# UWB IoT Documentation

*Release v04.06.05*

**NXP**

**Mar 13, 2025**

# CONTENTS

CHAPTER

ONE

# UWB IOT

## 1.1 Introduction

UWB IoT Middleware is aimed to be common middleware for various NXP UWB Devices.

The same middleware can be used to run UWB Demos across various development and reference boards, across configurations of RTOS/compilers, etc.

It supports the following UWB Devices:

- SR150
- SR160
- SR040

It runs on various platforms:

- Rhodes V4
- Finder V3
- Raspberry PI
- Crete
- Nordic (NRF52840)

The MW is ported and run against the following OS:

- FreeRTOS
- Linux/POSIX

See Section 1.4 *Block Diagram* for the block diagrams.

## 1.2 Acronyms

The following acronyms are used in this document.

**BLE**
Bluetooth Low Energy.

**HBCI**
Protocol used to download FW to SR100T, SR110T, SR150.

**MCTT**
Mac Conformance Test Tool.

**PCTT**
PHY Conformance Test Tool.

**SW MW**
Secure Element Middleware.

**SWUP**
Protocol used to download FW and DSP Image to SR040.

**TDoA**
Time Difference of Arrival.

**UCI**
The logical interface between Host and UWBS.

**UWB**
Ultra Wide Band.

**UWB MW**
Ultra Wide Band Middleware / Ultra Wide Band IoT Middleware.

**UWBC**
Ultra Wide Band Controller. For the scope of this SDK/Middleware, same as UWBS.

**UWBD**
Ultra Wide Band Device. For the scope of this SDK/Middleware, same as UWBS.

**UWBS**
Ultra Wide Band Sub System. It refers to SR100T, SR110T, SR150, SR040.

The UWBS is the hardware component implementing the FiRa PHY and MAC specifications. The UWBS interfaces with the FiRa Framework through the UWB Command Interface (UCI).

## 1.3 Folder Structure

The location and purpose of important files and folders is as shown here.

**Warning:** Some of these folders are to showcase how the future UWB IoT Middleware delivery would contain. And hence, some of these folders are not present in this MW Package.

### 1.3.1 Root Folder

| Folder / File | Purpose |
|---|---|
| `EULA.pdf` | End User License Agreement |

### 1.3.2 `/binaries`

**Note:** On some packages, some of these files are moved to `/firmware_images` folder.

| Folder / File | Purpose |
|---|---|
| `binaries` | Prebuilt / generated binary blobs.<br>• These are either pre-built executables / tools<br>• Or pre-built binaries to be flashed on boards<br>• Or header files for SR1XX and SR040 ICs. |
| `binaries` / `Rhodes3` | Binaries / Firmware for Rhodes 3 counterpart |
| `binaries` / `Rhodes4` | Binaries / Firmware for Rhodes 4 counterpart |
| `binaries` / `NRF52840_MK` | Binaries for Nordic platform for SR150 |
| `binaries`/ `UWBD` | Binaries for UWBD |
| `binaries` / `UWBD` / `SR1XX` | Header files / Firmware for SR1XX |
| `binaries` / `FinderV3` | Binaries for QN9090 UWB Finder V3 board<br>• See `board-qn9090-sr040` board-qn9090-sr040 |
| `binaries` / `UWBD` / `SR040` | Binaries / Firmware for SR040 |
| `binaries` / `UWBD` / `SR040` / `DSP` | DSP Image for SR040 |
| `binaries` / `NRF52840_SR040` | Binaries for Nordic platform for SR040 |

### 1.3.3 /boards

| Folder / File | Purpose |
| --- | --- |
| boards | Boards specific adaptations, modifications and configurations |
| boards / Host | Files / settings for specific host |

### 1.3.4 /demos

The demos are in this folder. Please visit

- See Section 4.1: *Demo List SR150*
- See Section 5.1: *Demo List SE051W*
- see `demo-list-sr040`: demo-list-sr040
- `demos/SR1XX` :
  - Demos that are common for SR100T, SR100S, SR150, SR160.
  - These are all non secure demos, which do not need a secure element.
- `demos/SR150` :
  - Demos that are specific for SR150 that do not need SE051W.
- `demos/SR150_SE051W` :
  - Demos that are specific for SR150. These demos work with/for SE051W.
- `demos/SR1XX_SN110` :
  - Demos that are specific for SR100T, SR100S. These demos work only with/for SN110.
- `demos/SR040` :
  - Demos that are specific for SR040.
- `demos/pnp` :
  - This folder contains files to create a PnP binary for SR1XX, SR040.
- `demos/pnp_socket` :
  - This folder contains files to create a PnP binary for socket.
- `demos/mctt_pctt` :
  - This folder contains files to create MCTT/PCTT binaries for SR1XX.
- `demos/cdc` :

    - This folder contains files for CDC framework support.

- `demos/common` :

    - This folder contains common files required for all the demos.

- `demos/rhodes4` :

    - This folder contains Rhodes V4 demos for firmware download from external flash.

- `demos/SR160` :

    - Demos that are specific for SR160.

### 1.3.5 /docs

| Folder / File | Purpose |
|---|---|
| `docs` | Documentation |

### 1.3.6 /ext/boards

External SDKs and BSP Code is kept here.

| Folder / File | Purpose |
|---|---|
| `ext` / `boards` | Board specific files. |

### 1.3.7 /ext/freertos

Operating system for the examples

| Folder / File | Purpose |
|---|---|
| `ext` / `freertos` | FreeRTOS specific files |

### 1.3.8 /ext/TML-libuwbd

UWB Kernel Mode Drivers

| Folder / File | Purpose |
|---|---|
| `sr1xx` / `src` | Kernel driver for sr1xx |

### 1.3.9 /firmware_images

Prebuild Firmware Packages are Kept here.

| Folder / File | Purpose |
|---|---|
| `firmware_images` / `SR1XX` | Binaries / Firmware for SR1XX |
| `firmware_images` / `SR040` | Binaries / Firmware for SR040 |

### 1.3.10 /libs

Common libraries. These do not create an executable. They provide common functionalities.

### 1.3.11 /libs/halimpl

Important folders are as under:

| Folder / File | Purpose |
|---|---|
| libs / halimpl / hal | Hardware Abstraction Layer.<br>Hardware Abstraction layer sends and receives the UCI messages from HW/UCI stack. It is the interface layer for UCI stack and HW. |
| libs / halimpl / inc | Include files for Hardware Abstraction Layer |
| libs / halimpl / fwd | This module implements the HBCI protocol and handle formation of HBCI packet and send it to the TML layer via HAL. Parses the HBCI responses from HW and sends result back to the application. |
| libs / halimpl / tml | Transport Mapping Layer |
| libs / halimpl / transport | Transport Layer manges the flow of UCI data between HOST and UWBS to prevent congestion and ensure that UCI data is transmitted and received. |
| libs / halimpl / log | This module responsible for capturing and recording the UCI data. Section 9.3 *Logging Configurations* for more information. |
| libs / halimpl / osal | Osal layer abstracts and provide a standardized interface for interacting with the underlying operating systems or Real Time Operating System. |

### 1.3.12 /libs/uci-core

| Folder / File | Purpose |
|---|---|
| libs / src | UWB Controller Interface Core.<br>This module implements the UCI protocol and handle formation of UCI packet based on the request from application and send it to the TML layer via HAL. Parses the UCI responses and notification from HW and sends result back to the application. |
| libs / inc | Include files for UWB Controller Interface Core. |

### 1.3.13 /libs/uwb-iot

UWB IoT API and implementation layer.

| Folder / File | Purpose |
|---|---|
| libs / uwb-iot / uwb_api | UWB platform independent APIs running on top of UCI core These APIs are exposed to the applications running on platforms supported by UWB IoT MW . |
| libs / uwb-iot / uwb_api / Api / SR040 | SWUP APIs (See swup swup) |
| libs / uwb-iot / uwb_api / PrintUtility | Print utility files for UWB APIs . |
| libs / uwb-iot / uwb_core / include | Include files for UWB APIs |
| libs / uwb-iot / uwb_core / adaptation | Adaptation layer between UWB and UCI , UCI Core functionality accessed by UWB API using adaptation layer . |

### 1.3.14 /libs/uwbradar

**Note:** These are specific to SR160 only.

UWB IoT Radar API and implementation layer.

| Folder / File | Purpose |
|---|---|
| libs / uwbradar | UWB Radar |
| libs / uwbradar / include | UWB Radar APIs |

### 1.3.15 /project

MCUXpresso based project for SR150 and SR040 is kept here

| Folder / File | Purpose |
| --- | --- |
| `project / FinderV3` | MCUXpresso IDE project for finder V3 Board |
| `project / RhodesV4_SE` | MCUXpresso IDE project (See Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*). MCUXpresso IDE project of RV4 board, **without** the Secure Element SE051W projects |
| `project / Nrf52840_SR1xx` | Segger IDE project for SR1XX on Nordic platform (see Section 3.3.1 *NRF52840-SR1XX Standalone Segger Embedded Studio Project*) |
| `project / Nrf52840_SR040` | Segger IDE project for SR040 on Nordic platform (see `nrf52840_sr040-project` nrf52840_sr040-project) |

### 1.3.16 `/scripts`

Common scripts. Important files are as under

| Folder / File | Purpose |
| --- | --- |
| `scripts / dk6_image_tool.py` | Where applicable, this script is required for generating binaries for QN9090 in MCUXpresso IDE project. Python version 3.8 or later is needed to run this script successfully. |

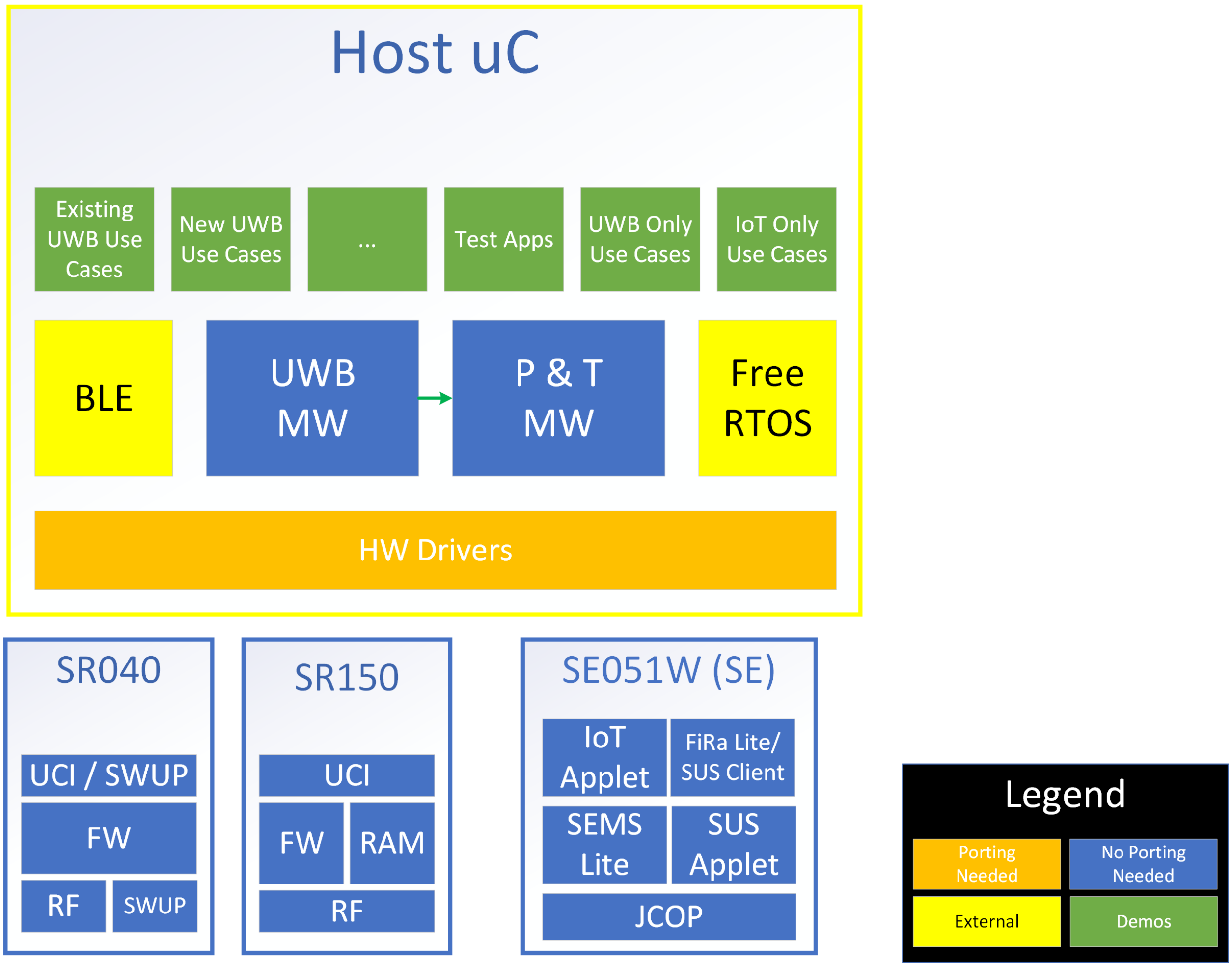
### 1.3.17 `/tools`

FiRa Lite adf provisioning helper

| Folder / File | Purpose |
| --- | --- |
| `tools / FiRaLite/fira_sgt_v0.5` | The tool is used to do ADF provisioning for FiRaLite applet for secure ranging use cases. |

## 1.4 Block Diagram

Basic diagram to show portable and reference code.




**UWB MW**
: This module is responsible for the UWB functionality

**P&T MW**
: This module is responsible for access to the Secure Element and corresponding functionality.

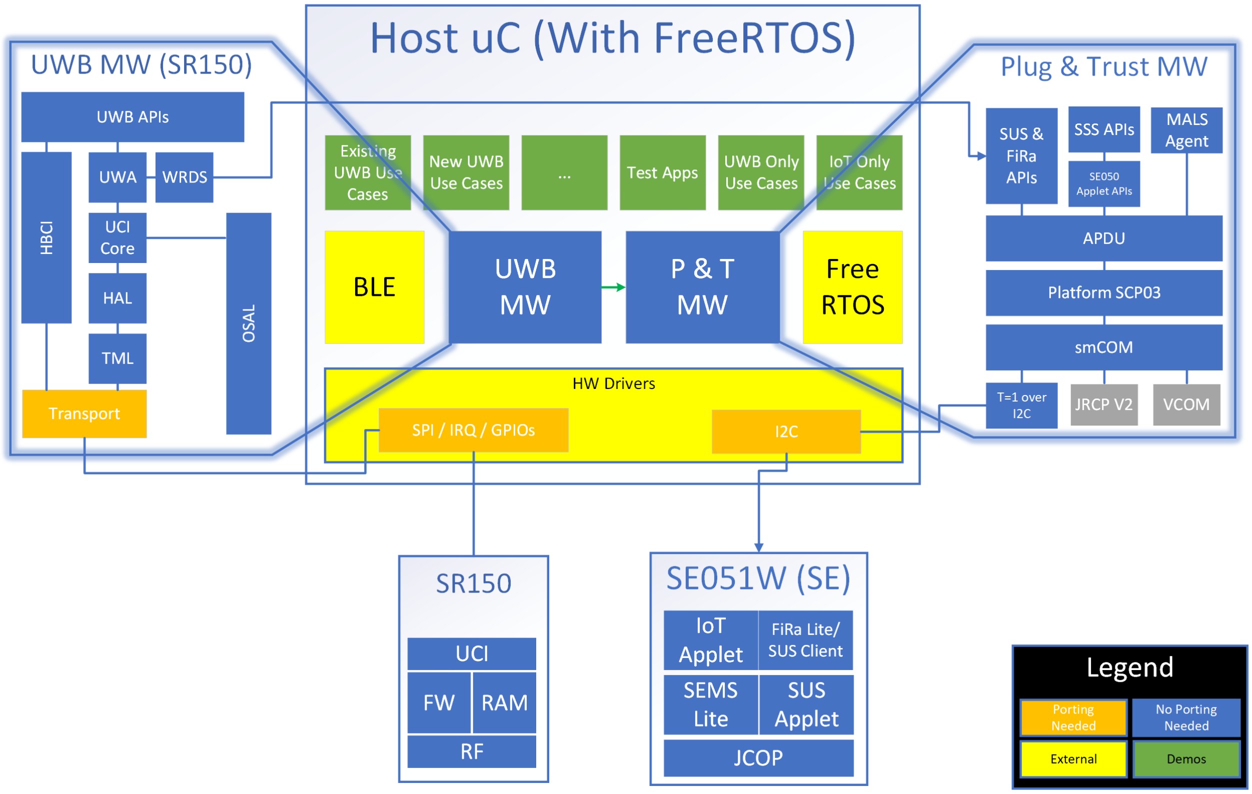
### 1.4.1 Components

Table below show overall UWB IoT MW architecture and it's components in various configurations.

| Item | Description |
| --- | --- |
| Demos / Integration / Applications | The user applications which uses exposed UWB and SE APIs to achieve the required use cases for UWB. |
| UWB APIs | This is the thin layer of platform independent APIs running on top of UCI core. These APIs are exposed to the applications running on platforms supported by UWB IoT MW. |
| UWA - UWB Adaptation Layer | This is the mediator layer between UWB API and UCI core. UCI Core functionality is accessed by UWB API using adaptation layer. |
| UCI stack | This module implements the UCI protocol and handle formation of UCI packet based on the request from application and send it to the TML layer via HAL. Parses the UCI responses and notification from HW and sends result back to the application. |
| WRDS | This layer supports handling of Wrapped RDS from SE051W. |
| OSAL layer | Operating system Abstraction Layer implements all the OS functionality like thread creation, buffer management, timer management and thread synchronization. |
| HAL | Hardware Abstraction layer sends and receives the UCI messages from HW/UCI stack. It is the interface layer for UCI stack and HW. This layer shall wake up the HW on request and trigger the FW download. |
| HBCI FW download | This layer implements the FW download functionality with Helios Boot Control Interface packets. FW download is triggered by HAL module and UWB HW is woken up.<br>This mode is used for SR100T, SR110T, SR150. |
| SWUP | This is the mode used for SW Update for SR040 |
| TML layer | Transport Management Layer Implements invokes SPI driver APIs to send and receive the UCI commands, responses and notification. |

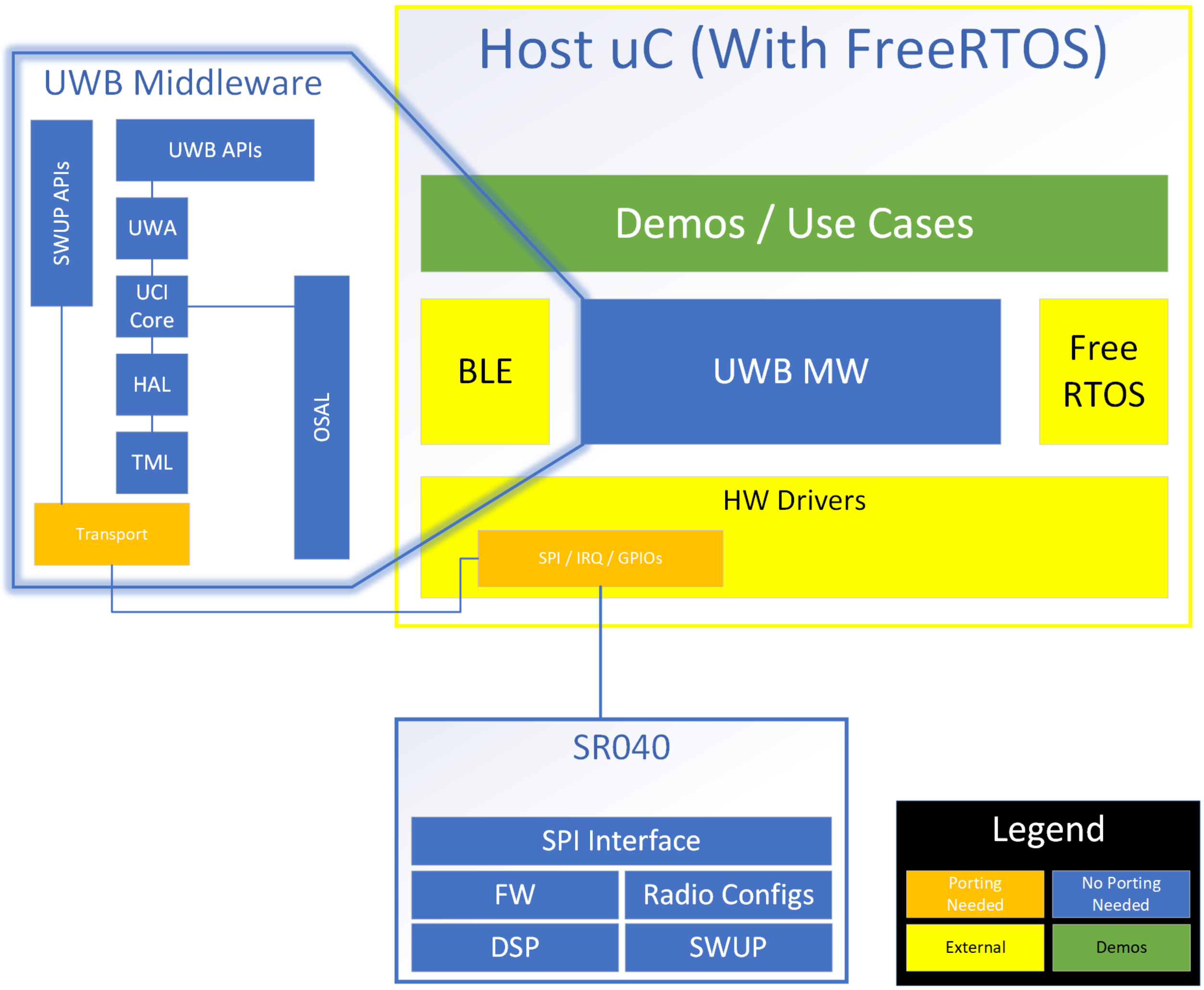
### 1.4.2 SR150 - FreeRTOS

Integration of the Middleware with SR150 and SE051W is as shown below.




