters" in `FIRA UCI Technical Specification` for the possible values on APP configuration parameter.

Parameter Name	Len (octet)	ID	Description	Default
NUM_PACKE	4	0x00	No. of packets	1000
T_GAP	4	0x01	Gap between start of one packet to the next in us.	2000
RANDOM-IZE_PSDU	1	0x04	0 - No randomization 1 - Take first byte of data supplied by command and it shall be used as a seed for randomizing PSDU	0

The periodic TX Test shall be triggered by using TEST_PERIODIC_TX_CMD command.

The TEST_PERIODIC_TX_CMD command SHALL be issued only after applying all required configuration parameters and the "Device Test Mode" session shall be in SESSION_STATE_IDLE Session State, otherwise UWBS SHALL respond TEST_PERIODIC_TX_RSP with Status of STATUS_ERROR_SESSION_NOT_CONFIGURED indicating that Test Session is not configured.

The UWBS shall respond TEST_PERIODIC_TX_RSP with the status of STATUS_OK and starts Periodic Test. The UWBS SHALL periodically starts sending UWB packets with PSDU Data as a payload. The periodicity is configured by the T_GAP Test Configuration Parameter and the number of packets to be transferred is configured by NUM_PACKETS APP Configuration Parameter.

The UWBS shall notify TEST_PERIODIC_TX_NTF notification with the status of STATUS_OK after NUM_PACKETS packets are sent over UWB to the intended destination UWB device.

TEST_PERIODIC_TX_CMD

Payload Field(s)	Size (octet)	Description
PSDU Data	N Octets	PSDU Data[0:N] bytes; 0 <= N <= 127 for BPRF; 0 <= N <= 4095 for HPRF.

TEST_PERIODIC_TX_RSP

Payload Field(s)	Size (octet)	Description
Status	1	Status code as per [FIRA_UCI_SPEC].

TEST_PERIODIC_TX_NTF

Payload Field(s)	Size (octet)	Description
Status	1	Status code as per [FIRA_UCI_SPEC].

2.3.4 run_fira_test_rx

Script **run_fira_test_rx** is provided to demonstrate FiRa RX Test. This RF test can be used to report signal parameters like SNR, AOA etc. Generally, this test is used to receive a single packet by using the TEST_RX_CMD command.

2.3.4.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-c / –channel	Refer to CHANNEL_NUMBER in `FIRA UCI Technical Specification`
-v / –verbose	Print additional debug information
–preamble-code-index	Refer to PREAMBLE_CODE_INDEX in `FIRA UCI Technical Specification`
–sfd-id	Refer to SFD_ID in `FIRA UCI Technical Specification`
–rframe-config	Refer to RFRAME_CONFIG in `FIRA UCI Technical Specification` (default 0)
–psdu-data-rate	Refer to PSDU_DATA_RATE in `FIRA UCI Technical Specification`
–phr-data-rate	Refer to BPRF_PHR_DATA_RATE in `FIRA UCI Technical Specification`
–preamble-duration	Refer to PREAMBLE_DURATION in `FIRA UCI Technical Specification`
–nb-sts-segments	Refer to NUMBER_OF_STS_SEGMENTS in `FIRA UCI Technical Specification`
–sts-length	Refer to STS_LENGTH in `FIRA UCI Technical Specification`
–en-diag	Set the Qorvo ENABLE_DIAGNOSTIC parameter to 1
–antenna-set-id	Set the antenna set to use for the session. (default: 0)

2.3.4.2 Application

Refer to “Application Configuration Parameters” in `FIRA UCI Technical Specification` for the possible values on APP configuration parameters.

Test configuration parameters for RX Test

Parameter Name	Len (octets)	ID	Description
RMARKER_TX_ST	4	0x06	Start time of TX in 1/(128*499.2MHz) units
STS_INDEX_AUTC	1	0x08	0x00: STS_INDEX config value is used for all PER Rx/ Periodic TX test. (default) Commented [SV4]: CR-241

The RX Test shall be triggered by using the TEST_RX_CMD command and UWBS shall respond by TEST_RX_RSP when a command is accepted successfully. The Configuration Parameters defined in the tables below affect the behavior of the test.