### 1.4.3 SR040

Integration of the Middleware with SR040 is as shown below.




## 1.5 Modes of operation of the MW

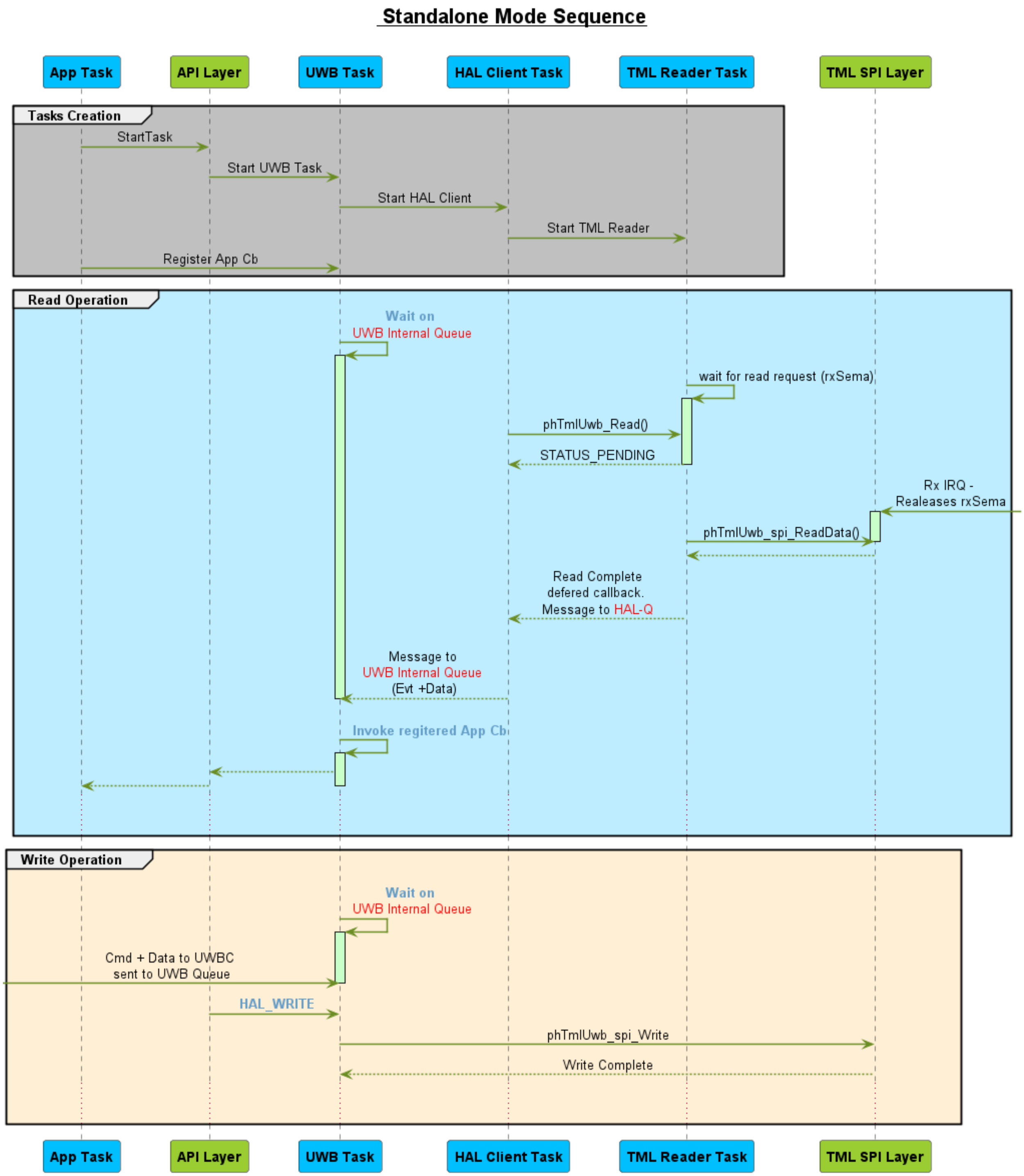
UWB IoT SDK has multiple modes of operations

### 1.5.1 Standalone Mode

- This is the default mode of operation.
- The demos run natively on the target MCU, and generally do not need any external stimulus.
- The SDK is delivered with UCI, HBCI[1] and SWUP[2] stack.

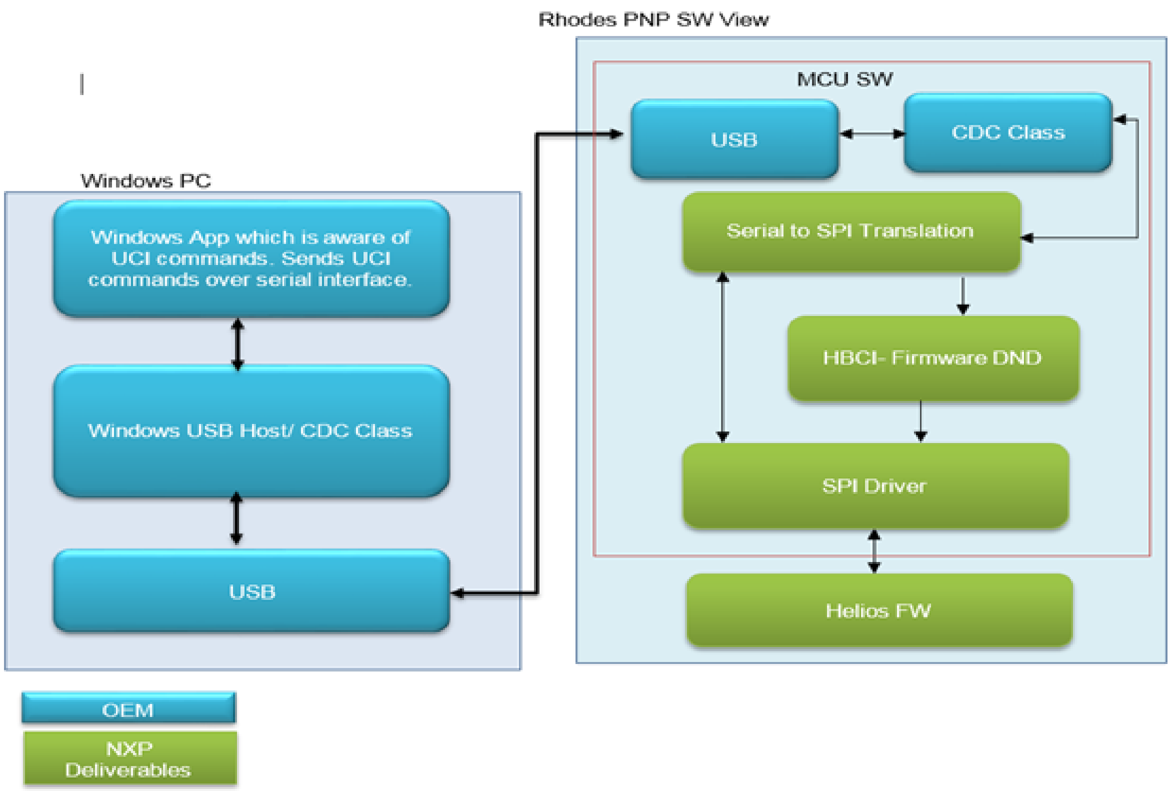
[1] Used for SR100T, SR110T and SR150.
[2] Used for SR040.

- Customer can run any application on MCU as dedicated application to configure the platform as Tag or Anchor
- If a customer decides to use their own MCU then UCI and HBCI stack should be re-usable.



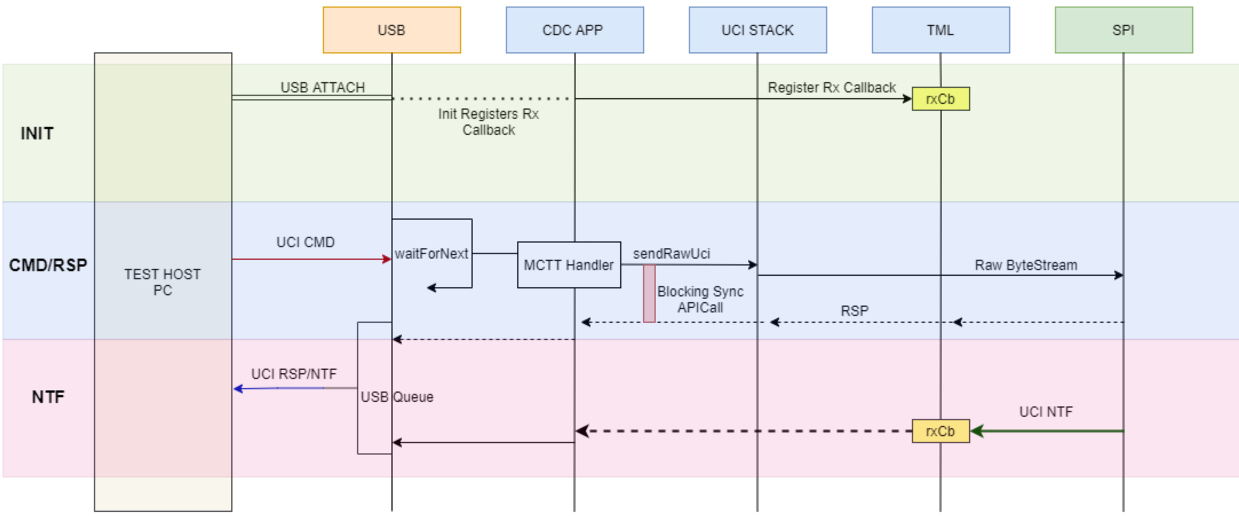
### 1.5.2 PNP Mode

- In this mode, the Host Micro Controller behaves as a tiny USB to SPI Bridge.
- The protocol between PC and the Host Micro Controller is designed to:
  - Reset the UWBD.
  - Get Unique ID of the Host Micro Controller.
  - Get version number of the PnP FW Running on the Host Micro Controller.
  - The protocol for PnP allows to download new FW via HBCI mode on SR100T, SR110T and SR150.
- This can be used to directly Send and receive RAW UCI Commands from PC to the UWBD.
- PnP Mode is available for Rhodes V3, Rhodes V4, Finder V3 and NRF52840 boards.
- Counterpart applications like UCI Tool, or small reference Python scripts can be used for quick prototyping and testing.



### 1.5.3 FiRa / MCTT / PCTT Mode

- In this mode, the host micro controller behaves as a tiny USB to SPI Bridge.
- The protocol is UCI.
- Only UCI Commands can be sent / received over this mode.
- Nothing special (as possible with *PNP Mode*) can be done in this mode.
- This mode is used for running PCTT and MCTT and related Test Suites.



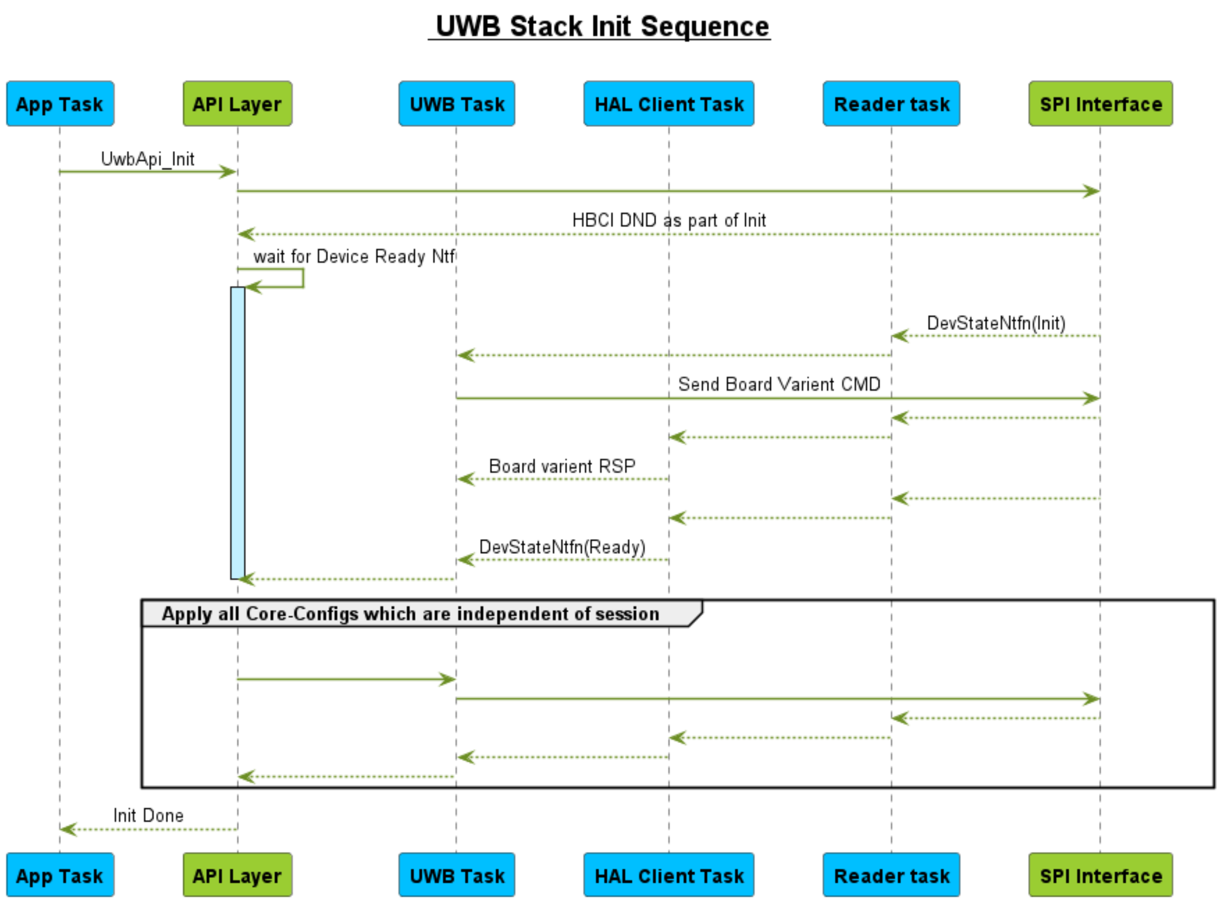

## 1.6 Dynamic View

**Contents**

### 1.6.1 UWB Stack Init Sequence

As a part of UWB Stack Init following things are done.

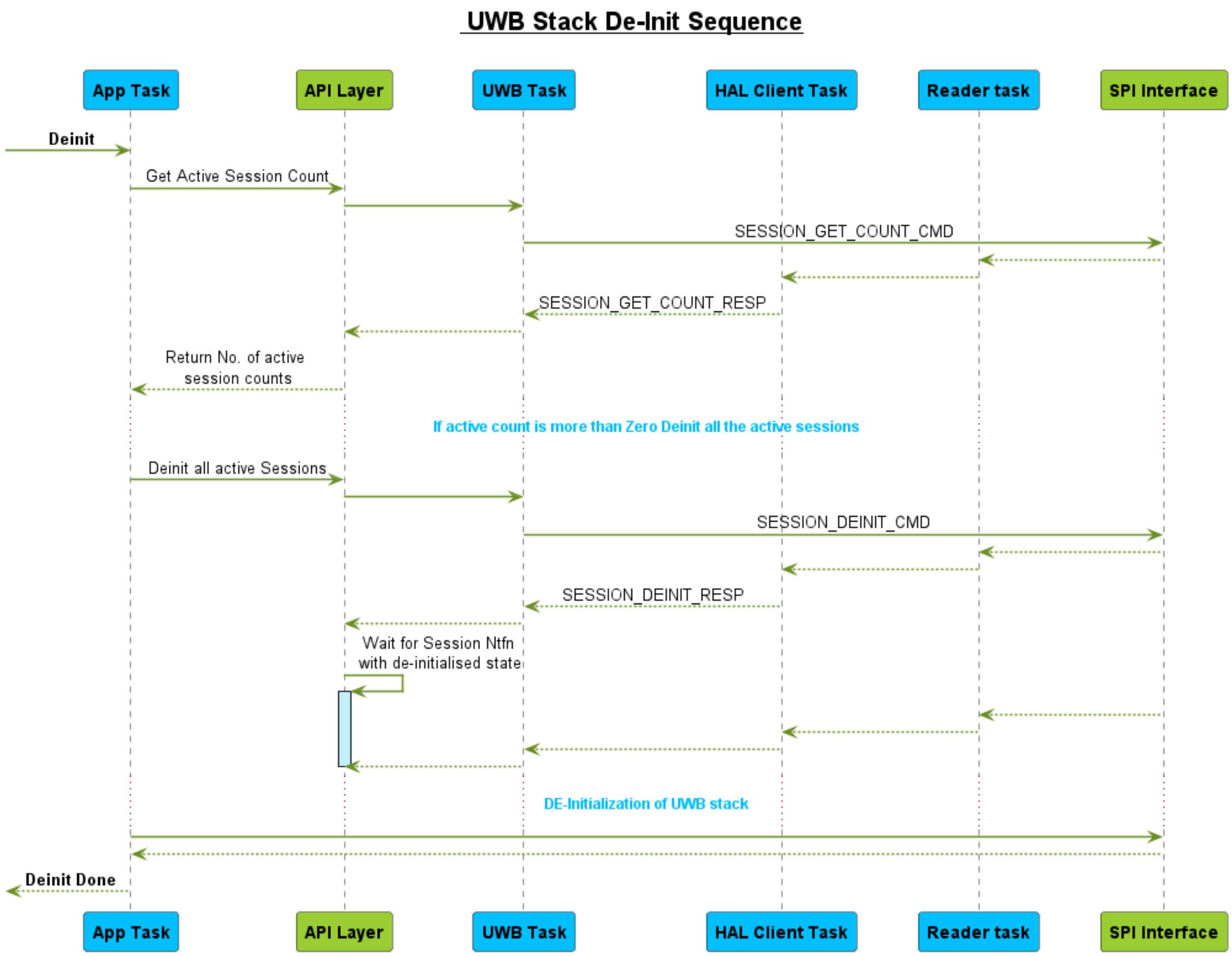
1) All required tasks are created and initialized.
2) Transport layer initialization.
3) HBCI FW download is done to initialize Helios.
4) Board variant is set as per Host setup.
5) All default configs are applied.



### 1.6.2 UWB Stack DeInit Sequence

As a part of UWB Stack DeInit following things are done.

1) Check if there are any active sessions created.
2) If active session count is more than Zero, de-initialize all active sessions.
3) MW stack de-init and memory cleanup.