The TEST_RX_CMD command shall be issued only after applying all required configuration parameters and the “Device Test Mode” session SHALL be in SESSION_STATE_IDLE Session State, otherwise UWBS SHALL respond TEST_RX_RSP with Status of STATUS_ERROR_SESSION_NOT_CONFIGURED, indicating that session is not configured.

UWBS shall notify TEST_RX_NTF notification after the UWB packet is received from the intended device.

TEST_RX_CMD

Payload Field(s)	Size (octet)	Description
Command	0 Octets	Start RX Test

TEST_RX_RSP

Payload Field(s)	Size (octet)	Description
Status	1	Status code as per [FIRA_UCI_SPEC].

TEST_RX_NTF

Payload Field(s)	Size (octet)	Description
Status	1	Refer to generic status codes
RX_DONE_TS_IN	4	Integer part of timestamp 1/124.8Mhz ticks
RX_DONE_TS_F	2	Fractional part of timestamp in 1/(128 * 499.2Mhz) ticks
AoA Azimuth	2	AoA Azimuth in degrees and it is a signed value in Q9.7 format. This field is zero if AOA_RESULT_REQ = 0.
AoA Elevation	2	AoA Elevation in degrees and it is a signed value in Q9.7 format. This field is zero if AOA_RESULT_REQ = 0.
ToA Gap	1	ToA of main path minus ToA of first path in nanoseconds
PHR	2	Received PHR (bits 0-12 as per IEEE spec)
PSDU Data Length	2	Length of PSDU Data(N) to follow
PSDU Data	N Octets	PSDU Data[0:N] bytes; 0 <= N <= 127 for BPRF; 0 <= N <= 4095 for HPRF.

2.3.4.3 Example

Use 2 different shells, one for each DUT. They will be called here after DUT-TX and DUT-RX.

In shell 'DUT-RX', start the device in a 'listener' test mod:

```
run_fira_test_rx -p <dut rx port>
```

In shell 'DUT-TX', request a given payload to be sent:

```
run_fira_test_rx  -p <dut tx port> --psdu '0a.0b.0c.0d.0e.0f'
```

In shell 'DUT-RX', verify the received data: You should expect the same data to be received ...

```
RxTestOutput gid:13, oid:5, Test Result:
...
PSDU data:     0a.0b.0c.0d.0e.0f
...
```

2.3.5 run_fira_test_ss_twr

Script **run_fira_test_ss_twr** is provided to demonstrate FiRa SS-TWR Test Mode. This mode can be used to measure single SS-TWR ToF using SP3 packets.

- Initiator transmits the POLL packet and waits for a RESPONSE packet from the responder, after SLOT_DURATION.
- Initiator then reports the time difference between the two packets as the round-trip time to AP.
- Similarly, responder waits for the POLL packet, transmits a RESPONSE packet after SLOT_DURATION and reports the time difference between the two packets as the reply time to AP. Based on these two measurements, AP can determine the ToF based on the SS-TWR formula.

2.3.5.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-c / –channel	Refer to CHANNEL_NUMBER in `FIRA UCI Technical Specification`
-r / –responder	Start as responder (default initiator) mode
-v / –verbose	Print additional debug information
–preamble-code-index	Refer to PREAMBLE_CODE_INDEX in `FIRA UCI Technical Specification`
–sfd-id	Refer to SFD_ID in `FIRA UCI Technical Specification`
–phr-data-rate	Refer to BPRF_PHR_DATA_RATE in `FIRA UCI Technical Specification`
–preamble-duration	Refer to PREAMBLE_DURATION in `FIRA UCI Technical Specification`
–nb-sts-segments	Refer to NUMBER_OF_STS_SEGMENTS in `FIRA UCI Technical Specification`
–sts-length	Refer to STS_LENGTH in `FIRA UCI Technical Specification`
–en-diag	Set the Qorvo ENABLE_DIAGNOSTIC parameter to 1
–antenna-set-id	Set the antenna set to use for the session. (default: 0)

2.3.5.2 Application

Refer to “Application Configuration Parameters” in `FIRA UCI Technical Specification` for the possible values on APP configuration parameters.

Test configuration parameters for FiRa SS-TWR Test

Parameter Name	Len (octets)	ID	Description
STS_INDEX_AUTO_IN	1	0x08	0x00: STS_INDEX config value is used for all PER Rx/ Periodic TX test. (default)

AP shall use the TEST_SS_TWR_CMD command to trigger the SS-TWR ranging test for SP3 packets and UWBS shall respond by TEST_SS_TWR_RSP when a command is accepted successfully.

The TEST_SS_TWR_CMD command shall be issued only after applying all the required configuration parameters and the “Device Test Mode” session SHALL be in SESSION_STATE_IDLE state, otherwise UWBS SHALL respond TEST_SS_TWR_RSP with Status of STATUS_ERROR_SESSION_NOT_CONFIGURED, indicating that session is not configured. UWBS SHALL notify TEST_SS_TWR_NTF notification after completing a single SS-TWR ranging measurement round involving POLL and RESPONSE packets.

TEST_SS_TWR_CMD

Payload Field(s)	Size (octet)	Description
PSDU Data	N Octets	PSDU Data[0:N] bytes; 0 <= N <= 127 for BPRF; 0 <= N <= 4095 for HPRF.

TEST_SS_TWR_RSP

Payload Field(s)	Size (octet)	Description
Status	1	Status code as per [FIRA_UCI_SPEC].

TEST_SS_TWR_NTF

Payload Field(s)	Size (octet)	Description
Status	1	Refer to generic status codes
Measure-ment	4	Contains Tround time of Initiator or Treply time of Responder depending on DEVICE_ROLE option. This is expressed in 1/(128 * 499.2Mhz) ticks.

2.3.6 run_fira_twr

Script **run_fira_twr** is provided to demonstrate a FiRa session ranging which can be easily modified.

2.3.6.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-t / –time	Set the duration of the ranging session (in second); `-1` - range forever
-p / –port	Specify communication interface
-c / –channel	Refer to CHANNEL_NUMBER in `FIRA UCI Technical Specification`
-s / –session	Set session ID (default: 42)
-v / –verbose	Prints additional debug information
–controlee	Refer to DEVICE_TYPE in `FIRA UCI Technical Specification`
–round	Refer to RANGING_ROUND_USAGE in `FIRA UCI Technical Specification`
–round-ctrl	Refer to RANGING_ROUND_CONTROL in `FIRA UCI Technical Specification`
–en-key-rot	Refer to KEY_ROTATION in `FIRA UCI Technical Specification`
–key-rot-rate	Refer to KEY_ROTATION_RATE in `FIRA UCI Technical Specification`
–sts	Refer to STS_CONFIG in `FIRA UCI Technical Specification`
–slot-span	Refer to SLOT_DURATION in `FIRA UCI Technical Specification`
–node	Refer to MULTI_NODE_MODE in `FIRA UCI Technical Specification`
–ranging-span	Refer to RANGING_DURATION in `FIRA UCI Technical Specification`
–en-diag	Set the Qorvo ENABLE_DIAGNOSTIC parameter to 1
–diag-fields	Set the Qorvo DIAGNOSTIC_FRAME_REPORTS_FIELD value OR flags: metrics, aoa, cir, cfo. (defa
–meas-max	Refer to MAX_NUMBER_OF_MEASUREMENTS in `FIRA UCI Technical Specification`
–skey SKEY	Refer to SESSION_KEY in `FIRA UCI Technical Specification`
–mac	Refer to DEVICE_MAC_ADDRESS in `FIRA UCI Technical Specification`
–dest-mac	Refer to DST_MAC_ADDRESS in `FIRA UCI Technical Specification`
–frame	Refer to RFRAME_CONFIG in `FIRA UCI Technical Specification`
–ssession	Refer to SUB_SESSION_ID in `FIRA UCI Technical Specification`
–sskey	Refer to SUB_SESSION_KEY in `FIRA UCI Technical Specification`
–en-rssi	Refer to RSSI_REPORTING in `FIRA UCI Technical Specification`
–stats	Enable Statistics report at end of the run
–diag_dump	Dump the Diagnostics in the provided JSON file
–n_controlees	Refer to NUMBER_OF_CONTROLEES in `FIRA UCI Technical Specification`
–block_stride_length	Refer to BLOCK_STRIDE_LENGTH in `FIRA UCI Technical Specification`
–sts-length	Refer to STS_LENGTH in `FIRA UCI Technical Specification`
–vendor-id	Refer to VENDOR_ID in `FIRA UCI Technical Specification`
–static-sts	Refer to STATIC_STS_IV `FIRA UCI Technical Specification`
–aoa-report	Refer to AOA_RESULT_REQ `FIRA UCI Technical Specification`
–preamble-idx	Refer to PREAMBLE_CODE_INDEX `FIRA UCI Technical Specification`
–sfd	Refer to SFD_ID `FIRA UCI Technical Specification`
–slots-per-rr	Refer to SLOTS_PER_RR `FIRA UCI Technical Specification`
–hopping-mode	Refer to HOPPING_MODE `FIRA UCI Technical Specification`