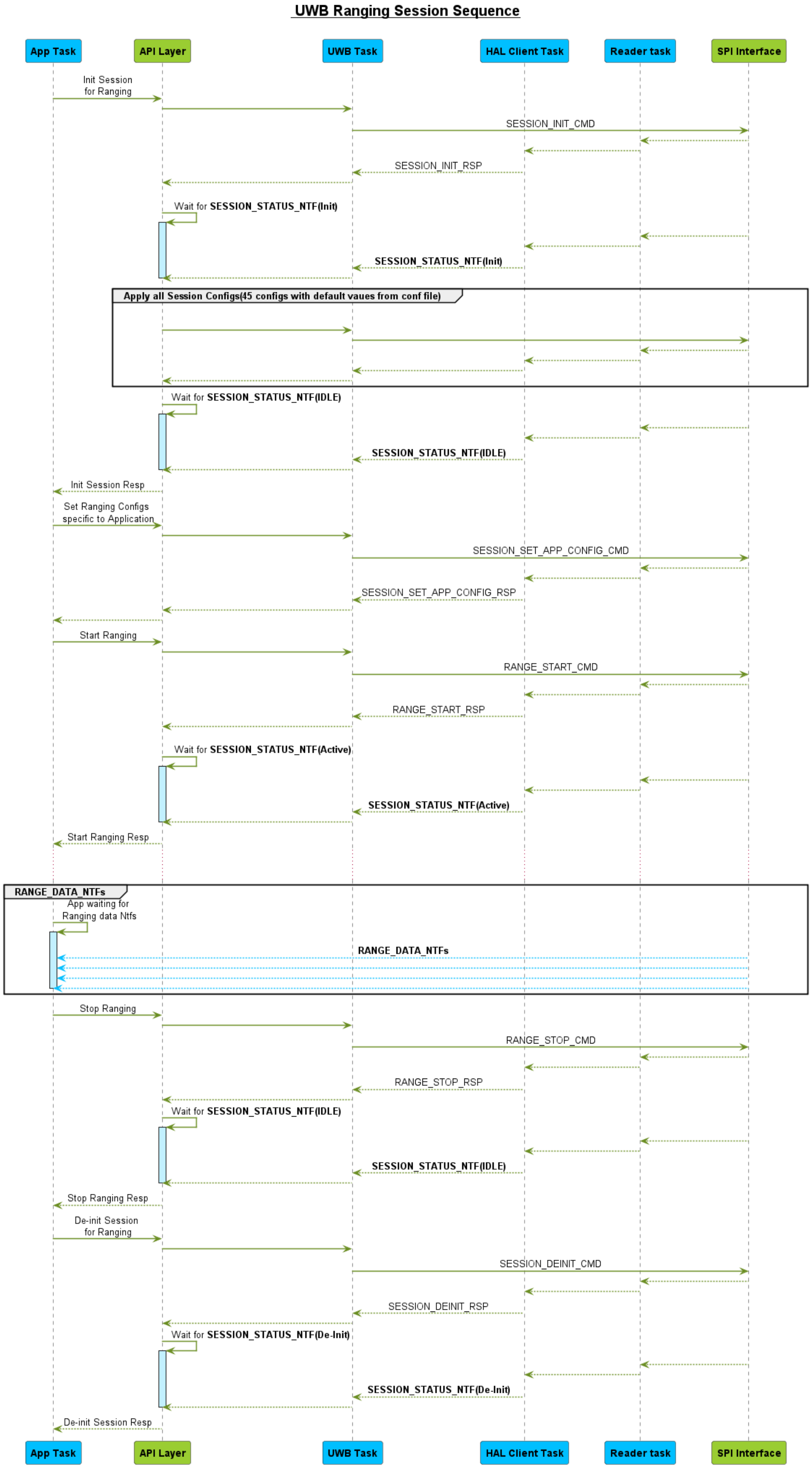



### 1.6.3 P2P, Multicast Ranging

As a part of P2P, Multicast ranging following operations are performed.

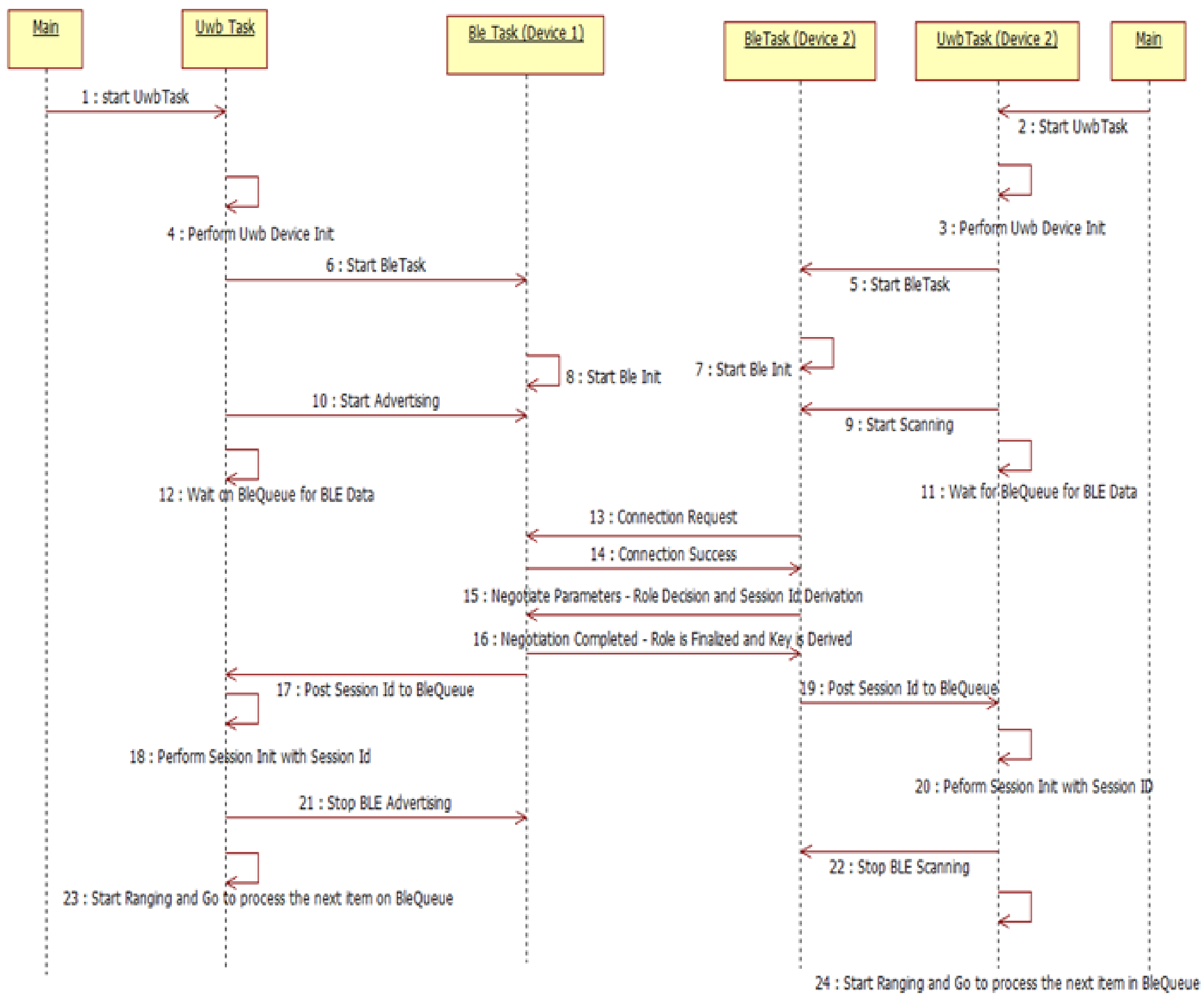
1) Before starting ranging UWB Stack should be in initialized state.
2) Below sequence is same for initiator and responder:
    1) Session is created for Ranging with 4 bytes unique session ID and session type `0x00`
    2) Same session ID should be used by both Initiator and responder.

3) All default session configs are applied at the time of session creation.
4) Application specific ranging configs are applied.
5) Ranging is started.
6) At the end ranging is stopped and session is de-initialized.


UWB Ranging Session Sequence
App Task
API Layer
UWB Task
HAL Client Task
Reader task
SPI Interface
Init Session for Ranging
SESSION_INIT_CMD
SESSION_INIT_RSP
Wait for SESSION_STATUS_NTF(Init)
SESSION_STATUS_NTF(Init)
Apply all Session Configs(45 configs with default vaues from conf file)
Wait for SESSION_STATUS_NTF(IDLE)
SESSION_STATUS_NTF(IDLE)
Init Session Resp
Set Ranging Configs specific to Application
SESSION_SET_APP_CONFIG_CMD
SESSION_SET_APP_CONFIG_RSP
Start Ranging
RANGE_START_CMD
RANGE_START_RSP
Wait for SESSION_STATUS_NTF(Active)
SESSION_STATUS_NTF(Active)
Start Ranging Resp
RANGE_DATA_NTFs
App waiting for Ranging data Ntfs
RANGE_DATA_NTFs
Stop Ranging
RANGE_STOP_CMD
RANGE_STOP_RSP
Wait for SESSION_STATUS_NTF(IDLE)
SESSION_STATUS_NTF(IDLE)
Stop Ranging Resp
De-init Session for Ranging
SESSION_DEINIT_CMD
SESSION_DEINIT_RSP
Wait for SESSION_STATUS_NTF(De-Init)
SESSION_STATUS_NTF(De-Init)
De-init Session Resp
App Task
API Layer
UWB Task
HAL Client Task
Reader task
SPI Interface

### 1.6.4 BLE - OOB

An out of the bounds setup for ranging with BLE would use the following sequence.

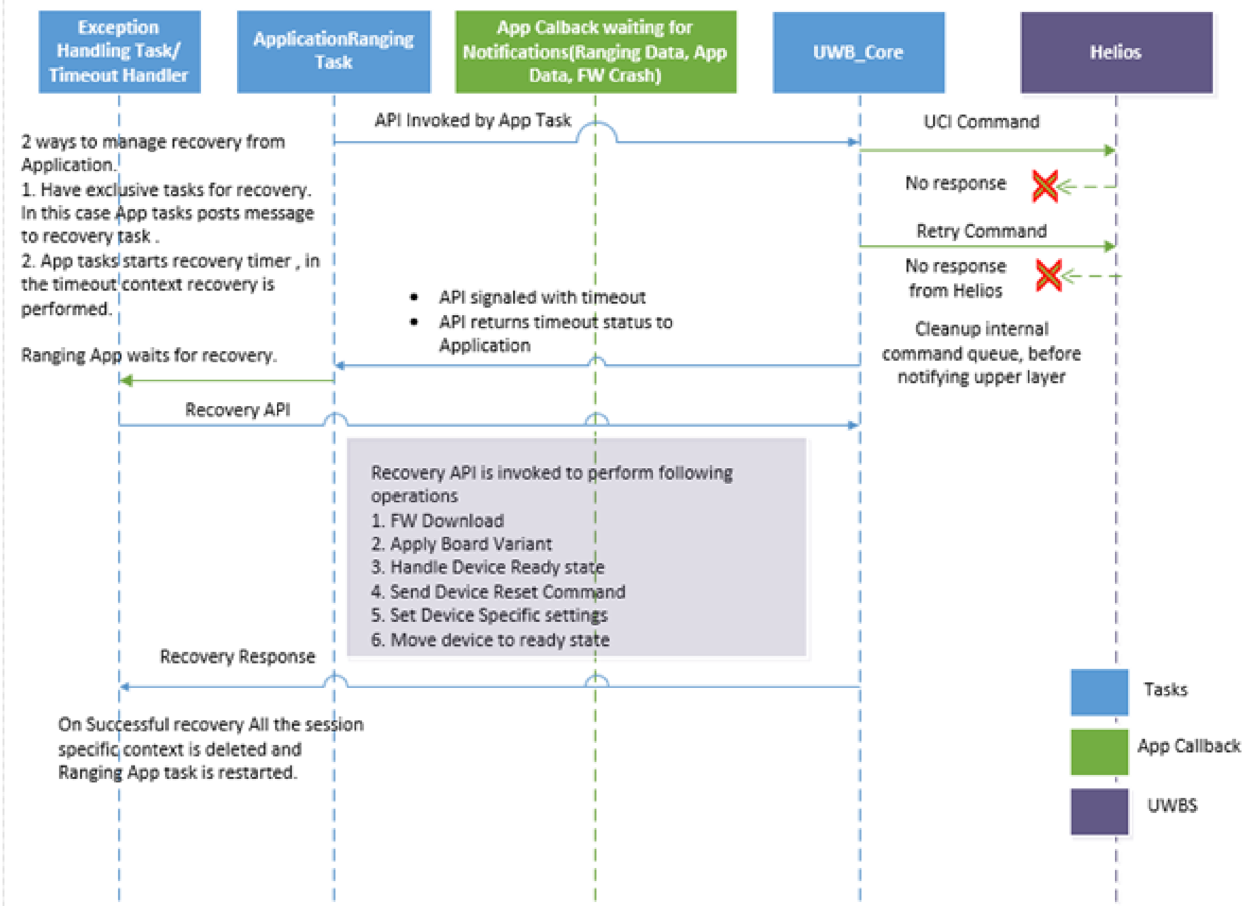



### 1.6.5 Recovery : UCI Command Timeout

Following steps are performed from MW to handle UCI command timeout

- API is invoked from Application Task, UCI command is sent to Helios as apart of API
- If response is not received command retry is attempted. If response is not received for the second time as well UCI Timeout status is notified to upper layers.
- API gets unblocked with status UCI timeout. Same status is returned to Application.
- Applications post message to Exception handling task or start recovery timer.
- Application tasks waits for recovery.

- Exception handling task/ or in the context of timeout, Recovery API is invoked. Recovery API performs certain operations are shown in the flow diagram.
- After recovery All session state context is deleted, and application task is restarted.

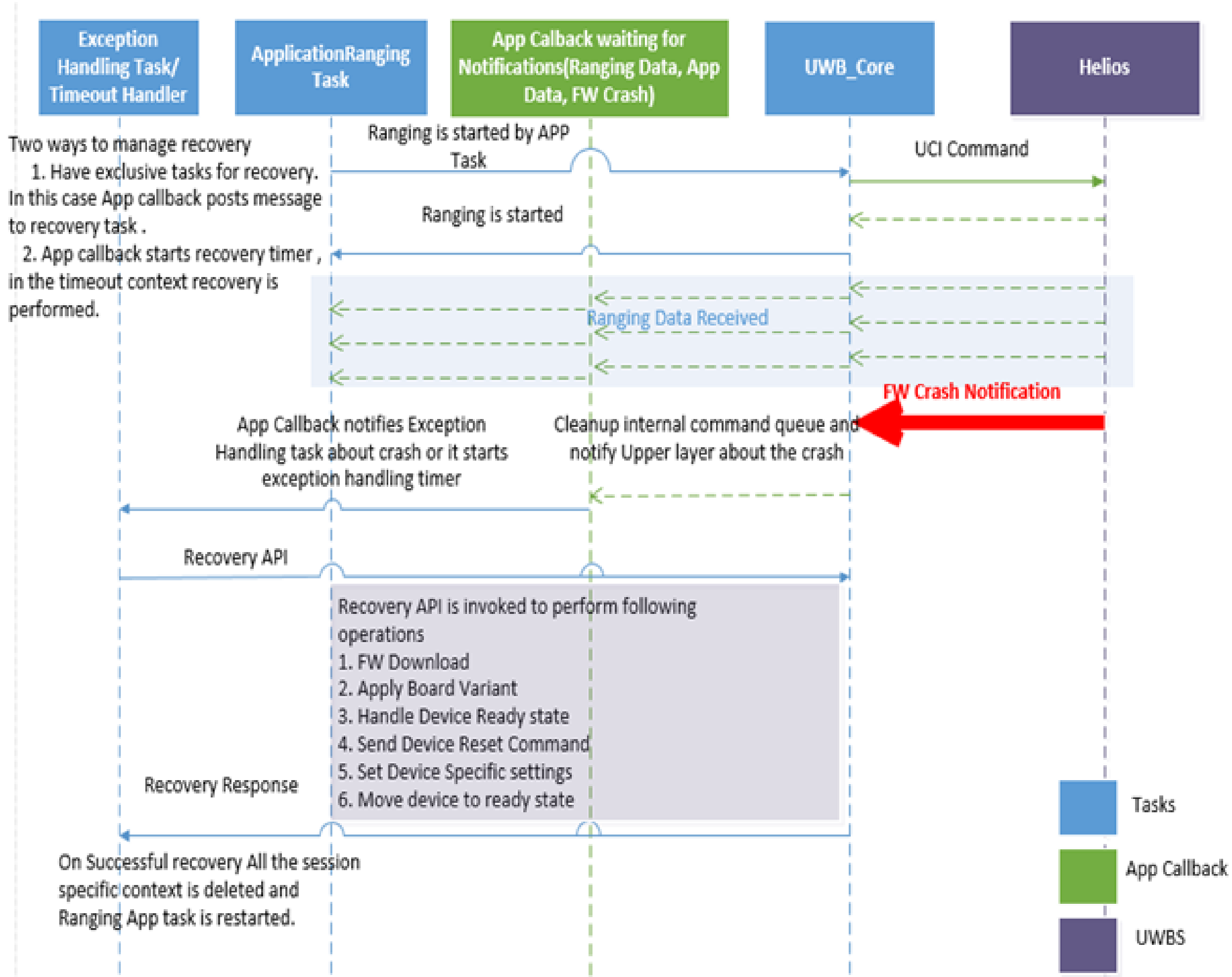



### 1.6.6 Recovery : FW Crash Handling from MW

Following steps are performed from MW to handle FW Crash

- During active ranging session FW can be crashed.
- FW Crash notification is sent to upper layers. (App callback)
- App callback posts message to Exception handling task or starts recovery timer.
- Application tasks waits for recovery.
- Exception handling tasks/ or in the context of timeout Recovery API is invoked. Recovery API performs certain operations are shown in the flow diagram.

After recovery All session state context is deleted, and application task is restarted


Exception Handling Task/ Timeout Handler
ApplicationRanging Task
App Calback waiting for Notifications(Ranging Data, App Data, FW Crash)
UWB_Core
Helios
Two ways to manage recovery
1. Have exclusive tasks for recovery. In this case App callback posts message to recovery task .
2. App callback starts recovery timer , in the timeout context recovery is performed.
Ranging is started by APP Task
UCI Command
Ranging is started
Ranging Data Received
FW Crash Notification
App Callback notifies Exception Handling task about crash or it starts exception handling timer
Cleanup internal command queue and notify Upper layer about the crash
Recovery API
Recovery API is invoked to perform following operations
1. FW Download
2. Apply Board Variant
3. Handle Device Ready state
4. Send Device Reset Command
5. Set Device Specific settings
6. Move device to ready state
Recovery Response
On Successful recovery All the session specific context is deleted and Ranging App task is restarted.
Tasks
App Callback
UWBS

CHAPTER

TWO

# CHANGES & RELEASE HISTORY (SR150)

## 2.1 Build v04.06.05 (current)

### 2.1.1 Scope of Build

- Added Support to increase maximum FOV to 90° by adding Extra Param in Calibration command.
- Added SNR fields support in 'CCC' ranging Notification.
- Disabled UL-TDOA Tag feature from MW for SR150 variant.
- Vendor App Config *ULTDOA_MAC_FRAME_FORMAT* disabled for the SR150.
- Added Support for Vendor Specific extensions in `SESSION_INFO_NTF` for UL-TDoA-Anchor.
- Added Android jetpack API CONFIG_ID support for Config Ids `CONFIG_ID_2` and `CONFIG_ID_3`.
- App config `CSA_ACTIVE_RR_CONFIG` support along with documentation is removed.
- Added support for CSA Session Type - *UWBD_CSA_SESSION*.
- Macro `RX_LOG_MAX_NUMBER_OF_BYTES` is added to provide support for configurable logger prints in case of RX data.
- Added Q-format conversion for signed integer
- Added support to print RSSI fields in Q7.1 format for DSTWR and DLTDOA-Tag feature
- Added MRR reporting in CCC Ranging.
  - Added `CSA_FINAL_DATA2_CONFIG` Vendor App Config to configure the sending of Final Data 2 from responder to Initiator.
  - Added `blockIndex` in `phAliroRangingData_t`.
  - Added `SESSION_SET_LOCALIZATION_ZONE_CMD/RSP` to set the Localization Zone.

- OID of EXT_UCI_MSG_UWB_WLAN_COEX_NTF has been updated.
- MW updated to configure PDoA configs for single channel.
  - Can be configured by selecting the desired channel number in UWB_DeviceConfig_SR1XX.h.

### 2.1.2 MW Changes

- Renamed EXT_UCI_MSG_UWB_WIFI_COEX_MAX_ACTIVE_GRANT_DUARTION_EXCEEDED_WAR_NTF to EXT_UCI_MSG_UWB_WLAN_COEX_NTF.
- updated MAX_NO_OF_CCC_SNR_MEASUREMENTS from 0x03 to 0x04.
- Reduced the UCI_CMD_MAX_RETRY_COUNT to 10 in order to retry within UWB_MAX_DEV_MGMT_RSP_TIMEOUT.