Warning(!), `--diag_dump` has an effect only with the `--stats` option.

The Diagnostics parameter save the diagnostics report in a json file named `range_data_<year>-<month>-<day>-<time>.json`. For instance: `range_data_23-04-12-10h47m25s.json`

The report contains one entry per Fira ranging sequence, each of these entries contains one entry per frame exchanged.

2.3.6.2 Example with Default values

Initialize a TWR FiRa session as controller on the first board.

```
run_fira_twr -p <controller port>
```

Open a second command shell in the Python script location.

Initialize a TWR FiRa session as a controlee on the second board.

```
run_fira_twr -p <controlee port> --controlee
```

Output of the script running on the controlee side with default parameter:

```
Initializing session 42...
Session 2 -> Init (StateChangeWithSessionManagementCommands)
Using Fira 2.0 session handle is : 2
Setting session 2 config ...
    DeviceType (0x0):                   0x0
    DeviceRole (0x11):                  0x0
    MultiNodeMode (0x3):                0x0
    RangingRoundUsage (0x1):            0x2
    DeviceMacAddress (0x6):             0x1
    ChannelNumber (0x4):                0x9
    ScheduleMode (0x22):                0x1
    StsConfig (0x2):                    0x0
    RframeConfig (0x12):                0x3
    ResultReportConfig (0x2e):          0xb
    VendorId (0x27):                    0x708
    StaticStsIv (0x28):                 0x60504030201
    AoaResultReq (0xd):                 0x1
    UwbInitiationTime (0x2b):           0x0
    PreambleCodeIndex (0x14):           0xa
    SfdId (0x15):                       0x2
    SlotDuration (0x8):                 0x960
    RangingInterval (0x9):              0xc8
    SlotsPerRr (0x1b):                  0x19
    MaxNumberOfMeasurements (0x32):     0x0
    HoppingMode (0x2c):                 0x0
    RssiReporting (0x13):               0x0
    BlockStrideLength (0x2d):           0x0
    NumberOfControlees (0x5):           0x1
    DstMacAddress (0x7):                [0]
    StsLength (0x35):                   0x1
Session 6 -> Idle (StateChangeWithSessionManagementCommands)
Starting ranging...
Device -> Active
Session 6 -> Active (StateChangeWithSessionManagementCommands)

[...]

# Ranging Data:
        session handle:     2
        sequence n:         8
        ranging interval:   200 ms
        measurement type:   Twr
        Mac add size:       2
```

(continues on next page)

(continued from previous page)

```
        primary session id: 0x0
        n of measurement:   1
        # Measurement 1:
            status:             Ok (0x0)
            mac address:        00:00 hex
            is nlos meas:       Unknown
            distance:           62.0 cm
            AoA azimuth:        -10.6328125 deg
            AoA az. FOM:        92.0 %
            AoA elevation:      0.0 deg
            AoA elev. FOM:      0.0
            AoA dest azimuth:   0.0 deg
            AoA dest az. FOM:   0.0 %
            AoA dest elevation: 0.0 deg
            AoA dest elev. FOM: 0.0 %
            slot in error:      0
            rssi:               -0.0 dBm

[...]

Stopping ranging...
Session 2 -> Idle (StateChangeWithSessionManagementCommands)
Device -> Ready
Deinitializing session...
Session 2 -> DeInit (StateChangeWithSessionManagementCommands)
Ok
```

2.3.6.3 Example with Statistics, Diagnostic and RSSI

Initialize a TWR FiRa session as controller on the first board.

```
run_fira_twr -p <controller port> --stats --en-diag -en-rssi
```

Open a second command shell in the Python script location.