### 2.1.3 API Changes

#### New API added

- Added New API *UwbApi_SessionSetLocZone()* to set the Session Localization Zone.

#### API removed:

#### API updated:

#### Structure Changes:

- Updated Structure phCccRangingData_t:
  - Added parameters: noOfSnrMeasurements and cccSnrMeasurements.
- Added structure phCccSnrMeasurements_t for CCC SNR measurements.
- Added structure phPdoaTableDef_t to lists out the pdoa table define config
- Updated structure phDeviceConfigData_t
  - added parameter:
    - A new parameter of type phPdoaTableDef_t is added
- Added structure phTdoaRangingVsData_t for vendor data handling UL-TDOA Anchor.
- Updated structure *VENDORSPECIFIC_MEAS*
  - Added parameter:

    - A new parameter of type phTdoaRangingVsData_t is added

- Updated structure phRangingMesrTdoa_t
  - Updated parameter:
    - In parameter rx_timestamp Datatype of parameter is changesd from `uint64_t` to array of `uint8_t`.
    - In parameter ul_tdoa_device_id Datatype of parameter is changesd from `uint64_t` to array of `uint8_t`.
    - In parameter tx_timestamp Datatype of parameter is changesd from `uint64_t` to array of `uint8_t`.
  - Removed parameter:
    - noOfPdoaMeasures
    - rssi_rx1
    - rssi_rx2
    - pdoaFirst
    - pdoaFirstIndex
    - pdoaSecond
    - pdoaSecondIndex
- Updated structure phRangingData_t
  - In parameter authenticationTag Datatype of parameter is changesd from `uint32_t` to array of `uint8_t`.
- Updated structure phTdoaRangingVsData_t
  - A new field vendorExtLength is added.
- Updated structure UwbPhoneConfigData_t
  - renamed profile_id to config_id.
- Updated structure UwbDeviceConfigData_t
  - renamed supported_profile_ids to supported_config_ids.
- Renamed phCccRangingData_t to phAliroRangingData_t.
- Added blockIndex in phAliroRangingData_t.
- Added phSessionSetLocZone_t to set the Session Localization Zone.

#### Enum changes:

- Updated enum `eDeviceConfig`:
  - Added parameters:
    - *PDOA_CALIB_TABLE_DEFINE*
- Updated enum *kUWBAntCfgRxMode_t*:
  - Added Extented parameter :
    - *AOA_ANTENNAS_PDOA_CALIB_EXTENDED_SUPPORT* for PDOA 90 FoV.
- renamed `UWB_ProfileID_t` to `eUWB_ProfileID_t`.
- Updated `eSessionType` with a new Session Type for Aliro session *UWBD_CSA_SESSION*.
- Updated enum `eVendorAppConfig`
  - Added new Vendor App Config *CSA_FINAL_DATA2_CONFIG* to configure the sending of Final Data 2 from responder to Initiator.

### 2.1.4 Folder Restructuring

### 2.1.5 New Features Added

### 2.1.6 Features Removed

### 2.1.7 Features Updated

### 2.1.8 Build system changes

- `UWBFTR_UL_TDoA_Tag` feature macro support is disabled for SR150 UWBS.

### 2.1.9 Demo updates

### 2.1.10 Folder Restructuring

## 2.2 Release v04.05.07 (SR150)

### 2.2.1 Scope of Build

- FIRA-461 Added support for Session Handle.

- FIRA-479 Mandatory application configurations are updated.
- FIRA-459 UCI generic specification configurations updated in contention based ranging.
- FIRA-1080 Merging commands to configure active ranging rounds and responder MAC address list for DT-Anchors.
- FIRA-490 UCI flow control in OWR for AoA measurement final.
- FIRA-1138 UCI Spec Sequence Number field size correction in `SESSION_DATA_TRANSFER_STATUS_NTF`.
- FIRA-1037 Corrections and clarifications on status codes.
- FIRA-1058 Reporting of slot index in `SESSION_INFO_NTF` is updated.
- RSSI Measurement support added for CCC Range Rata Notification.
- New Vendor App configuration added to swap the antenna pair for RFM reception.
- New Vendor app config has been added to configure the near and far proximity bounds to add the devices in RML.
- FreeRTOS task exit bug fixed
- New Vendor app config has been added to configure `RFRAME_LOG_NTF` along with handling of same notification.
- Data transfer Phase Configuration response and notification handling has been moved under Session Management handlers.
- New Vendor app config has been added to configure CSA MAC mode and CSA Active Ranging Round.
- Additional modes added in RX Antenna Configurations for CSA ToA and CSA AoA modes.
- Added Support of 2D AoA Fov Feature
- Added Support for `DT_TAG_BLOCK_SKIPPING_ID` in device capability, to indicate the block skipping capability for DT-Tag.
- Added Support for `PSDU_LENGTH_SUPPORT_ID` in device capability, to indicate the supported PSDU length.

### 2.2.2 MW Changes

- Task exit logic for FreeRTOS and Native builds is cleaned up. FreeRTOS tasks will be suspended and native tasks will exit. All tasks are deleted through *`UwbApi_ShutDown()`*.
- For EmbedLinux Platform default baud rate is set to 1Mbps.
- For EmbedLinux Platform, PCTT testing is supported at 1Mbps baud rate.

### 2.2.3 API Changes

#### New API added

- Added support of Set/Get vendor app configuration parameters via *UwbApi_SetVendorAppConfigs()* and *UwbApi_GetVendorAppConfigs()* API.

#### API removed:

- Function `UwbApi_SetInitiatorAnchorRRRDMlist()` is removed.

#### API updated:

- Calibration parameters updated in `setDefaultCoreConfigs()` API for SR1XX and SR2XX.
- Following APIs have been updated to only set FIRA-specific application configurations
  - *UwbApi_SetAppConfig()*
  - *UwbApi_SetAppConfigMultipleParams()*
  - *UwbApi_GetAppConfig()*
  - *UwbApi_GetAppConfigMultipleParams()*
- Function *UwbApi_SetDebugParams()* is updated to use set vendor app configuration command instead of *UwbApi_SetAppConfig()* API.
- Function *UwbApi_GetDebugParams()* is updated to use get vendor app configuration command instead of *UwbApi_GetAppConfig()* API.
- Api `UwbApi_DoCalibration()` is renamed to *UwbApi_DoVcoPllCalibration()* for VCO PLL calibration in factory FW. - Removed function argument `paramId` from *UwbApi_DoVcoPllCalibration()*
- All APIs are updated to handle argument `sessionHandle` instead of `sessionId`
- New function parameter `macAddressingMode` added in *UwbApi_UpdateActiveRoundsAnchor()* of type *UWB_MacAddressMode_t*
- API signature updated for `UwbApi_TestConnectivity()` for ESE Connectivity check in factory FW for SR100T:
  - Argument changed from `uint8_t *wtxCount` to `SeConnectivityStatus_t *ConnectivityStatus`.
- API signature of *UwbApi_ConfigureData_iOS()* is updated :
  - To accept `UWB_VendorAppParams_List_t` instead of `UWB_AppParams_List_t`.

  - To pass the number of `noOfVendorAppParams` instead of `noOfAppParams`.

- API signature of *UwbApi_ConfigureData_Android()* is updated :
  - To accept `UWB_VendorAppParams_List_t` instead of `UWB_AppParams_List_t`.
  - To pass the number of `noOfVendorAppParams` instead of `noOfAppParams`.
- APIs *UwbApi_ConfigureData_iOS()* and *UwbApi_ConfigureData_Android()* have been moved to it's respective UWBD proprietary file.

### Structure Changes:

- Updated structure `phUwbDataPkt_t`
  - Removed parameters: `dst_endpoint`
  - Size of `sequence_number` changed to 2 bytes
- Updated structure `phUwbRcvDataPkt_t`
  - Removed parameters: `dst_endpoint` and `src_endpoint`
  - Size of `sequence_number` changed to 2 bytes
- Updated structure `phRangingParams_t`
  - Removed parameters: `noOfControlees`, `dstMacAddr`
  - Added parameters: `scheduledMode`, `rangingRoundUsage`
- Removed Structure `phUwbRRRDMList_t`
- Updated structure `phActiveRoundsConfig_t`
  - Removed parameters: `deviceRole`
  - Added parameters: `rangingRole`, `noofResponders`, `responderMacAddressList`, `responderSlotScheduling`
- Updated structure `phUwbDevInfo_t` to support vendor specific information
  - Added parameters: `maxPpmValue`, `txPowerValue`
- Updated structure `phSeDoBindStatus_t` to support ESE error specific information for SN110
  - Added parameters: `se_instruction_code` and `se_error_status`
- Updated structure `phSeGetBindingStatus_t` for supporting ESE error specific information for SN110
  - Added parameters: `se_instruction_code` and `se_error_status`
- Updated structure `phMulticastControleeListContext_t`

- Removed parameters: `message_control`
- Updated parameters: `action` for `0x03` value to add the Controlee with its 32-bit Sub-Session Key to the multicast list.
- Size of parameters `SubSessionKey` changed to hold a max of 32B key.

- Updated structure `phRangingMesr_t`
  - Moved `rssi_rx1`: to new structure `phOneWayRangingVsData_t` as `rssi`
  - Moved `rssi_rx2`: to new structure `phOneWayRangingVsData_t` as `rssi`
  - Moved `distance_2`: to new structure `phTwoWayRangingVsData_t` as `distance_2`
  - Moved `pdoaFirst`: to new structure `phAoaPdoaMesr_t` as `pdoa`
  - Moved `pdoaFirstIndex`: to new structure `phAoaPdoaMesr_t` as `pdoaIndex`
  - Moved `pdoaSecond`: to new structure `phAoaPdoaMesr_t` as `pdoa`
  - Moved `pdoaSecondIndex`: to new structure `phAoaPdoaMesr_t` as `pdoaIndex`
- Added structure `phRxInfoMesr_t` for vendor specific Rx Antenna Info for AoA Measurements
- Added structure `phRxInfoDebugNtf_t` for vendor specific Rx Antenna Info for Debug Notifications
- Added structure `phAoaPdoaMesr_t` for vendor specific Rx Antenna Info for AoA / PDoA measurements per RX
- Added structure `phSnrPathIndexMesr_t` for vendor specific Rx Antenna Info for SNRFirst / SNRMain / FirstIndex Main Index measurements per RX entry
- Added structure `phTwoWayRangingVsData_t` for vendor specific information for TWR
- Added structure `phOneWayRangingVsData_t` for vendor speicifc information for OWR
- Added structure(union) *`VENDORSPECIFIC_MEAS`* for vendor specific Information of TWR ranging and TDoA ranging
- Updated structure `phRangingData_t`
  - Added parameters: `vs_length`, `vs_data_type`, `vs_data`, `authInfoPrsen`, `authenticationTag`
- Updated structure `phDataTxPhaseConfig_t`
  - Renamed parameter `dtpcm_SessionID` to `dtpcm_SessionHandle`.
- Updated structures `phMulticastControleeListNtfContext_t`
  - Included `phMulticastControleeStatusList_t` for the Multicast controlee list
- Added structure `phMulticastControleeStatusList_t` for storing Multicast Controlee Status List Context

- Notification handling added for *UWBD_WIFI_COEX_IND_NTF* and UWB_WIFI_COEX_MAX_ACTIVE_GRANT_DUARTION_EXCEEDED_WAR_NTF
- Updated structure phUwbDataTransmit_t
  - Renamed parameters:
    - sessionHandle to transmitNtf_sessionHandle
    - sequence_number to transmitNtf_sequence_number
    - status to transmitNtf_status
    - txcount to transmitNtf_txcount
- Updated Structure phCccRangingData_t:
  - Added parameters: noOfRssiMeasurements and cccRssiMeasurements.
  - Renamed sessionId to sessionHandle.
- Added structure phCccRssiMeasurements_t for CCC RSSI measurements.
- Added structure SeConnectivityStatus_t to check the connectivity between UWBS and SE with SUS applet installed in SE
- Updated structure phCalibRespStatus_t
  - Updated parameters:
    - Datatype of parameter calibState changed from uint8_t to *eCalibState*
- Added structure UWB_VendorAppParams_List_t for vendor specific config parameters
  - Structure phUwbCapInfo_t has been updated with the following parameters:
    - tagBlockSkipping to indicate the block skipping capability for DT-Tag.
    - psduLengthSupport to indicate the supported PSDU length.

### Enum changes:

- Updated enum eAppConfig
  - Removed all Extended Application Configurations parameters
  - Added parameters:
    - *DEV_MAC_ADDRESS*
    - *DEVICE_ROLE*
    - *SESSION_TIME_BASE*
    - *SESSION_TIME_BASE*

    - *DL_TDOA_RESPONDER_TOF*
    - *NO_OF_CONTROLEES*
    - *DST_MAC_ADDRESS*
    - *APPLICATION_DATA_ENDPOINT*
    - *RX_GPIO_ANTENNA_SELECTION*
    - *UWB_DEVICE_TYPE*
    - *DEV_MAC_ADDRESS*
    - *DEVICE_ROLE*

- Moved app configs to Vendor configs from `eAppConfig` to `eVendorAppConfig`:
    - `CIR_CAPTURE_MODE` to *CIR_CAPTURE_MODE*
    - `MAC_PAYLOAD_ENCRYPTION` to *MAC_PAYLOAD_ENCRYPTION*
    - `RX_ANTENNA_POLARIZATION_OPTION` to *RX_ANTENNA_POLARIZATION_OPTION*
    - `SESSION_SYNC_ATTEMPTS` to *SESSION_SYNC_ATTEMPTS*
    - `SESSION_SCHED_ATTEMPTS` to *SESSION_SCHED_ATTEMPTS*
    - `SCHED_STATUS_NTF` to *SCHED_STATUS_NTF*
    - `TX_PEAK_POWER_DELTA_FCC` to *TX_POWER_DELTA_FCC*
    - `TX_POWER_TEMP_COMPENSATION` to *TX_POWER_TEMP_COMPENSATION*
    - `WIFI_COEX_MAX_TOLERANCE_COUNT` to *WIFI_COEX_MAX_TOLERANCE_COUNT*
    - `ADAPTIVE_HOPPING_THRESHOLD` to *ADAPTIVE_HOPPING_THRESHOLD*
    - `AUTHENTICITY_TAG` to *AUTHENTICITY_TAG*
    - `RX_NBIC_CONFIG` to *RX_NBIC_CONFIG*
    - `MAC_CFG` to *MAC_CFG*
    - `INBAND_DATA_TX_BLOCK_SIZE` to *SESSION_INBAND_DATA_TX_BLOCKS*
    - `INBAND_DATA_RX_BLOCK_SIZE` to *SESSION_INBAND_DATA_RX_BLOCKS*
    - `ANTENNAE_CONFIGURATION_TX` to *ANTENNAE_CONFIGURATION_TX*
    - `ANTENNAE_CONFIGURATION_RX` to *ANTENNAE_CONFIGURATION_RX*
    - `ANTENNAE_SCAN_CONFIGURATION` to *ANTENNAE_SCAN_CONFIGURATION*
    - `ULTDOA_MAC_FRAME_FORMAT` to *ULTDOA_MAC_FRAME_FORMAT*
    - `WRAPPED_RDS` to `WRAPPED_RDS`

- Renamed parameters:

        - RANGE_DATA_NTF_BOUND_AOA to *AOA_BOUND_CONFIG*
        - RNG_DATA_NTF_PROXIMITY_FAR to *FAR_PROXIMITY_CONFIG*
        - RNG_DATA_NTF_PROXIMITY_NEAR to *NEAR_PROXIMITY_CONFIG*
        - RNG_DATA_NTF to *SESSION_INFO_NTF*
    - updated parameters:
        - Tag ID of RESPONDER_SLOT_INDEX has been updated to 0xA2 from 0x1E.
- Updated enum UWB_SR1XX_DBG_CFG_t
    - Added new tag IDs
- Updated enum eNotificationType
    - Added parameters: *UWBD_CIR_DATA_NTF*
    - Removed parameters: UWBD_CIR0_DATA, UWBD_CIR1_DATA
- Updated enum eVendorAppConfig
    - Added parameters:
        - *SWAP_ANTENNA_PAIR_3D_AOA* to swap the antenna pair for RFM reception.
        - *RML_PROXIMITY_CONFIG* to define near and far proximity
        - *RAN_MULTIPLIER*
        - *STS_LAST_INDEX_USED*
        - *CIR_LOG_NTF*
        - *PSDU_LOG_NTF*
        - *RSSI_AVG_FILT_CNT*
        - *SESSION_TIME_BASE*
        - *DL_TDOA_RESPONDER_TOF*
        - *DATA_TRANSFER_TX_STATUS_CONFIG*
        - *TX_ADAPTIVE_PAYLOAD_POWER*
        - *SWAP_ANTENNA_PAIR_3D_AOA*
        - *RML_PROXIMITY_CONFIG*
        - *CSA_MAC_MODE*
        - CSA_ACTIVE_RR_CONFIG
        - RESPONDER_SLOT_INDEX
- Updated enum *eSESSION_STATUS_REASON_CODES_t*

- Added parameter:
  - *UWB_SESSION_INVALID_ANTENNA_PAIR_SWAP_CONFIGURATION*
  - *UWB_SESSION_CSA_INVALID_CFG*
- Updated enum `UWB_SR1XX_DBG_CFG_t`:
  - Added parameters:
    - *kUWB_SR1XX_DBG_CFG_DATA_LOGGER_NTF*
    - *kUWB_SR1XX_DBG_CFG_TEST_CONTENTION_RANGING_FEATURE*
    - *kUWB_SR1XX_DBG_CFG_CIR_CAPTURE_WINDOW*
    - *kUWB_SR1XX_DBG_CFG_RANGING_TIMESTAMP_NTF*
    - *kUWB_SR1XX_DBG_CFG_THREAD_SECURE*
    - *kUWB_SR1XX_DBG_CFG_THREAD_SECURE_ISR*
    - *kUWB_SR1XX_DBG_CFG_THREAD_NON_SECURE_ISR*
    - *kUWB_SR1XX_DBG_CFG_THREAD_SHELL*
    - *kUWB_SR1XX_DBG_CFG_THREAD_PHY*
    - *kUWB_SR1XX_DBG_CFG_THREAD_RANGING*
    - *kUWB_SR1XX_DBG_CFG_THREAD_SECURE_ELEMENT*
    - *kUWB_SR1XX_DBG_CFG_THREAD_UWB_WLAN_COEX*
- Updated enum `eNotificationType`:
  - Added parameters:
    - *UWBD_CIR_DATA_NTF*
  - Removed parameters:
    - `UWBD_CIR0_DATA`
    - `UWBD_CIR1_DATA`
- Updated enum `UWB_Session_InfoNtf_t`:
  - Renamed parameters:
    - `kUWB_DisableRange_Data_Ntf` to *kUWB_DisableSession_Info_Ntf*
    - `kUWB_EnableRange_Data_Ntf` to *kUWB_EnableSession_Info_Ntf*
    - `kUWB_RangeData_Ntf_Proximity` to *kUWB_SessionInfo_Ntf_Proximity*
    - `kUWB_RangeData_Ntf_AOABounds` to *kUWB_SessionInfo_Ntf_AOABounds*

- kUWB_RangeData_Ntf_AOABounds_n_Proximity to *kUWB_SessionInfo_Ntf_AOABounds_n_Proximity*
- kUWB_RangeData_Ntf_Enter_Leave_Proximity to *kUWB_SessionInfo_Ntf_Enter_Leave_Proximity*
- kUWB_RangeData_Ntf_Enter_Leave_AOABounds to *kUWB_SessionInfo_Ntf_Enter_Leave_AOABounds*
- kUWB_RangeData_Ntf_Enter_Leave_AOABounds_n_Proximity to *kUWB_SessionInfo_Ntf_Enter_Leave_AOABounds_n_Proximity*

- Updated enum *eSESSION_STATUS_REASON_CODES_t*:
  - Added parameters:
    - *UWB_SESSION_DT_ANCHOR_RANGING_ROUNDS_NOT_CONFIGURED*
    - *UWB_SESSION_DT_TAG_RANGING_ROUNDS_NOT_CONFIGURED*
    - *UWB_SESSION_ERROR_INVALID_ANTENNA_CFG*

      **Note:** When CSA_ACTIVE_RR_CONFIG config is set to 0x02 and configured antenna mode is 2 or 3 or 4 then UWBS shall report with this reason code.