Initialize a TWR FiRa session as controlee on the second board (in this example, in COM40).

```
run_fira_twr -p <controlee port> --controlee
```

Output of the script running on the controller side:

```
Initializing session 42...
Session 4 -> Init (StateChangeWithSessionManagementCommands)
Using Fira 2.0 session handle is : 4
Setting session 4 config ...
    DeviceType (0x0):                   0x0
    DeviceRole (0x11):                  0x0
    MultiNodeMode (0x3):                0x0
    RangingRoundUsage (0x1):            0x2
    DeviceMacAddress (0x6):             0x1
    ChannelNumber (0x4):                0x9
    ScheduleMode (0x22):                0x1
    StsConfig (0x2):                    0x0
    RframeConfig (0x12):                0x3
    ResultReportConfig (0x2e):          0xb
```

(continues on next page)

(continued from previous page)

```
    VendorId (0x27):                    0x708
    StaticStsIv (0x28):                 0x60504030201
    AoaResultReq (0xd):                 0x1
    UwbInitiationTime (0x2b):           0x0
    PreambleCodeIndex (0x14):           0xa
    SfdId (0x15):                       0x2
    SlotDuration (0x8):                 0x960
    RangingInterval (0x9):              0xc8
    SlotsPerRr (0x1b):                  0x19
    MaxNumberOfMeasurements (0x32):     0x0
    HoppingMode (0x2c):                 0x0
    RssiReporting (0x13):               0x0
    BlockStrideLength (0x2d):           0x0
    NumberOfControlees (0x5):           0x1
    DstMacAddress (0x7):                [0]
    StsLength (0x35):                   0x1
Session 4 -> Idle (StateChangeWithSessionManagementCommands)
Starting ranging...
Device -> Active
Session 4 -> Active (StateChangeWithSessionManagementCommands)

[...]

# Ranging Data:
        session handle:         2
        sequence n:         23
        ranging interval:   200 ms
        measurement type:   Twr
        Mac add size:       2
        primary session id: 0x0
        n of measurement:   1
        # Measurement 1:
            status:             Ok (0x0)
            mac address:        00:01 hex
            is nlos meas:       Unknown
            distance:           76.0 cm
            AoA azimuth:        0.0 deg
            AoA az. FOM:        0.0 %
            AoA elevation:      0.0 deg
            AoA elev. FOM:      0.0
            AoA dest azimuth:   7.296875 deg
            AoA dest az. FOM:   92.0 %
            AoA dest elevation: 0.0 deg
            AoA dest elev. FOM: 0.0 %
            slot in error:      0
            rssi:               -60.0 dBm

# Ranging Diagnostic Data:
        Session handle:     2
        Sequence n:     23
        Nbr of reports: 6
        # Ranging Diag. Report 0:
            Message id:    Control
            Action:        Tx
            Antenna_set:   0
```

(continues on next page)

(continued from previous page)

```
            Nbr of fields: 1
            # Frame Status Report:
                is processing ok  : 1
                is wifi activated : 0
        # Ranging Diag. Report 1:
            Message id:    RangingInitiation
            Action:        Tx
            Antenna_set:   0
            Nbr of fields: 1
            # Frame Status Report:
                is processing ok  : 1
                is wifi activated : 0
        # Ranging Diag. Report 2:
            Message id:    RangingResponse
            Action:        Rx
            Antenna_set:   0
            Nbr of fields: 3
            # Frame Status Report:
                is processing ok  : 1
                is wifi activated : 0
            # CFO Report:
                cfo:     -1.669 ppm
            # Segment Metrics Reports:
                Nbr of Segment Metrics: 1
                # Segment Metrics       0:
                    segment type:       1
                    primary_recv:       1
                    receiver Id:        0x0
                    noise_value:        -80
                    rsl_dBm:            -64.18359375
                    path1_rsl_dbm:      -65.88671875
                    path1_idx:          362
                    path1_snr:          14.11328125
                    path1_t:            23183
                    peak_rsl_dbm:       -64.18359375
                    peak_idx:           704
                    peak_snr:           15.81640625
                    peak_t:             45056
        # Ranging Diag. Report 3:
            Message id:    RangingFinal
            Action:        Tx
            Antenna_set:   0
            Nbr of fields: 1
            # Frame Status Report:
                is processing ok  : 1
                is wifi activated : 0
        # Ranging Diag. Report 4:
            Message id:    MeasurementReport
            Action:        Tx
            Antenna_set:   0
            Nbr of fields: 1
            # Frame Status Report:
                is processing ok  : 1
                is wifi activated : 0
        # Ranging Diag. Report 5:
```

(continues on next page)

(continued from previous page)

```
            Message id:    RangingResultReport
            Action:        Rx
            Antenna_set:   0
            Nbr of fields: 3
            # Frame Status Report:
                is processing ok  : 1
                is wifi activated : 0
            # CFO Report:
                cfo:     -1.714 ppm
            # Segment Metrics Reports:
                Nbr of Segment Metrics: 1
                # Segment Metrics       0:
                    segment type:       0
                    primary_recv:       1
                    receiver Id:        0x0
                    noise_value:        -80
                    rsl_dBm:            -56.03515625
                    path1_rsl_dbm:      -57.69140625
                    path1_idx:          739
                    path1_snr:          22.30859375
                    path1_t:            47326
                    peak_rsl_dbm:       -56.03515625
                    peak_idx:           320
                    peak_snr:           23.96484375
                    peak_t:             20480

[...]

Deinitializing session...
Session 2 -> DeInit (StateChangeWithSessionManagementCommands)
Device -> Ready
Ok
Device: 00:01
                    25 Successful/ 26 Total
                    AVG Ranging: 68.80
                    STDEV Ranging: 3.75
                    AVG AoA Azimuth: 0.000
                    STDEV AoA Azimuth: 0.000
                    AVG AoA Elevation: 0.000
                    STDEV AoA Elevation: 0.000
```

Statistics of the ranging are calculated at the end of the script.

2.3.6.4 Example for one-to-many Ranging

Initialize a One-To-Many TWR FiRa session as controller on the first board.

```
run_fira_twr -p <controller port> --node onetomany --dest-mac [0x01,0x02] --n_controlees 2
```

Open a second command shell in the Python script location.

Initialize a TWR FiRa session as controlee on the second board (in this example, in COM40).

```
run_fira_twr -p <controlee 1 port> --node onetomany --controlee --mac 1
```

Open a third command shell in the Python script location.

Initialize a TWR FiRa session as controlee on the third board (in this example, in COM18).

```
run_fira_twr -p <controlee 2 port>  --node onetomany --controlee --mac 2
```

Output of the script running on the controller side:

```
Initializing session 42...
Session 42 Init (StateChangeWithSessionManagementCommands)
Setting session 42 config ...
DeviceRole (0x11):                  0x1
DeviceType (0x0):                   0x1
MultiNodeMode (0x3):                0x1
RangingRoundUsage (0x1):            0x2
DeviceMacAddress (0x6):             0x0
ChannelNumber (0x4):                0x9
ScheduleMode (0x22):                0x1
CapSizeRange (0x20):                0x510
StsConfig (0x2):                    0x0
RframeConfig (0x12):                0x3
ResultReportConfig (0x2e):          0xf
VendorId (0x27):                    0x708
StaticStsIv (0x28):                 0x60504030201
AoaResultReq (0xd):                 0x1
UwbInitiationTime (0x2b):           0x3e8
PreambleCodeIndex (0x14):           0x9
SfdId (0x15):                       0x2
SlotDuration (0x8):                 0x960
RangingInterval (0x9):              0xc8
SlotsPerRr (0x1b):                  0x19
MaxNumberOfMeasurements (0x32):     0x0
HoppingMode (0x2c):                 0x0
RssiReporting (0x13):               0x1
NumberOfControlees (0x5):           0x2
DstMacAddress (0x7):                [1, 2]
Starting ranging...
Session 42 Idle (StateChangeWithSessionManagementCommands)
Device Active
Session 42 Active (StateChangeWithSessionManagementCommands)

[...]

# Ranging Data:
    session id:         42
    sequence n:         0
    ranging interval:   200 ms
    measurement type:   Twr
    Mac add size:       2
    primary session id: 0x0
    n of measurement:   2
    # Measurement 1:
        status:             Ok (0x0)
        mac address:        01.00
        is nlos meas:       no
        distance:           45.0 cm
        AoA azimuth:        -30.9375 deg
        AoA az. FOM:        88.0 %
        AoA elevation:      0.0 deg
```