    - *UWB_SESSION_HUS_NOT_ENOUGH_SLOTS* to indicate if the UWBS detects that the number of cumulated slots exceeds the number of slots per ranging round of the primary session.
  - Updated parameters:
    - *UWB_SESSION_HUS_CFP_PHASE_TOO_SHORT* id is updated to 0x27.
    - *UWB_SESSION_HUS_CAP_PHASE_TOO_SHORT* id is updated to 0x28.
- Updated enum eSessionType:
  - Added parameters:
    - *UWBD_CCC_SESSION*
- Updated enum eDeviceConfig:
  - Added parameters:
    - *RX_GPIO_ANTENNA_SELECTION*
    - *WIFI_COEX_UART_USER_CFG*
    - *CLK_CONFIG_CTRL*
    - AOA_MODE
    - *CLOCK_PRESENT_WAITING_TIME*

    - DDFS_CONFIG_PER_PULSE_SHAPE

  - Removed parameters:

    - RF_XTAL_CAP
    - MANUAL_TX_POW_CTRL
    - PAPPPA_CALIB_CTRL
    - AOA_ANTENNAE_PDOA_CALIB
    - SESSION_SCHED_ATTEMPTS
    - AOA_ANTENNAE_PDOA_CALIB
    - AOA_ANTENNAE_MULTIPOINT_CALIB
    - RX_ANT_DELAY_CALIB
    - PDOA_OFFSET_CALIB
    - AOA_THRESHOLD_PDOA
    - RSSI_CALIB_CONSTANT_HIGH_PWR
    - RSSI_CALIB_CONSTANT_LOW_PWR
    - SNR_CALIB_CONSTANT_PER_ANTENNA
    - SNR_CALIB_CONSTANT_PER_PAIR
    - TX_POWER_PER_ANTENNA
    - TX_TEMPERATURE_COMP_PER_ANTENNA

- Updated enum `eDeviceConfig` for SR1XX:

  - Added parameters:

    - *RF_CLK_ACCURACY_CALIB*
    - *RX_ANT_DELAY_CALIB*
    - *PDOA_OFFSET_CALIB*
    - *TX_POWER_PER_ANTENNA*
    - *MANUAL_TX_POW_CTRL*
    - *PA_PPA_CALIB_CTRL*
    - *AOA_ANTENNAS_PDOA_CALIB*
    - *AOA_ANTENNAS_MULTIPOINT_CALIB*
    - *PDOA_MANUFACT_ZERO_OFFSET_CALIB*
    - *AOA_THRESHOLD_PDOA*

        - *TX_TEMPERATURE_COMP_PER_ANTENNA*
        - *SNR_CALIB_CONSTANT_PER_ANTENNA*
        - *RSSI_CALIB_CONSTANT_HIGH_PWR*
        - *RSSI_CALIB_CONSTANT_LOW_PWR*

- Updated enum *kUWBAntCfgRxMode_t*:
    - Added parameters:
        - *kUWBAntCfgRxMode_Radar_Mode*
        - *kUWBAntCfgRxMode_ToA_Rfm_Mode*
        - *kUWBAntCfgRxMode_ToA_Rfm_Mode*
        - *kUWBAntCfgRxMode_CSA_ToA_Mode*
        - *kUWBAntCfgRxMode_CSA_ToA_Mode*
    - Renamed parameters:
        - `kUWBAntCfgRxMode_ToF_Mode` to *kUWBAntCfgRxMode_ToA_Mode*
- Updated enum `UWB_RangingRoundUsage_t`:
    - Added parameters:
        - *kUWB_RangingRoundUsage_DTx* for Data transfer mode
- Removed enum `eDoCalibParam`.
- Added enum *eCalibState* with values of calibration status
- New Application Configuration parameter `SESSION_DATA_TRANSFER_STATUS_NTF_CONFIG` added to configure DATA TRANSFER STATUS Notification
- Updated enum *ANTENNAE_CONFIGURATION_RX* for new configuration modes `0x05` for ToA mode for CSA and `0x06` for AoA mode for CSA.

### 2.2.4 Folder Restructuring

### 2.2.5 New Features Added

- New UCI Generic status code `UCI_STATUS_UNKNOWN` is added.
- Extended parameters Tag-IDs of get device info response are updated to support new vendor specific information.
- In DL-TDOA, the Anchor CFO Field and CFO Field value are converted from Q511 to Q610 format.

- GET_CAPS_INFO param extented TAG-ID changed from 0xE0 to 0xE3.
- CIR0 & CIR1 LOG notification are combined into one notification.
- CIR LOG NTF & PSDU_LOG_NTF are now moved to new vendor proprietary group GID
- New GID added for UCI_GID_INTERNAL_GROUP.
- New Vendor App configuration parameter is added for configuring RFRAME_LOG_NTF
- DBG_RFRAME_LOG_NTF & RANGING_TIMESTAMP_NTF are now moved to new Internal group GID
- Added support of new vendor app config parameter ENABLE_FOV & AZIMUTH_FIELD_OF_VIEW for 2D AoA FoV.
- Added aoaFoVFlag of 1 Octet in Vendor Specific Extension for both OWR and TWR Range data Notification

### 2.2.6 Features Removed

- In DLTdoa anchor, command UCI_MSG_SET_INITIATOR_ANCHOR_RR_RDM_LIST has been removed
- Removed support for deprecated command SEND_BLINK_APP_DATA
- Removed proprietary parameters CALIB_DATA_STORAGE_OPTION, AOA_FINE_CALIB_CTRL, PDOA_CALIB_TABLE_DEFINE and RSSI_AVG_FILT_CNT.
- Extented parameter TAG-ID 0xE3 is removed for get device info response to support new vendor specific information.
- Extented parameter TAG-ID 0xE4 is removed for Debug configurations.

### 2.2.7 Features Updated

- Moved DATA_CREDIT_NTF to RANGE_MANAGEMENT group (0x02). Updated to use common OID for all UWB devices - 0x04
- Moved DATA_TRANSMIT_NTF to RANGE_MANAGEMENT group (0x02). Updated to use common OID for all UWB devices - 0x05
- Updated GID/OID of SESSION_QUERY_DATA_SIZE_IN_RANGING to 0x010B.
- Updated GID/OID of SET_HUS_CONTROLLER_SESSION_CONFIG to 0x010C.
- Updated GID/OID of SET_HUS_CONTROLEE_SESSION_CONFIG to 0x010D.
- Updated GID/OID of *UwbApi_SetCalibration()* to 0x0F21.
- Updated GID/OID of *UwbApi_GetCalibration()* to 0x0F22.

- Size included for DT-Anchor location in WGS-84 coordinate system was 10 Octets - updated to 12 octets. In a relative coordinate system it was 12 octets - updated to 10 octets.
- Updated GID/OID of `GET_ALL_UWB_SESSIONS` from `0x0E0F` to `0x0F02`
- Updated app config param `HOP_MODE_KEY` description and length from 16 bytes to 4 bytes in `eAppConfig` .
- Updated app config param *`HOPPING_MODE`* description in `eAppConfig` .

### 2.2.8 Build system changes

- `OWR_AOA_advertiser` feature is excluded for the SR100T UWBS.
- `UWBFTR_OWR_AOA` feature macro support is removed and it is enabled by default for SR1XX and SR2XX.
- `UWBFTR_FIRATestModes` feature macro support is removed and test mode code is enabled by default.
- `UWBFTR_WIFI_COEX` feature macro support is removed and it is enabled by default for SR150 and SR160 .

### 2.2.9 Demo updates

- Secure element specific demos: Added support to select different logical channels for secure element from the application. See *`Se_API_Init()`*.
- All the demos have been updated for `sessionHandle`.
- In case of sessions where in-band data transfer is possible, MCTT-PCTT demo changes the default value of data transfer related vendor application configuration parameters for TX & RX Block config.
- Demo TEST TX/RX updated to support HPRF configuration with PSDU size 1023.
- New Demo Section 4.20 *Demo nearby interaction Client* added for SR150 .
- DLTDOA Anchor demos updated with the App config `DLTDOA_TX_TIMESTAMP_CONF` with value 2 (Local Time Base with 64Bit timestamp value).
- DataTransfer demos updated by changing *`ANTENNAE_CONFIGURATION_RX`* mode of configuration to *`kUWBAntCfgRxMode_ToA_Mode`* mode.
- Application Configuration parameter *`UL_TDOA_TX_INTERVAL`* added in ultdoa tag demo *TDOA Tag* with value 10.
- MCTT demo has been update to apply the calibration parameters to `channel 5` and `channel 9`.

- MCTT demo is updated to calibrate TX_POWER_PER_ANTENNA instead of MANUAL_TX_POW_CTRL for OTP_TX_POWER_ID.
- following demos are renamed :

  Demo demo_ccc_controlee renamed to demo_csa_controlee

  - Refer Section 4.28 *Demo CSA Controlee* .

  Demo demo_dltdoa_anchor1 renamed to demo_dltdoa_initiator

  - Refer Section 4.15 *SR1XX DLTDOA Ranging Initiator* .

  Demo demo_dltdoa_anchor2 renamed to demo_dltdoa_responder

  - Refer Section 4.16 *SR1XX DLTDOA Ranging Responder* .

  Demo demo_dltdoa_receiver renamed to demo_dltdoa_tag

  - Refer Section 4.17 *SR1XX DLTDOA Ranging Tag* .

  Demo demo_data_transfer_rx renamed to demo_inband_data_transfer_rx

  - Refer Section 4.22 *SR150 In Band Data Transfer Rx* .

  Demo demo_data_transfer_tx renamed to demo_inband_data_transfer_tx

  - Refer Section 4.23 *SR150 IN Band Data Transfer Tx* .

- following demos are removed :
  - Demo demo_ranging_multisession .
  - Demo demo_semslite_FiRaLite_A739_Run4 .
  - Demo demo_semslite_FiRaLite_A739_Run4_Run6 .
  - Demo demo_semslite_FiRaLite_A739_Run5_Run6 .

## 2.3 Release v04.04.05 (SR150)

### 2.3.1 Scope of Build

1) PSDU Data rate 850kbit support for SR150.
2) Hybrid Scheduling support added
3) Data transfer phase configuration support added.
4) Get device capability info as per latest FIRA Specification
5) Removed Advertisement mode Observer support

6) New status for negative distance *UWBAPI_STATUS_OK_NEGATIVE_DISTANCE_REPORT*
7) OTP read write for calibration value of *OTP_PDOA_MFG_ZERO_OFFSET_CALIB* and *OTP_AOA_ANT_MULTIPOINT_CALIB*.
8) Reason codes are updated for *SESSION_STATUS_NTF* notification.
9) Session type *UWBD_RANGING_WITH_INBAND_DATA_TRANSFER* support
10) New Command response support added for *SESSION_QUERY_DATA_SIZE_IN_RANGING*
11) Default logging speed changed from 115200 Mbps to 3 Mbps for all Host platform.
12) Default logging speed set to 1150200 Mbps for Nordic platfrom.
13) *DATA_TRANSFER_MODE* renamed to *LINK_LAYER_MODE*
14) enum *dataTransferModes* renamed to *linkLayerModes*
15) *Data_Transfer_Mode_Raw* renamed to *Link_Layer_Mode_Bypass*

### 2.3.2 API Changes

- **The API `UwbApi_SetHusSession` for configuring Hybrid Session was modified for controller and controlee as follows :**
    - `UwbApi_SetControllerHusSession` for controller.
    - `UwbApi_SetControleeHusSession` for controlee.
- New API added *UwbApi_SessionQueryDataSize* to get the Max Application Data size in a single Ranging Round.
- New API added *UwbApi_SessionDataTxPhaseConfig* to set the Data transfer phase configuration.
- New API added `UwbApi_SetRdsParam` to set the RDS parameters for given UWB session.

### 2.3.3 Folder Restructuring

### 2.3.4 New Features Added

- **SR1XX:**

    1) **Added support for new session type `UWBD_RANGING_WITH_INBAND_DATA_TRANSFER`, and updated the** demo data transfer tx and rx use the same session type.

    2) Added new status for negative distance UWBAPI_STATUS_OK_NEGATIVE_DISTANCE_REPORT.

3)**Updated get device capability type `phUwbCapInfo_t`, for new and updated controlee device capability.**

IDs changed for all the below mentioned configs. All other IDs also adjusted: `MAX_MESSAGE_SIZE`, `MAX_DATA_PACKET_PAYLOAD_SIZE`. Extra paratmers added: `SUSPEND_RANGING` `SESSION_KEY_LENGTH` Length changed for the below parameters: `DEVICE_ROLES` `RANGING_METHOD`

4) Default logging speed changed from 115200 Mbps to 3 Mbps.

5) Added New Vendor Specific STS type `0xA0` as `kUWB_StsConfig_ShenzengTongSts`.

- SR150:

1) Added support for OTP read write for calibration value of *OTP_PDOA_MFG_ZERO_OFFSET_CALIB* and *OTP_AOA_ANT_MULTIPOINT_CALIB*.

2) As per new UCI Generic Specification reason code are updated for ` SESSION_STATUS_NTF` notification.

3) Application Configuration parameter 'RANGING_INTERVAL' is renamed to 'RANGING_DURATION'.

4) PSDU Data rate 850kbit support for SR150.

5) Added support for Hybrid Scheduled Ranging and added Hybrid Demo ranging controller and controlee

6) Default logging speed changed from 115200 Mbps to 3 Mbps for all Host platform.

7) Default logging speed set to 1150200 Mbps for Nordic platfrom.

8) Addeds Support for new proprietary command and notification `SESSION_SET_RDS_PARAM`.

•**SR160:**

1) Added support for Radar App configuration value of `RADAR_FCC_TEST_MODE`.

2) Added new status for RADAR FCC TIMEOUT `UWB_SESSION_RADAR_FCC_LIMIT_REACHED`.

3) Default logging speed changed from 115200 Mbps to 3 Mbps

- SR040

1) Default logging speed changed from 115200 Mbps to 3 Mbps for all Host platform.

2) Default logging speed set to 1150200 Mbps for Nordic platfrom.

### 2.3.5 Features Removed

### 2.3.6 Features Updated

### 2.3.7 Build system changes

### 2.3.8 Demo updates

## 2.4 Release v04.04.03 (SR150)

### 2.4.1 Firmware

- IOT_12 RC2 FW integrated, with FW version v44.00.02

### 2.4.2 Scope of Build

1) PSDU Data rate 850kbit support for SR150.
2) Hybrid Scheduling support added
3) Get device capability info as per latest FIRA Specification
4) Removed Advertisement mode Observer support
5) New status for negative distance *UWBAPI_STATUS_OK_NEGATIVE_DISTANCE_REPORT*
6) OTP read write for calibration value of *OTP_PDOA_MFG_ZERO_OFFSET_CALIB* and *OTP_AOA_ANT_MULTIPOINT_CALIB*.
7) Reason codes are updated for *SESSION_STATUS_NTF* notification.
8) Session type *UWBD_RANGING_WITH_INBAND_DATA_TRANSFER* support
9) New Command response support added for *SESSION_QUERY_DATA_SIZE_IN_RANGING*
10) Default logging speed changed from 115200 Mbps to 3 Mbps for all Host platform.
11) Default logging speed set to 1150200 Mbps for Nordic platfrom.
12) *DATA_TRANSFER_MODE* renamed to *LINK_LAYER_MODE*
13) enum *dataTransferModes* renamed to *linkLayerModes*
14) *Data_Transfer_Mode_Raw* renamed to *Link_Layer_Mode_Bypass*

### 2.4.3 API Changes

New API added:

- `UwbApi_SetHusSession` for configuring Hybrid Session.
- `UwbApi_SessionQueryDataSize` to get the Max Application Data size in a single Ranging Round.
- `UwbApi_SuspendDevice`

API Signature changed:

- Signature of the API *UwbApi_SetStaticSts* is changed.

  The parameter `staticStsIv` is now `const *const`.

### 2.4.4 Folder Restructuring

### 2.4.5 New Features Added

- **SR1XX:**

  1) **Added support for new session type `UWBD_RANGING_WITH_INBAND_DATA_TRANSFER`, and updated the**
     demo data transfer tx and rx use the same session type.
  2) Added new status for negative distance UWBAPI_STATUS_OK_NEGATIVE_DISTANCE_REPORT.
  3) **Updated get device capability type `phUwbCapInfo_t`, for new and updated controlee device capability.**
     IDs changed for all the below mentioned configs. All other IDs also adjusted: `MAX_MESSAGE_SIZE`, `MAX_DATA_PACKET_PAYLOAD_SIZE`. Extra paratmers added: `SUSPEND_RANGING` `SESSION_KEY_LENGTH` Length changed for the below parameters: `DEVICE_ROLES` `RANGING_METHOD`
  4) Default logging speed changed from 115200 Mbps to 3 Mbps.

- SR150:

  1) Added support for OTP read write for calibration value of *OTP_PDOA_MFG_ZERO_OFFSET_CALIB* and *OTP_AOA_ANT_MULTIPOINT_CALIB*.
  2) As per new UCI Generic Specification reason code are updated for ` SESSION_STATUS_NTF` notification.
  3) Application Configuration parameter 'RANGING_INTERVAL' is renamed to 'RANGING_DURATION'.

4) PSDU Data rate 850kbit support for SR150.
5) Added support for Hybrid Scheduled Ranging and added Hybrid Demo ranging controller and controlee
6) Default logging speed changed from 115200 Mbps to 3 Mbps for all Host platform.
7) Default logging speed set to 1150200 Mbps for Nordic platfrom.

- SR040
  1) Default logging speed changed from 115200 Mbps to 3 Mbps for all Host platform.
  2) Default logging speed set to 1150200 Mbps for Nordic platfrom.

### 2.4.6 Features Removed

- **SR150:**
  1) Advertisement Observer mode support is removed from SR150 & SR160 package.

### 2.4.7 Features Updated

### 2.4.8 Build system changes

- **SR150:**
  1) Advertisement Observer mode support is removed from SR150 & SR160 builds.

### 2.4.9 Demo updates

- *demo_UWB_ble_sr1xx* & *demo_UWB_ble_sr1xxi* demos are merged & updated in *demo_nearby_interaction* demo.

## 2.5 Release `v04.03.15` (SR150)

### 2.5.1 Scope of Build

### 2.5.2 API Changes

### 2.5.3 Folder Restructuring

### 2.5.4 New Features Added

- **SR1XX:**
  1) Notification handling added for `UWBD_WIFI_COEX_IND_NTF` and `UWB_WIFI_COEX_MAX_ACTIVE_GRANT_DUARTION_EXCEEDED_WAR_NTF` .

### 2.5.5 Features Removed

### 2.5.6 Features Updated

- OWR AOA Advertiser Observer Demos added.
- CCC Controlee Demo added.
- Antenna Pair Info, Number of PDoA Measurements and PDoA Measurements have been added in the CCC Range Data Notification.

### 2.5.7 API Changes

Structures Changed:

1) the Structure `phUwbApiContext_t` in `UwbApi_Internal.h` at location `libs\uwb-iot\uwb_api\Api\UwbApi_Internal.h`, is updated to include CCC Range Data Ntf.
2) the Structure `phCccRangingData_t` in `UwbApi_Types.h` at location `libs\uwb-iot\uwb_api\Api\UwbApi_Types.h`, is added to include CCC Range Data Ntf.
3) the structure `phUwbCapInfo_t` in `UwbApi_Types_Proprietary.h` at location `libs\uwb-iot\uwb_api\Api\SR1XX\UwbApi_Types_Proprietary.h`, is updated to include CCC parameters.
4) the structure `phUwbDevInfo_t` is updated to include *UCI_CCC_Version* and *CCC_Version*.
5) the Structure :cpp:type:`phCccRangingData_t`has been updated for antenna pair info, number of PDoA Measurements and PDoA Measurements.

Enum Updated:

1) the enum `eSessionType` in `UwbApi_Types.h` at location `libs\uwb-iot\uwb_api\Api\UwbApi_Types.h`, is updated to include CCC Session Type.
2) the enum `eAppConfig` in `UwbApi_Types_Proprietary.h` at location `libs\uwb-iot\uwb_api\Api\SR1XX\UwbApi_Types_Proprietary.h`, is updated to include CCC app configurations.

### 2.5.8 Build system changes

- UWB feature macro for CCC *UWBFTR_CCC* added.

## 2.6 Release v04.02.01 (SR150)

### 2.6.1 Scope of Build

- IOT 11 Release
- GeoFencing Support
- Provisioned STS Support
- Wifi Coex Support
- DLTDoA 2.0 Spec Updates
- QN9090 BLE Low Power issue
- FIRA Lite Applet update to v1.0.14 Run6
- Advertisement Mode Support
- Linux VCOM interface Support for MCTT/PCTT

### 2.6.2 API Changes

- Removed UwbApi_SetRxAntennaeDelayCalib from libs.
- In *UwbApi_SetCalibration()* *length* parameter type is changed from *uint8_t* to *uint16_t*.
- `phUwbappContext_t`. Type of `phUwbappContext_t::phUwbFWImageContext_t` added.
- *SetFirmwareImage* Api is removed from UwbAdaption.c.
- Writer thread Removed along with its TML Context.

- `UwbApi_SetDebugParams()` *uint8_t noOfparams* parameter added to set multiple parameters.
- `UwbApi_SetDebugParams()` *pDebugParams* parameter type is changed from *phDebugParams_t* to *UWB_DebugParams_List_t*.
- `UwbApi_GetDebugParams()` *uint8_t noOfparams* parameter added to get multiple parameters.
- `UwbApi_GetDebugParams()` *pDebugParams* parameter type is changed from *phDebugParams_t* to *UWB_DebugParams_List_t*.
- `UwbApi_ConfigureShareableData()` parameters added for SR150 *noOfAppParams*, *AppParams_List*,`noOfDebugParams`,`DebugParams_List`.
- `Update_Nearby()` API added to Update the GATT server with the Accessory configuration data.
- `Erase_Nearby()` API added to Erase Service characteristic in GATT.
- `serializeDataFromDebugParams()` *pDebugParams* parameter type is changed from *phDebugParams_t* to *UWB_DebugParams_List_t*.
- `serializeDataFromDebugParams()` *uint8_t noOfParams* parameter added for serializing parameters.
- `printDebugParams()` *pDebugParams* parameter type is changed from *phDebugParams_t* to *UWB_DebugParams_List_t*.
- `printDebugParams()` *uint8_t noOfParams* parameter added for printing parameters.
- `parseDebugParams()` *pDebugParams* parameter type is changed from *phDebugParams_t* to *UWB_DebugParams_List_t*.
- `parseDebugParams()` *uint8_t noOfParams* parameter added for parsing parameters.

1) `phRangingMesrDlTdoa_t` added new parameter *rssi* to measure rssi by the DT-Tag upon reception of DTM.
2) `phUwbRRRDMList_t` added new parameter *responderSlots* to configure responder slots.
3) `UwbApi_ConfigureShareableData()` parameters added for SR040 *noOfAppParams*, *AppParams_List*.

- Removed phOsalUwb_ConsumeSemaphore from libs since not used.
- Removed phOsalUwb_Timer_Cleanup from libs since not used.
- Removed phOsalUwb_IsTimersRunning from libs since not used.
- Removed phHbci_GetInfo from libs since not used.
- Removed phHbci_GetGeneralInfo from libs since not used.
- Removed phHbci_QueryInfo from libs since not used.

- Removed phOsalUwb_CreateRecursiveMutex from libs since not used.
- Removed phOsalUwb_LockRecursiveMutex from libs since not used.
- Removed phOsalUwb_UnlockRecursiveMutex from libs since not used.

### 2.6.3 Folder Restructuring

- demos are rearranged as per device support.

### 2.6.4 New Features Added

- Nordic SR150 secure ranging support.
- AoA Azimuth and Elevation PROXIMITY NTF is updated in `RNG_DATA_NTF`
- New Application config added `RANGE_DATA_NTF_BOUND_AOA`
- New One Wire WiFi-CoEx feature added.
- New BLE Demo with pairing feature added.
- **Following UCI Extention Parameters added for Set/Get Debug API:**
  - *UCI_EXT_PARAM_ID_TEST_CONTENTION_RANGING_FEATURE*
  - *UCI_EXT_PARAM_ID_RANGING_TIMESTAMP_NTF*
  - *UCI_EXT_PARAM_ID_THREAD_SECURE_ELEMENT*
  - *UCI_EXT_PARAM_ID_THREAD_UWB_WLAN_COEX*
- **App configurations added:**
  - *RSSI_REPORTING*
  - *DL_TDOA_BLOCK_STRIDING*
- Structures Changed: #) `phUwbRadarNtf_t` is added for receive Notfication handling in Radar. #) `UWB_DebugParams_List_t` is updated to include *UWB_DebugParams_type_t*. #) `AccessoryUwbConfigDataContent_t` new config added to set CLock frequency drift value. #) `phRangingMesrDlTdoa` new DLTDoA ranging measurement parameters added.
- `service_nearby()` new BLE service added.

### 2.6.5 Features Removed

- *kUWB_StsConfig_StaticSts_Tdoa* is removed from `UWB_StsConfig_t`
- *phDebugParams_t* is removed

### 2.6.6 Features Updated

- *DL_TDOA_INTER_CLUSTER_SYNCH_PERIOD* is renamed to *DL_TDOA_HOP_COUNT*.

### 2.6.7 Build system changes

- *UWBFTR_WIFI_COEX* added.

### 2.6.8 Demo updates

- Added semslite demos to update FiraLite applet to `1.0.14 Run6`. See `sr150-demo_semslite_FiRaLite_A739_Run4_to_Run6` sr150-demo_semslite_FiRaLite_A739_Run4_to_Run6

  See `sr150-demo_semslite_FiRaLite_A739_Run5_to_Run6` sr150-demo_semslite_FiRaLite_A739_Run5_to_Run6

## 2.7 Release `v03.15.11` (SR150)

### 2.7.1 Scope of Build

- SR150: IOT 10 Release
- SR150: Data transfer feature support
- SR150: +/- 90 degree field of view support
- SR150: BLE demo Multiphone Support upto 5 connections
- SR150: QN9090 SDK 2.6.5 Integration

### 2.7.2 API Changes

New API added:

API Signature Changed:

1) `UwbApi_SetCalibration()` *length* parameter type is changed from *uint8_t* to *uint16_t*.

Structures Changed:

1) `UwbApi_SendData()` the structure `phUwbDataPkt_t` is updated to include *dst_endpoint* and *sequence_number*.
2) `phUwbRcvDataPkt_t` is added for receive data handling in data transfer.

API removed:

1) `UwbApi_StartDataSession()`
2) `UwbApi_SetRxAntennaeDelayCalib()`

### 2.7.3 Folder Restructuring

### 2.7.4 New Features Added

- Added support for UCI command chaining using pbf bit.
- Added support for Data Transfer feature as per FIRA CR311. It supports send and receive data to exchange application data.
- Added support for multiple phone connections using ble demo sr150-demo-UWB-ble-sr1xx.
- Added QN9090 SDK 2.6.5 Support
- Added support for RV4 Revision C board
- Added Support for runtime RV4 REV-B/REV-C board detection.
- New Apk is added for Multiple BLE connections .

### 2.7.5 Features Removed

- Removed support for Smart Home Data Transfer feature.

### 2.7.6 Features Updated

### 2.7.7 Build system changes

## 2.8 Release v03.15.03 (SR150)

### 2.8.1 Scope of Build

- QN9090 SDK REL_SDK_JN_QN_K32W_2.6.5_MR4 Updated

### 2.8.2 API Changes

- `phUwbappContext_t`. Type of `phUwbappContext_t::fwImage` change to `const uint8_t *`.
- Signature of `SetFirmwareImage()` changed. Type of `fwImgPtr` changed to `const uint8_t *`.

### 2.8.3 Folder Restructuring

- demos are rearranged as per device support.

### 2.8.4 New Features Added

- Added support for RV4 Revision C board
- Added Support for runtime RV4 REV-B/REV-C board detection.

### 2.8.5 Features Removed

### 2.8.6 Features Updated

### 2.8.7 Build system changes

## 2.9 Release v03.14.04 (SR150)

### 2.9.1 Scope of Build

- SR150: IoT_9 release

- SR150: Reverted 3D AoA Antenna inversion logic
- SR150: Feature based compilation support is added
- SR150: Sem wait is changed to Sem wait with timeout

### 2.9.2 API Changes

- For `UwbApi_GetCalibration()` in the response structure `phCalibRespStatus_t`

  an input rx antenna pair parameter *inRxAntenaaPair* is added for *AOA_ANTENNAE_PDOA_CALIB*
- Use standard GNU C type uint8_t, uint16_t and family. (UINT8, UINT16, UINT32 and such types are no longer used within the middleware)
- `#define STATIC` is now reanmed to `UWB_STATIC`

### 2.9.3 Folder Restructuring

- Added tools folder to package for FiRaLite provisioning.
- `UwbCore_Types.h` is removed and all the required definitions are moved to `phUwbTypes.h`

### 2.9.4 New Features Added

- Added support for pnp over socket for linux based platform
- Added support for calibration specific configuration:
  - `SNR_CALIB_CONSTANT_PER_PAIR`,
  - `TX_POWER_PER_ANTENNA`, and
  - `TX_TEMPERATURE_COMP_PER_ANTENNA`
- Added semslite demo to update FiRalite applet for Run4 sample.

### 2.9.5 Features Removed

- Calibration configs *TX_POWER*, *TX_TEMPERATURE_COMP* are removed.
- 360FoV demos are removed.
- Removed the semslite demos for old samples. List as below:-
- demo_semslite_FiRaLite_A693

- demo_semslite_FiRaLite_A739
- demo_semslite_IOT_A739
- demo_semslite_SUS_A693
- demo_semslite_SUS_A739
- demo_FiRaLite_ADF_Provision

### 2.9.6 Features Updated

- Updated calibration specific parameter *SNR_CALIB_CONSTANT_UNIFIED* to `SNR_CALIB_CONSTANT_PER_PAIR`,
- *AddDelayInMicroSec* definition is moved from libs to boards and renamed to *uwb_port_DelayinMicroSec()*
- UART Baud rate is changed to 3Mbps when verbose logging is enabled. Therefore, baudrate shall be set to 3Mbps for viewing logs in terminal emulator(ex. Tera term).

### 2.9.7 Build system changes

- Use standard GNU C type uint8_t, uint16_t and family.

## 2.10 Release `v03.13.03`

### 2.10.1 Scope of Build

- SR150: This is IoT_8 release for 360degree FoV feature support and Unified UCI changes for Antenna configurations
- Added support of reading CIR in Appcallback

### 2.10.2 API Changes

New API added:

1) *UwbApi_GetFwCrashLog()*
2) *UwbApi_SetDefaultCoreConfigs()*
3) `UwbApi_SetRxAntennaeDelayCalib()`
4) `UwbApi_ConfigureShareableData()`

5) `UwbApi_GetUwbConfigData()`
6) *`UwbApi_UpdateActiveRoundsReceiver()`*
7) *`UwbApi_UpdateActiveRoundsAnchor()`*
8) *`Se_API_RemoteGetData_WithoutTunnel()`*
9) *`Se_API_RemotePutData_WithoutTunnel()`*

API Signature Changed:

1) *`Se_API_RemotePutData()`*
2) *`Se_API_RemoteGetData()`*
3) *`Se_API_LocalPutData()`*

Changes in role of enum:

1) `phUwbSessionsContext_t::sessioncnt` is now an Input and Output Prameter.

### 2.10.3 TML Refactoring

TML refactoring is done, tml interface is made bus agnostic, and bus specific apis are added in respective board folder.

New APIs added:

1) *`uwb_uwbs_tml_init()`*
2) *`uwb_uwbs_tml_deinit()`*
3) *`uwb_uwbs_tml_setmode()`*
4) *`uwb_uwbs_tml_data_tx()`*
5) *`uwb_uwbs_tml_data_rx()`*
6) *`uwb_uwbs_tml_data_trx()`*
7) *`uwb_uwbs_tml_reset()`*
8) *`uwb_bus_init()`*
9) *`uwb_bus_deinit()`*
10) *`uwb_bus_reset()`*
11) *`uwb_bus_io_val_get()`*
12) *`uwb_bus_io_val_set()`*
13) *`uwb_bus_io_irq_wait()`*
14) *`uwb_bus_io_irq_en()`*

15) *uwb_bus_io_irq_dis()*
16) *uwb_bus_data_tx()*
17) *uwb_bus_data_rx()*
18) *uwb_bus_data_tx_no_assert()*
19) `uwb_bus_data_crc16_xmodem_init()`
20) `uwb_bus_data_crc16_xmodem()`

APIs Removed:

1) phNxpUwb_InitUWBS
2) phNxpUwb_DeInitUWBS
3) phNxpUwb_HbciTransceive
4) phNxpUwb_UciRead
5) phNxpUwb_UciWrite
6) phNxpUwb_SpiWrite
7) phNxpUwb_SpiRead
8) phNxpUwb_SpiInit
9) phNxpUwb_SpiDeInit
10) phNxpUwb_RtcSyncWrite
11) phNxpUwb_RtcSyncRead
12) phNxpUwb_HeliosIrqEnable
13) phNxpUwb_HeliosInteruptStatus
14) phNxpUwb_GpioIrqEnable
15) phNxpUwb_GpioIrqDisable

Types Added:

1) `UWB_AppParams_type_t`
2) `UWB_AppParams_value_au8_t`
3) `UWB_AppParams_value_t`
4) `UWB_AppParams_List_t`
5) `UWB_AppParams_type_t::kUWB_APPPARAMS_Type_Unknown`
6) `UWB_AppParams_type_t::kUWB_APPPARAMS_Type_u32`
7) `UWB_AppParams_type_t::kUWB_APPPARAMS_Type_au8`

New API added:

1) *UwbApi_GetAppConfigMultipleParams()*

Deprecated types:

1) *SetAppParams_type_t*
2) *SetAppParams_value_au8_t*
3) *SetAppParams_value_t*
4) *SetAppParams_List_t*
5) SetAppParams_type_t::kAPPPARAMS_Type_Unknown
6) SetAppParams_type_t::kAPPPARAMS_Type_u32
7) SetAppParams_type_t::kAPPPARAMS_Type_au8

### 2.10.4 Folder Restructuring

Re-structuring done for <TOP>/libs/uci-core/

All include files from <TOP>/libs/uci-core/uci/include are moved to <TOP>/libs/uci-core/inc/ All source files inside <TOP>/libs/uci-core/ are moved to <TOP>/libs/uci-core/src/

Moved various board specific transport and IO mapping files like phNxpUwb_DriverInterface.c, UWB_GpioIrq.c, etc. from folders in <TOP>/libs/halimpl/transport to relevant <TOP>/boards folders.

Renamed OSAL related files in <TOP>/libs/halimpl/osal based on FreeRTOS and POSIX OS:

```
phOsalUwb.c                 => phOsalUwb_FreeRTOS.c
phOsalUwb_linux.c           => phOsalUwb_posix.c
phOsalUwb_Queue.c           => phOsalUwb_Queue_FreeRTOS.c
phOsalUwb_Queue_linux.c     => phOsalUwb_Queue_posix.c
phOsalUwb_Thread.c          => phOsalUwb_Thread_posix.c
phOsalUwb_Timer.c           => phOsalUwb_Timer_FreeRTOS.c
phOsalUwb_Timer_linux.c     => phOsalUwb_Timer_posix.c
```

Added phOsalUwb_Thread_FreeRTOS.c OSAL file for FreeRTOS in <TOP>/libs/halimpl/osal

### 2.10.5 New Features Added

- Added support for Antennae defines Device configuration:
  - *ANTENNA_TX_IDX_DEFINE*,
  - *ANTENNA_RX_IDX_DEFINE* and
  - *ANTENNAE_RX_PAIR_DEFINE*
- Added support for setting antennae pair Calibration:
  - `AOA_ANTENNAE_PDOA_CALIB`,
  - `AOA_ANTENNAE_MULTIPOINT_CALIB`,
  - *RX_ANT_DELAY_CALIB*,
  - *PDOA_OFFSET_CALIB*,
  - *PDOA_MANUFACT_ZERO_OFFSET_CALIB* and
  - *AOA_THRESHOLD_PDOA*
- Added support for Session specific configuration:
  - *ANTENNAE_CONFIGURATION_TX*,
  - *ANTENNAE_CONFIGURATION_RX*
- Added support for calibration specific configuration:
  - *RSSI_CALIB_CONSTANT_HIGH_PWR*,
  - *RSSI_CALIB_CONSTANT_LOW_PWR*, and
  - `SNR_CALIB_CONSTANT_UNIFIED`

#### SR150 Features

- Bug fix for PNP stress test
- New api's added to set default core configs and get firmware crash log.
- Bug fix for BLOB command regression from FW side.
- Bug fix from MW for Generic error notification.
- Added 2 new API for DLTDOA feature support
- Added demos for DLTDOA
- Added BLE demo for RV4

### 2.10.6 Features Removed

- Application configs *RX_ANTENNA_SELECTION*, *TX_ANTENNA_SELECTION*, *RX_MODE and TOA_MODE* are removed, use unified antennae configuration from latest UCI spec.
- Demos with SUSClient related to Ranging and Semslite are NXP Internal.
- Wiring pi support is removed for Kernel driver
- Data transfer feature support is removed

### 2.10.7 Features Updated

- UCI Refactorization
- `phUwbappContext_t` is updated with new callbacks functions. Operating mode will be set as per the Callback functions.
- Demos *demo_ranging_controller* and *demo_ranging_controlee* are with static STS.
- Demos *demo-sc-initiator* and *demos-sc-responder* are with secure ranging and with SE are NXP Internal. NO non SE support.

### 2.10.8 Build system changes

- Removed all deprecated build configuration names of the Plug & Trust MW. use `PTMW_` as prefix for all CMake Defines.
- PNP demos, changed folder anmes to be consistant with Host names.

## 2.11 Release `v03.10.02` (SR150)

### 2.11.1 Scope of Release

- SR150: The Scope of release is for SR150 with SE051W and Linux SR150 MW

### 2.11.2 API Changes

New API added:

1) *UwbApi_UpdateActiveRoundsReceiver()*
2) *UwbApi_UpdateActiveRoundsAnchor()*

### 2.11.3 Folder Restructuring

Re-structuring done for `<TOP>/libs/uci-core/`

All include files from `<TOP>/libs/uci-core/uci/include` are moved to `<TOP>/libs/uci-core/inc/` All source files inside `<TOP>/libs/uci-core/` are moved to `<TOP>/libs/uci-core/src/`

Moved various board specific transport and IO mapping files like `phNxpUwb_DriverInterface.c`, `UWB_GpioIrq.c`, etc. from folders in `<TOP>/libs/halimpl/transport` to relevant `<TOP>/boards` folders.

Renamed OSAL related files in `<TOP>/libs/halimpl/osal` based on FreeRTOS and POSIX OS:

```
phOsalUwb.c                 => phOsalUwb_FreeRTOS.c
phOsalUwb_linux.c           => phOsalUwb_posix.c
phOsalUwb_Queue.c           => phOsalUwb_Queue_FreeRTOS.c
phOsalUwb_Queue_linux.c     => phOsalUwb_Queue_posix.c
phOsalUwb_Thread.c          => phOsalUwb_Thread_posix.c
phOsalUwb_Timer.c           => phOsalUwb_Timer_FreeRTOS.c
phOsalUwb_Timer_linux.c     => phOsalUwb_Timer_posix.c
```

Added `phOsalUwb_Thread_FreeRTOS.c` OSAL file for FreeRTOS in `<TOP>/libs/halimpl/osal`

### 2.11.4 New Features Added

- Kernel mode driver support added for RPI
- GPIO handling is moved to Kernel space

#### SR150 Features

- Added 2 new API for DLTDOA feature support
- Added demos for DLTDOA
- Added BLE demo for RV4

### 2.11.5 Features Removed

- Wiring pi support is removed for Kernel driver
- Data transfer feature support is removed

### 2.11.6 Features Updated

- UCI Refactorization
- *phUwbappContext_t* is updated with new callbacks functions. Operating mode will be set as per the Callback functions.