(continues on next page)

(continued from previous page)

```
        AoA elev. FOM:      0.0
        AoA dest azimuth:   10.375 deg
        AoA dest az. FOM:   92.0 %
        AoA dest elevation: 0.0 deg
        AoA dest elev. FOM: 0.0 %
        slot in error:      0
        rssi:               -63.0 dB
    # Measurement 2:
        status:             Ok (0x0)
        mac address:        02.00
        is nlos meas:       no
        distance:           22.0 cm
        AoA azimuth:        -66.578125 deg
        AoA az. FOM:        92.0 %
        AoA elevation:      0.0 deg
        AoA elev. FOM:      0.0
        AoA dest azimuth:   0.0 deg
        AoA dest az. FOM:   0.0 %
        AoA dest elevation: 0.0 deg
        AoA dest elev. FOM: 0.0 %
        slot in error:      0
        rssi:               -35.5 dB

[...]

Stopping ranging...
Deinitializing session...
Session 42 DeInit (StateChangeWithSessionManagementCommands)
Device Ready
Ok
```

2.4 qorvo

2.4.1 Qorvo Specific Scripts Introduction

The **qorvo** directory contains tests that evaluate various functionalities of Qorvo UWB chip. Each script in this folder is designed to perform a specific test, assessing various aspects of performance and functionality.

For detailed information on using these tests, please read the subsequent chapters or README.md files in the appropriate subdirectories.

2.4.2 run_qorvo_test_pll_lock

Script **run_qorvo_test_pll_lock** is provided to demonstrate Qorvo PLL Lock TX Test. This test requests the UWB radio to lock the PLL (IDLE_PLL) and reports the lock status. The test sets the radio to INIT_RC state once completed.

2.4.2.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-c / –channel	Refer to CHANNEL_NUMBER in `FIRA UCI Technical Specification`
-v / –verbose	prints additional debug information

2.4.2.2 Application

The Qorvo PLL Lock TX Test shall be triggered by using TEST_PLL_LOCK_CMD command and UWBS shall respond by TEST_PLL_LOCK_RSP when command is accepted successfully.

TEST_PLL_LOCK_CMD (GID = 0xB, OID = 0x1)

Payload Field(s)	Size (octet)	Description
N/A	N/A	N/A

TEST_PLL_LOCK_RSP (GID = 0xB, OID = 0x2)

Payload Field(s)	Size (octet)	Description
Status	1	Status code as per [FIRA_UCI_SPEC]

TEST_PLL_LOCK_NTF (GID = 0xB, OID = 0x2)

Payload Field(s)	Size (octet)	Description
TLV count	1	The number of TLVs following
TLF list	1	See TLV descriptions below

TLVs TEST_PLL_LOCK_NTF

Type	Leng	Value / Description
0x01	1 octet	Error code: 0 - SUCCESS: see PLL lock status table 1 - PLL_LOCK_ERROR: setting the channel failed 2 - INTERNAL_ERROR: setting the PLL state failed
0x02	4 octet	PLL_STATUS_CH9_BIST_FAIL_BIT_MASK (0x4000): PLL channel 9 BIST fail PLL_STATUS_CH5_BIST_FAIL_BIT_MASK (0x2000): PLL channel 5 BIST fail PLL_STATUS_LD_CODE_BIT_MASK (0x1f00): Counter-based lock-detect status indicator PLL_STATUS_XTAL_AMP_SETTLED_BIT_MASK (0x0040): Status flag from the XTAL indicating that the amplitude has settled PLL_STATUS_VCO_TUNE_UPDATE_BIT_MASK (0x0020): Flag to indicate that the COARSE_TUNE codes have been updated by cal and are ready to read PLL_STATUS_PLL_OVRFLOW_BIT_MASK (0x0010): PLL calibration flag indicating all VCO_TUNE values have been cycled through PLL_STATUS_PLL_HI_FLAG_BIT_MASK (0x0080): VCO freq too high indicator (active-high) PLL_STATUS_PLL_LO_FLAG_N_BIT_MASK (0x0004): VCO freq too low indicator (active-low) PLL_STATUS_PLL_LOCK_FLAG_BIT_MASK (0x0002): PLL lock flag PLL_STATUS_CPC_CAL_DONE_BIT_MASK (0x0001): PLL cal done and PLL locked

2.4.3 run_qorvo_test_tx_cw

Script **run_qorvo_test_tx_cw** is provided to demonstrate Qorvo Continuous Wave TX Test. This test configures the device to transmit a Continuous Wave (CW) at a specified channel of Qorvo UWB radio to transmit a Continuous Wave (CW) at a specified channel frequency. This may be of use as part of the crystal trimming procedure.