## 2.12 Release `v03.09.00` (SR150)

### 2.12.1 Scope of Release

- SR150: The Scope of release is for SR150 with SE051W

### 2.12.2 New Features Added

- RV4 support available
- MCTT mode added for RV4
- PnP mode support enabled for RV4

### 2.12.3 Build system changes

- Added MCUXpresso project for RV4: *RhodesV4_SE*

## 2.13 Release v03.08.01 (SR150)

### 2.13.1 Scope of Release

- SR150: The Scope of release is for SR150 with SE051W

### 2.13.2 API Changes

New API added:

1) *UwbApi_WriteOtpCmd()*
2) *UwbApi_ReadOtpCmd()*
3) *UwbApi_SetProfileParams()*
4) *UwbApi_GetTrng()*
5) *Se_API_GetFiraLiteVersion()*

### 2.13.3 Folder Restructuring

Limited the number of folders needed to include in IDE for compilation (-I). Few header files have been moved as shown below:

```
halimpl/hal                   => halimpl/inc
halimpl/log                   => halimpl/inc
halimpl/osal                  => halimpl/inc
halimpl/tml                   => halimpl/inc
halimpl/transport/SPI         => halimpl/inc
halimpl/utils                 => halimpl/inc

halimpl/transport/SPI/common => halimpl/transport/SPI

uci-core/uwa/include          => uci-core/uci/include
uci-core/uwb/include          => uci-core/uci/include

uwb-iot/uwb_api/PrintUtility => uwb-iot/uwb_api/Api
uwb-iot/uwb_api/Types         => uwb-iot/uwb_api/Api
```

(continues on next page)

(continued from previous page)

```
uwb-iot/uwb_api/Api/SE051_Wrapper => uwb-iot/uwb_api/Api
uwb-iot/uwb_api/Api/StateMachine  => uwb-iot/uwb_api/Api

uwb-iot/uwb_core/gki/common  => uwb-iot/uwb_core/include
uwb-iot/uwb_core/hal/include => uwb-iot/uwb_core/include
```

**Note:** Only few header files have been moved. Location of `.c` files remains as is.

### 2.13.4 New Features Added

#### SR150 Features

- Added New Api support for the TRNG
- Added support for proprietary device config "INITIAL_RX_ON_OFFSET_ABS" and "INITIAL_RX_ON_OFFSET_REL"

### 2.13.5 Features Updated

- Updated SE MW as per UGM spec v1.1
- UWB device specific configurations are moved to boards module

## 2.14 Release v03.06.01 (SR150)

### 2.14.1 Scope of Release

- The Scope of release is for SR150 with SE051W

### 2.14.2 API Changes

### 2.14.3 API Changes

New API added:

1) *UwbApi_GetTrng()*
2) *Se_API_GetFiraLiteVersion()*

### 2.14.4 Folder Restructuring

### 2.14.5 New Features Added

#### SR040 Features

- Added New Api support for the TRNG
- SHORT MAC ADDRESS and EXTENTED MAC ADDRESS WITH HEADER shall be generated using TRNG api in TDOA Tag demo
- Renamed `TRIM_BACKOFF_PREAMBLE_10` to `TRIM_POWER_CTRL_PREAMBLE_10`
- Renamed `TRIM_BACKOFF_PREAMBLE_27` to `TRIM_POWER_CTRL_PREAMBLE_27`
- New Trim Param `TRIM_PREAMBLE_10_27_PAYLOAD_PS_COEFFS_OVERRIDE` support is added

#### SR150 Features

- Added New Api support for the TRNG
- Added support for proprietary device config "INITIAL_RX_ON_OFFSET_ABS" and "INITIAL_RX_ON_OFFSET_REL"

### 2.14.6 Features Updated

- Updated SE MW as per UGM spec v1.1
- UWB device specific configurations are moved to boards module
- SE051W specific MCU project is removed, all SE demos are now part of "QN9090_MK_SHIELD_V4_SE" project

## 2.15 Release `v03.05.00` (SR150)

### 2.15.1 Scope of Release

- The Scope of release is for SR150 with SE051W

### 2.15.2 New Features Added

#### SR150 Features

- PDOA_OFFSET parameter is updated with PDOA_OFFSET_1 and PDOA_OFFSET_2 parameter
- Support for different data transfer modes
- Support for "UWBS_INBAND_DATA_BUFFER_BLOCK_SIZE" and "UWBS_INBAND_DATA_MAX_BLOCKS" is added in UwbApi_GetStackCapabilities
- All files are converted from cpp to c

### 2.15.3 Features Updated

- Semslite SUS version updated to 1.1
- Added standalone project for RhodesV4

## 2.16 Release `v03.04.00` (SR150)

### 2.16.1 Scope of Release

- The Scope of release is for SR150 with SE051W

### 2.16.2 API Changes

New API added:

1) *`Se_API_GetWrappedRDS()`*

API Updated:

1) `UwbApi_SetAppConfigWrappedRDS()`

### 2.16.3 Folder Restructuring

- Added Dedicated Folder for SR1XX firmware in pacakge
  - `Dev_Rhodes`: Developement variant for Rhodes3 board
  - `Prod_Rhodes`: Development variant for Rhodes3 board
  - `Prod_ROW`: Production variant for rest of the World

### 2.16.4 New Features Added

#### SR150 Features

- Added UCI 1.0 support
- Support of eSE + Non-eSE features based on D23
- Support of eSE + Non-eSE legacy features based on D24
- Added smart home data transfer demo
- Added support of installing FiraLite applet
- Added dedicated demo for ADF provisioning for FiRaLite applet
- Added E2E demo for Firalite OOB channel initialization over UART
- Added Bluetooth LE ™ demo for SR150

### 2.16.5 Features Updated

- Seperated semslite SUS and SUS Client demo for applet update
- I2C speed reduced to 400KHz for MK Shield

### 2.16.6 Build system changes

- Added python script `UWBIOT_APP_BUILD.py` for configuring demo build in MCU Xpresso standalone Project
- Added Demo build config file `UWBIOT_APP_BUILD.h` in MCU Xpresso standalone Project
- Added `UWBIOT_SR1XX_FW` macro for different FW support.
  - `Rhodes3_PROD`: Production variant for Rhodes3 board
  - `ROW_PROD`: Production variant for rest of the World
  - `Rhodes3_Dev`: Devlopment variant got Rhodes3 board

- `ROW_Dev`: Devlopment variant for rest of the World

## 2.17 Release v03.02.00 (SR150)

### 2.17.1 Scope of Release

- SR150 Data Transfer Feature including both Rx and Tx part
- SR150 Calibration Configurations OTP Read/write
- Single Rx Feature Support during normal ranging
- All legacy features including DSTWR Ranging
- TDoA Anchor and Tag Support

### 2.17.2 API Changes

New API added:

1) *UwbApi_Init_New()*
2) *UwbApi_SendData()*
3) `UwbApi_StartDataSession()`
4) `UwbApi_WriteOtpCalibDataCmd()`
5) `UwbApi_ReadOtpCalibDataCmd()`

#### SR150 Features

- Added App Config `INBAND_DATA_TX_BLOCK_SIZE` for smart Data transfer
- Added App Config `INBAND_DATA_RX_BLOCK_SIZE` for smart Data transfer
- Added App Config `RANGING_SUSPEND_MODE` for smart Data transfer
- **smart home data transfer use case:**
  - set default configuration as like data transfer
  - **set the below configurations for smart home datatransfer:**
    - `INBAND_DATA_TX_BLOCK_SIZE`
    - `INBAND_DATA_RX_BLOCK_SIZE`
    - `RANGING_SUSPEND_MODE`

- Initilize session with `UWBD_RANGING_SESSION`

- NOTE: Demo support for smart home datatransfer use case is yet to be added
- Added support for SR040 extended application configuration parameter LOG_PARAMS_CONF
- Demo support for Resonder-mode for SR040 is added.

### 2.17.3 New Features Added

- SR150 Data transfer feature upto 2031 bytess
- SR150 Calibration Configurations OTP Read/write

### 2.17.4 Build system changes

- Renamed UWBD from `QN9090_SR150_NON_SE` to `QN9090_SR150`

## 2.18 Release `v03.00.01` (SR150)

### 2.18.1 Scope of Release

- SR150 Data Transfer Feature including both Rx and Tx part
- SR150 Calibration Configurations OTP Read/write
- Single Rx Feature Support during normal ranging
- All legacy features including DSTWR Ranging
- TDoA Anchor and Tag Support

### 2.18.2 API Changes

New API added:

1) *`UwbApi_SendData()`*
2) `UwbApi_StartDataSession()`
3) `UwbApi_WriteOtpCalibDataCmd()`
4) `UwbApi_ReadOtpCalibDataCmd()`
5) `UwbApi_PerformBinding()`

6) `UwbApi_PerformLocking()`
7) `UwbApi_SetAppConfigWrappedRDS()`

### 2.18.3 New Features Added

- SR150 Data transfer feature upto 2031 bytess
- SR150 Calibration Configurations OTP Read/write
- SR150 Binding Process
- SR150 Locking Process
- Added API for Set App Config for wrapped RDS.
- SR150 Secure Ranging Process

## 2.19 Build `v01.04.00`

### 2.19.1 Scope of Release

- Added calibration applications

### 2.19.2 New Features Added

- Calibrataion routines
- SR040 FW Version `v00.05.05`

### 2.19.3 API Changes

Removed following API's Support.

- `UwbApi_SendRangingIntervalUpdateRequest`
- `UwbApi_GetSessionCount`
- `UwbApi_SendAppData`
- `UwbApi_RecvAppData`
- `UwbApi_GetRangingCount`

## 2.20 Release `v01.00.00`

### 2.20.1 Scope of Release

- This release enables board bring up and basic ranging functionality for SR040.

### 2.20.2 Validation platforms / counterparts

### 2.20.3 API Changes

- None

### 2.20.4 Folder Restructuring

- Major restructuring to allow access to multiple variants of UWBD, on multiple platforms from same code base.

### 2.20.5 New Features Added

- Added support for SR040
- Added SWUP for SR040

### 2.20.6 Features Removed

- NA

### 2.20.7 Features not available

- Only features applicable to SR040 are added in this document. Other features that are not ported to SR040 have been removed.

  e.g. CDC Framework build / PNP Application build for Rhodes V3 is not available as of now.

### 2.20.8 Features with limited testing

The SR040 FW Runs in CCC MAC Mode and not FiRA MAC The Test Object for SR100T is also configured to work in CCC Mode

Lauterbach debugger is needed to program the IC for the first time After that, a program ported on host controller can be used to update the SR040 FW over SPI interface.

CHAPTER

# THREE

# COMPILING & RUNNING

## 3.1 Pre-Requisites for compilation

This section describes the pre-requisites for compiling / building the middleware stack.

1) IDE: MCUXpresso.

   Please ensure MCUXpresso IDE is downloaded and installed from https://www.nxp.com/design/:MCUXpresso-IDE

2) MCUXpresso User SDKs

   In order to debug/download code for Finder V3, Rhodes V4, specific User SDK needs to be installed into the MCUXpresso IDE.

   Please install QN9090 SDK for Finder V3, Rhodes V4.

## 3.2 MCUXpresso IDE Projects

### 3.2.1 RhodesV4-SE Standalone MCUXpresso Project

The QN9090_SR150 SE and Non SE based application can be build and deploy using `RhodesV4_SE` MCUXpresso project .

One can use this project to build new UWBIoT based Application for RhodesV4-SR150 with SE051W platform.

### Demo Application

Below demo applications are part of this project

#### QN9090-SR150 supported Demos

1. *SR150 In Band Data Transfer Rx*
2. *SR150 IN Band Data Transfer Tx*
3. *Demo Ranging Controlee*
4. *Demo Ranging Controller*
5. *SR1XX OTP Storage Factory Mode*
6. *SR1XX OTP Storage Mainline Mode*
7. *TDOA Anchor UCI 2.0*
8. *TDOA Tag*
9. *Demo nearby interaction*
10. *Demo nearby interaction Client*
11. *SR1XX Demo Test TX*
12. *SR1XX Demo Test Rx*

#### QN9090-SR150-SE051W supported Demos

1. sr150-demo-semslite-FiRaLite-A739-Run4
2. SR150-demo-FiRaLite-ADF-Provision
3. *demo_binding*
4. *SR150 FiraLite Initiator*
5. *SR150 FiraLite Responder*

#### Prerequisites

- Requires MCUXpresso IDE v11.2.0 or later
- QN9090DK6 SDK Version 2.x should be installed in MCUXpresso IDE

#### How to Build Demo Application

For compiling any of above demo application one need to enable specific demo from `UWBIOT_APP_BUILD.h` file

**Note:** only one demo application can be build & deploy at a time.

## 3.3 SEGGER IDE Projects

### 3.3.1 NRF52840-SR1XX Standalone Segger Embedded Studio Project

The NRF52840-SR1XX and SE051W based application can be build and deploy using `Nrf52840_SR1XX.emProject` Segger Embedded Studio Project.

One can use this project to build new UWBIoT based Application for NRF52840-SR1XX platform.

#### Demo Application

Below demo applications are part of this project

#### NRF52840-SR1XX supported Demos

1. *SR150 In Band Data Transfer Rx*
2. *SR150 IN Band Data Transfer Tx*
3. *Demo Ranging Controlee*
4. *Demo Ranging Controller*
5. *SR1XX OTP Storage Factory Mode*
6. *SR1XX OTP Storage Mainline Mode*
7. *TDOA Anchor UCI 2.0*
8. *TDOA Tag*

#### NRF52840-SR150-SE051W supported Demos

1. sr150-demo-semslite-FiRaLite-A739-Run4
2. SR150-demo-FiRaLite-ADF-Provision
3. *demo_binding*
4. *SR150 FiraLite Initiator*
5. *SR150 FiraLite Responder*

#### Prerequisites

- Requires Segger Embedded Studio for ARM v6.30 or later

#### How to Build Demo Application

For compiling any of above demo application one need to enable specific demo from `UWBIOT_APP_BUILD.h` file

---

**Note:** only one demo application can be build & deploy at a time.

---

## 3.4 Raspberry PI + MK Shield CMake Project

The Raspberry PI + MK Shield Board based application can be build and deploy using cmake based build system. `AN13333_SR150_Linux_Setup.pdf` explains steps needed for this.

Please contact local FAE/CAS Support team for `AN13333_SR150_Linux_Setup.pdf`

## 3.5 Pre Compiled binaries

For different platforms, pre-compiled binaries are available for directly running a demo or for running other tools and scripts.

### 3.5.1 Precompiled Binaries for Rhodes V4 : SE Demos

These files are in `<TOP>/binaries/Rhodes4_SE` folder.

They contain binaries for Rhodes V4 board for Use Cases with Secure Element.

### 3.5.2 Precompiled Binaries for Rhodes V4

These files are in `<TOP>/binaries/Rhodes4` folder.

They contain binaries for Rhodes V4 board.

### 3.5.3 SEMS Lite Update header files

These files are in `<TOP>/binaries/SE051W` folder.

There are not pre-compiled binaires as such, but they contian binary code for the SEMS Lite based Update for the applets on the SE051W Secure Element.

CHAPTER

# FOUR

# SR150 DEMOS

## 4.1 Demo List SR150

### 4.1.1 Standalone C Examples for SR150

These examples are supported for: - Section 9.7.1 *RHODES IV*.

| UWB Demos Ranging | Description |
|---|---|
| Section 4.2 *Demo Ranging Controlee* | This demo showcases normal ranging with one device configured as a Controlee - Responder and another device configured as a Controller - Initiator [Another demo]. |
| Section 4.3 *Demo Ranging Controller* | This demo showcases normal ranging with one device configured as a Controller - Initiator and another device configured as a Controlee - Responder [Another demo]. |
| Section 4.21 *OWR AoA advertiser* | This demo showcases OWR Special use case App ranging with SR150 configured owr aoa advertiser and sends data packets to a owr aoa observer. |
| Section 4.22 *SR150 In Band Data Transfer Rx* | This demo showcases data transfer feature with one SR150 configured as a Controllee - Responder and another SR150 configured as a Controller - Initiator [Another demo]. |
| Section 4.23 *SR150 IN Band Data Transfer Tx* | This demo showcases data transfer feature with one SR150 configured as a Controller - Initiator and another SR150 configured as a Controlee - Responder [Another demo]. |
| Section 4.24 *SR1XX Demo Test TX* | This demo showcases how to test SR1XX in Test mode. This demo is tested with session ID `0x00000000` and used to check RF parameters. Here data is sent over RF (over the air). |
| Section 4.25 *SR1XX Demo Test Rx* | This demo showcases how to test SR1XX in Test mode. This demo is tested with session ID `0x00000000` and used to check RF parameters. Here data is received over RF (over the air). |
| Section 4.26 *Demo Hybrid Scheduled Ranging Controlee* | This demo showcases hybrid scheduled ranging with one device configured as a Controlee - Responder and another device configured as a Controller - Initiator [Another demo]. |
| Section 4.27 *Demo Hybrid Scheduled Ranging Controller* | This demo showcases hybrid scheduled ranging with one device configured as a Controller - Initiator and another device configured as a Controlee - Responder [Another demo]. |
| Section 4.28 *Demo CSA Controlee* | This demo showcases CSA scenario with one device (SR150) configured as a Controlee - Responder and another device (SR100) configured as a Controller - Initiator. |

| UWB Demos Ranging using FiraLite | Description |
|---|---|
| Section 4.4 *SR150 FiraLite Initiator* | This demo showcases secure ranging between two SR150-Se051W board, using FiraLite applet. This demo will configure the device as a Initiator-Controller.<br>In case of RTOS based platform with USB or UART, communication between initiator and responder is handled using serial communication via PC, and python application listening on UART com ports. And in case of embed linux hosts communication between initiator and responder is handled over IP, using socket, with initiator and responder devices as clients, and a python server running on a PC in the same network. |
| Section 4.5 *SR150 FiraLite Responder* | This demo showcases secure ranging between two SR150-Se051W board, using FiraLite applet. This demo will configure the device as Responder-Controlee.<br>In case of RTOS host platform with USB or UART, communication between initiator and responder is handled using serial communication via PC, and python application listening on UART com ports. And in case of embed linux hosts communication between initiator and responder is handled over IP, using socket, with initiator and responder devices as clients, and a python server running on a PC in the same network. |
| Section 4.6 *SR150 FiraLite Responder IOT Concurrency* | This demo showcases concurrency, it communicates with iot applet to get random no and also secure ranging with FiRaLite between two SR150-Se051W board, using FiraLite applet. This demo will configure the device as Responder-Controlee.<br>In case of RTOS host platform with USB or UART, communication between initiator and responder is handled using serial communication via PC, and python application listening on UART com ports. And in case of embed linux hosts communication between initiator and responder is handled over IP, using socket, with initiator and responder devices as clients, and a python server running on a PC in the same network. |

| UL TDoA Demos | Description |
|---|---|
| Section 4.9 *TDOA Anchor UCI 2.0* | This demo showcases TDOA Special use case App ranging with device configured as a Controller - Initiator and UWBS configured as a Controlee - Responder.<br>For details on Peer-to-Peer ranging sequence and configuration details for UWBS, refer to *Peer-to-Peer ranging*. |
| Section 4.10 *TDOA Tag* | This demo showcases TDOA Special use case App ranging with SR150 configured TDoA Tag and sends blink packets to a TDoA Anchor. |
| Section 4.11 *TDOA Sync Anchor UCI 2.0* | This demo showcases TDOA Special use case App ranging with device configured as a Controller - Initiator (UT-Synchronization Anchor) and UWBS configured as a Controlee - Responder.<br>For details on Peer-to-Peer ranging sequence and configuration details for UWBS, refer to *Peer-to-Peer ranging*. |

| Calibration / OTP Demos | Description |
|---|---|
| Section 4.7 *SR1XX OTP Storage Factory Mode* | This demo showcases OTP Write/Read operation for SR1XX in factory mode. |
| Section 4.8 *SR1XX OTP Storage Mainline Mode* | This demo showcases OTP Read from SR1XX in mainline mode. |

| Plug and Play Firmware | Description |
|---|---|
| Section 4.13 *RV4 Plug-n-Play FW* | This firmware is used as a PnP FW for RhodesV4 to run applications from PC. PC would send commands over UART to the PnP firmware, which would be forwarded to SR150 and the responses and notifications are returned to PC. |

| Plug and Play Firmware over Socket | Description |
|---|---|
| Section 4.14 *Plug-n-Play FW Over Socket* | This demo is used as a PnP socket server for embed-linux or other platform with socket, to run applications from PC. PC would send commands over socket to the PnP firmware, which would be forwarded to SR1xx/SR2xx and the responses and notifications are returned to PC. |

| DL TDoA Demos | Description |
| --- | --- |
| Section 4.15 *SR1XX DLTDOA Ranging Initiator* | This demo showcases DLTDOA ranging with two SR1XX configured as Initiator ( Initiator and Responder), and third SR1XX configured as a Mobile node(Receiver,Controlle) [Another demo]. |
| Section 4.16 *SR1XX DLTDOA Ranging Responder* | This demo showcases DLTDOA ranging with two SR1XX configured as a Anchor( Initiator and responder), and third SR1XX configured as a Mobile node(Receiver,Controlle) [Another demo]. |
| Section 4.17 *SR1XX DLTDOA Ranging Tag* | This demo showcases DLTDOA ranging with one SR1XX configured as a Tag node(Tag,Controlle), second SR1XX configured as Slave Anchor( Initiator and responder, Controlee) [Another demo] and third SR1XX configured as a Master anchor ( Initiator and Responder,Controller)[Another demo]. |

| MCTT/PCTT Demos | Description |
| --- | --- |
| Section 4.18 *MCTT & PCTT Demo (SR1XX)* | MCTT : MAC FIRA Conformance Test Tool PCTT : PHY FIRA Conformance Test Tool<br>This demo is used as a MCTT/PCTT FW QN9090(RV4) boards to run applications from PC. PC application has to send MCTT/PCTT compliant commands over USB/UART to the MCTT/PCTT firmware, which would be forwarded to SR1XX and the responses and notifications are returned to PC. |

| BLE Demos | Description |
| --- | --- |
| Section 4.19 *Demo nearby interaction* | This demo showcases ranging via Bluetooth LE ™ with background(pairing) feature with SR150 configured as a either Controller - Initiator or Controlee - Responder. |
| Section 4.20 *Demo nearby interaction Client* | This demo showcases ranging via Bluetooth LE ™ with background(pairing) feature with SR150 configured as a Controller - Initiator. This is the counter part of the : Section 4.19 *Demo nearby interaction* |

## 4.2 Demo Ranging Controlee

This demo showcases normal ranging with one device configured as a Controlee - Responder and another device configured as a Controller - Initiator [Another demo].