2.4.3.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Values
-p / –port	Specify communication interface
-c / –channel	Refer to CHANNEL_NUMBER in `FIRA UCI Technical Specification`
-t / –time	set the duration of the ranging session (in second); `-1` - range forever
-v / –verbose	prints additional debug information

2.4.3.2 Application

APP configuration parameters for Continuous Wave TX Test

Parameter Name	Tag	Length	Description
CHANNEL_NUMBER	0x04	1	0x05 for channel 5; >0x09 for channel 9

Test configuration parameters for Continuous Wave TX Test

Parameter Name	Tag	Length	Description
TX_ANTENNA_SELECTION	0xE7	1	ID of the TX antenna group

The Qorvo Continuous Wave TX Test shall be triggered by using TEST_TX_CW_CMD command and UWBS SHALL respond by TEST_TX_CW_CMD when a command is accepted successfully.

TEST_TX_CW_CMD (GID = 0xB, OID = 0x1)

Payload Field(s)	Size (octet)	Description
Enable	1	1 = start, 0 = stop

TEST_TX_CW_RSP (GID = 0xB, OID = 0x1)

Payload Field(s)	Size (octet)	Description
Status	1	Status code as per [FIRA_UCI_SPEC]

2.5 utils

2.5.1 Utilities Scripts Introduction

The **utils** directory contains general utilities that function as standalone tools or supporting scripts for the UQT package. These scripts are designed to work independently, without need of a connected device or communicating with any devices.

To learn more about detailed use cases, please read the subsequent chapters or README.md files in the appropriate subdirectories.

2.5.2 decode_uci

This script decodes command, response, and notification byte streams into a human-readable format. For more details, use `decode_uci -h`.

2.5.2.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Description
`bytes`	Byte stream
`-o` / `--output`	Set output file path. (default: `./tmp.csv`)

2.5.2.1.1 Byte Stream Format

The byte stream should be a series of bytes in the format: 1, 2, 3, etc. Each byte can be separated by:

- No separator
- A space ( )
- A dot (.)
- A colon (:)

Note: If using a space ( ) as the separator, the entire byte stream must be enclosed in single quotes (`' '`).

2.5.2.2 Example

Examples:

```
decode_uci 20.02.00.00
decode_uci '20 02 00 00'
decode_uci 410300020000 41.00.00.01.00  61.02.00.06.2a.00.00.00.00.00
decode_uci 410300020000 41.00.00.01.00  61.02.00.06.2a.00.00.00.00.00 -o ./output_file.csv
```

2.5.3 fp

Script **fp** is provided for handling conversion between natural and fixed-point numbers.

2.5.3.1 Parameters

Arguments with expected parameter available in this script:

Parameter	Description
v	Value to convert. May be an int, float or bytearray
I	Number of bits for the integer part
F	Number of bits for the fractional part
-r / –reverse	Convert a fix point to float instead (default: False)
-s / –signed	Presence of a sign bit

2.5.3.2 Example

```
fp 28 7 1
0x38
```

2.5.4 uqt_info

Print environment information

2.5.4.1 Example

```
uqt_info

# Python3 version:
    ...
# Customization (UQT_CUSTOM):
    ....
# Wanted Extensions (UQT_ADDINS):
    addin_name_1
    addin_name_2
    addin_name_3
    ...
# Loaded Extensions:
    ...
# DUT 'Default':
    unknown port ('UQT_PORT' not defined.)
```

2.5.5 uqt_ls

List available tools

2.5.5.1 Example

```
uqt_ls
-> <script_name_1>                          One line description of <script_name_1>
-> <script_name_2>                          One line description of <script_name_2>
-> <script_name_3>                          One line description of <script_name_3>
...
```