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Set the application ranging parameters
- Perform normal ranging with static STS.

### 4.2.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- For embed linux raspberry pi with crete setup build-rpi-crete
- Source: `demo_ranging_controlee`

### 4.2.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the demo_ranging_controlee.bin file.
- On linux platform run the built executable.
- Run the other demo Section 4.3 *Demo Ranging Controller* for normal ranging.

### 4.2.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                  :62000055 2B010000 01000000 00C80000 ..
↪ 308DD080 D0
TMLUWB  :TX > :SEND                  :22010004 01000000
TMLUWB  :RX < :RECV                  :60070001 0A
TMLUWB  :RX < :RECV                  :62000055 2C010000 01000000 00C80000 ..
↪ 3070D080 D0
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                  :22010004 01000000
TMLUWB  :RX < :RECV                  :42010001 00
```

(continues on next page)

(continued from previous page)

```
TMLUWB  :RX < :RECV                  :61020006 01000000 0300
TMLUWB  :TX > :SEND                  :21010004 01000000
TMLUWB  :RX < :RECV                  :60010001 01
TMLUWB  :RX < :RECV                  :41010001 00
TMLUWB  :RX < :RECV                  :61020006 01000000 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_
↪ranging_controlee/demo_ranging_controlee.c : Success!
```

If such a log is not seen, re-run the program.

## 4.3 Demo Ranging Controller

This demo showcases normal ranging with one device configured as a Controller - Initiator and another device configured as a Controlee - Responder [Another demo].

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Set the application ranging parameters
- Perform normal ranging with static STS.

### 4.3.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- For embed linux raspberry pi with crete setup *Raspberry PI + Crete CMake Project*
- Source: `demo_ranging_controller`

### 4.3.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the demo_ranging_controller.bin file.
- On linux platform run the built executable.
- Run the other demo Section 4.2 *Demo Ranging Controlee* for normal ranging.

### 4.3.3 TOF, 2D and 3D AoA Modes

This demo can also be used to be run for TOF single antenna, 2D Azimuth AoA, 2D Elevation AoA and 3D AoA.

```
#if UWBIOT_UWBD_SR100S
#define UWB_BOARD_RX_ANTENNA_CONFIG_MODE_VAL UWB_BOARD_RX_ANTENNA_CONFIG_
↪MODE_TOF
#else
// On Naked board, it's 2D AoA, Post packaging, it wil be set to 3D AoA
#define UWB_BOARD_RX_ANTENNA_CONFIG_MODE_VAL UWB_BOARD_RX_ANTENNA_CONFIG_
↪MODE_2DAOA
#endif
```

### 4.3.4 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                 :62000055 2E010000 01000000 ..␣
↪309BD080 D0
TMLUWB  :RX < :RECV                 :62000055 2F010000 01000000 ..␣
↪00000000 00
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                 :22010004 01000000
TMLUWB  :RX < :RECV                 :42010001 00
TMLUWB  :RX < :RECV                 :62000055 30010000 01000000 ..␣
↪00000000 00
TMLUWB  :RX < :RECV                 :61020006 01000000 0300
TMLUWB  :TX > :SEND                 :21010004 01000000
TMLUWB  :RX < :RECV                 :60010001 01
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 01000000 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_
↪ranging_controller/demo_ranging_controller.c : Success!
```

If such a log is not seen, re-run the program.

## 4.4 SR150 FiraLite Initiator

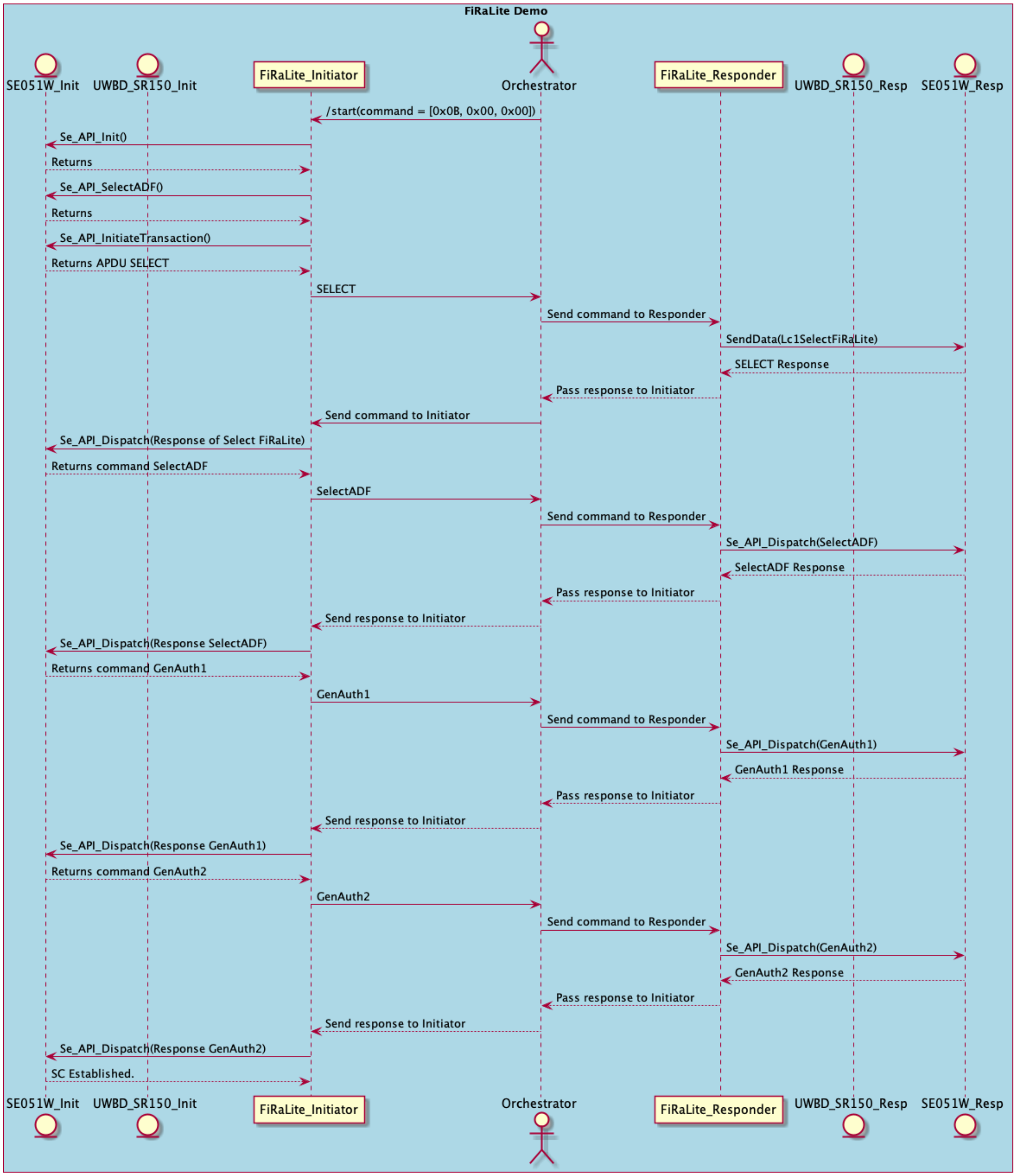
This demo showcases secure ranging between two SR150-Se051W board, using FiraLite applet. This demo will configure the device as a Initiator-Controller.

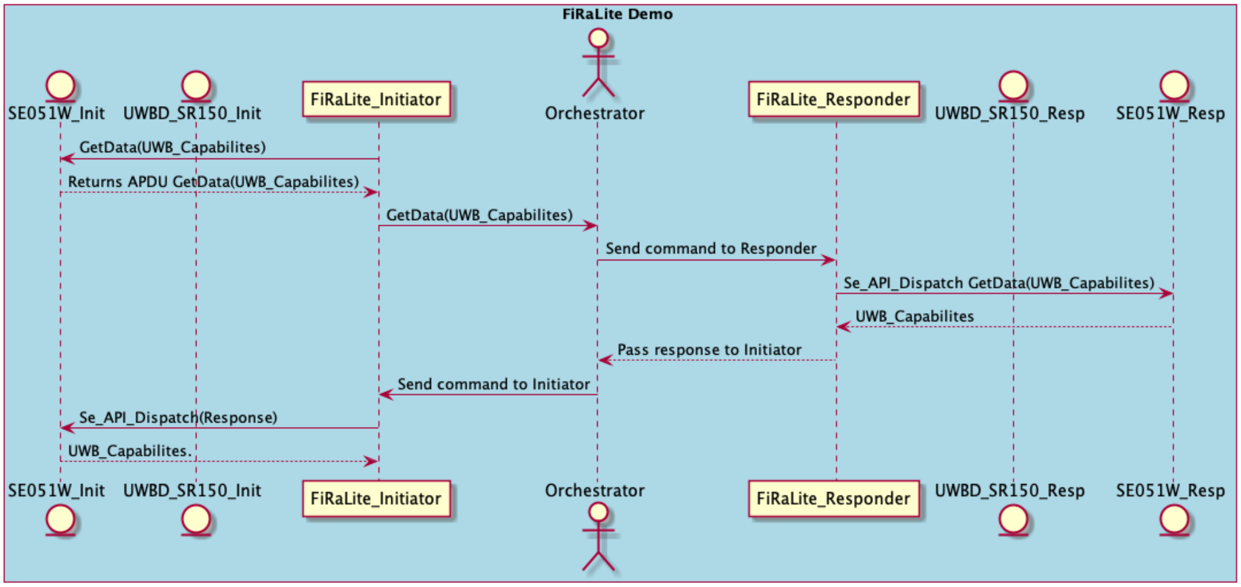
In case of RTOS based platform with USB or UART, communication between initiator and responder is handled using serial communication via PC, and python application listening on UART com ports. And in case of embed linux hosts communication between initiator and responder is handled over IP, using socket, with initiator and responder devices as clients, and a python server running on a PC in the same network.

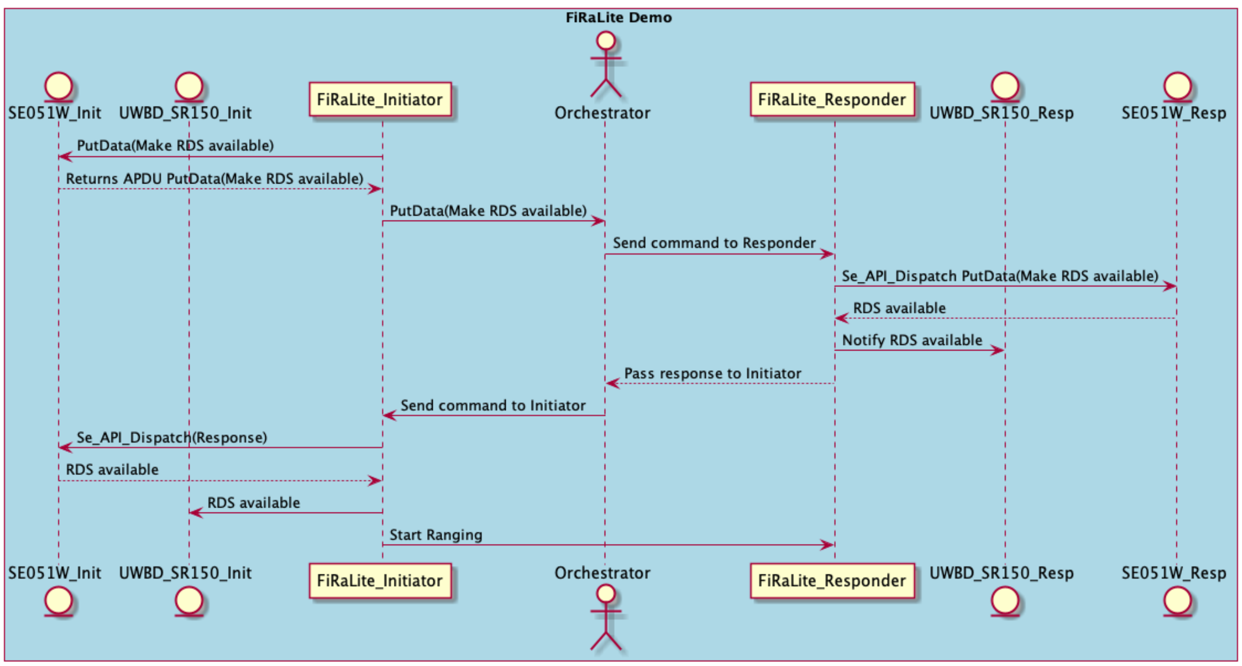
Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Authenticate with responder device using FiraLite applet
- Get the capabilities of the responder device
- Get the wrapped ranging data set using FiraLite applet
- Set the application ranging parameters
- Perform secure ranging with dynamic STS.

### 4.4.1 Secure Ranging on SR150-SE051W with FiRaLite Applet






- Following is the TLV coding followed for communication between initiator and responder.
- Tag is 1 byte coding Length is 2 bytes.

TAG

```
SE_SELECT_APPLET   0x78
SE_DISPATCH        0x79
SE_TUNNEL_GETDATA  0x7A
SE_TUNNEL_PUTDATA  0x7B
SE_START_RANGING   0x7C
```

- Once the Python Orchestrator starts TLV_TYPE_START 0x0B,0x00,0x00 is sent to initiator to start the channel initialization.
- The end of communication once wrapped RDS is available is sent by initiator to python orchestrator with TLV_START_RANGING `7C 00 00` and then from Orchestrator to responder, and Ranging procedure is started on both sides.

### 4.4.2 FiRaLite Applet Version

The demo is modified for FiRaLite(R4) 1.0.9 Release which available with On OEF A739 Run3 with FiRaLite updated to latest applet. Or on OEF A739 Run5 (already has latest FiRalite). There is no hybrid testing permitted. Mean both on the initiator and responsder should be on same latest FiRaLite versions. The demo has changes for Remote get data tag as per FiRa Consortium.

### 4.4.3 Prerequisites

- SE051W should be connceted to host Refer Section **??** *SE051W ARD Board*
- SR150 should be bound to SE051W. Refer Section 4.12 *demo_binding*
- Perform the ADF provisioning Refer *FiRaLite ADF Provisioning*
- Once ADF provisioning sucessful we can run `demo_fl_initator` binary.

### 4.4.4 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_fl_initiator`

### 4.4.5 Steps to be followed to run from RTOS-embedded device:

1) Flash `demo_fl_initiator.bin` file to the device
2) Flash `demo demo_fl_responder.bin` file to 2nd device
3) Get the comport of both initiator and responder device from device manager.
4) Connect to the com ports using any terminal application like putty/teraterm and reset both the device
5) On initiator port check the following log:

```
APP     :INFO :Initiator Starting OOB Session
```

6) On responder port check the following log:

```
APP     :INFO :responder Starting OOB Session
```

7) If these logs are not seen reset the device
8) Disconnect the com ports from terminal application
9) Run the python script in command prompt present at demos/SR150/fira_lite/ as:

```
python fira_lite_serial_transport.py <Initiator-COM-Port> <Responder-
↪COM-Port>
```

### 4.4.6 Steps to be followed to run from linux host like Rpi

1) Two linux host devices should be connected in same network.

2)**Run the python script from a pc connceted in same network present at demos/SR150/fira_lite/ as::**
python fira_lite_socket_transport.py

3) you will see the log with your ip address

```
server started 192.168.29.75 Waiting for client on port 8080
```

4) Build the demo demo_fl_initiator on one host linux device.
5) Set the environment variable UWBIOT_ENV_COM with the server ip address and run the demo:

```
export UWBIOT_ENV_COM=192.168.29.75:8080
./demo_fl_initiator
```

6) Build the demo demo_fl_responder on other host linux device, and follow the previous step.

### 4.4.7 Log (Success)

Once the authentication and storing wrapped RDS is done, python script will end with following log:

```
InitiatorRX:  79 00 25 0c db 3f ff 20 c3 68 41 89 86 42 87 29 9b fe f6 2e␣
↪6c 88 88 6a 6c eb fe 74 81 3e 5a 58 55 92 ce df 87 2b ce 40
```

(continues on next page)

(continued from previous page)

```
ResponderRX:  79 00 12 2a 2a ba c5 07 3e 2d 7f 9f 69 84 99 36 38 c7 23 90␣
↪00

InitiatorRX:  7c 00 00
Initialisation completed start ranging
FiraLite transport done
```

After this, reconnect to both the com ports using any terminal application like putty/teraterm, to see the ranging notifications, nothing needs to be done for linux host devices. Ranging is done for 5 mins and then session is closed.

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV        :6200003D 38010000 44332211 ... 00
TMLUWB  :RX < :RECV        :6200003D 39010000 44332211 ... 00
TMLUWB  :TX > :SEND        :22010004 44332211
TMLUWB  :RX < :RECV        :60070001 0A
TMLUWB  :TX > :SEND        :22010004 44332211
TMLUWB  :RX < :RECV        :42010001 00
TMLUWB  :RX < :RECV        :61020006 44332211 0300
TMLUWB  :TX > :SEND        :21010004 44332211
TMLUWB  :RX < :RECV        :60010001 01
TMLUWB  :RX < :RECV        :41010001 00
TMLUWB  :RX < :RECV        :61020006 44332211 0100
APP     :INFO :Finished D:/UWB_Iot_WorkArea/UWB_Iot_Top_SR150/uwbiot-top/
↪demos/SR150/demo_fl_initiator/demo_fl_initiator.c : Success
```

If such a log is not seen, re-run the steps .

## 4.5 SR150 FiraLite Responder

This demo showcases secure ranging between two SR150-Se051W board, using FiraLite applet. This demo will configure the device as Responder-Controlee.

In case of RTOS host platform with USB or UART, communication between initiator and responder is handled using serial communication via PC, and python application listening on UART com ports. And in case of embed linux hosts communication between initiator and responder is handled over IP, using socket, with initiator and responder devices as clients, and a python server running on a PC in the same network.

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- This will wait for the communication from initiator

- Authenticate with initiator device using FiraLite applet
- Get the wrapped ranging data set using FiraLite applet
- Set the application ranging parameters
- Perform secure ranging with dynamic STS.

### 4.5.1 Prerequisites

- SE051W should be connceted to host
- SR150 should be bound to SE051W. Refer Section 4.12 *demo_binding*
- Perform the ADF provisioning Refer Section 9.6 *FiRaLite ADF Provisioning*
- Once ADF provisioning successful we can run `demo_fl_responder` binary

### 4.5.2 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_fl_responder`

### 4.5.3 How to Run

Steps to be followed to run: Refer Section 4.4: *SR150 FiraLite Initiator*.

### 4.5.4 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV    :6200003D 1B010000 44332211 ... 00
TMLUWB  :RX < :RECV    :6200003D 1C010000 44332211 ... 00
TMLUWB  :RX < :RECV    :6200003D 1D010000 44332211 ... 00
TMLUWB  :RX < :RECV    :6200003D 1E010000 44332211 ... 00
TMLUWB  :TX > :SEND    :22010004 44332211
TMLUWB  :RX < :RECV    :60070001 0A
TMLUWB  :TX > :SEND    :22010004 44332211
TMLUWB  :RX < :RECV    :42010001 00
TMLUWB  :RX < :RECV    :61020006 44332211 0300
TMLUWB  :TX > :SEND    :21010004 44332211
```

(continues on next page)

(continued from previous page)

```
TMLUWB  :RX < :RECV    :60010001 01
TMLUWB  :RX < :RECV    :41010001 00
TMLUWB  :RX < :RECV    :61020006 44332211 0100
APP     :INFO :Finished D:/wa/WB_Iot_WorkArea/UWB_Iot_Top_SR150/uwbiot-
↪top/demos/SR150/demo_fl_responder/demo_fl_responder.c : Success!
```

If such a log is not seen, re-run the steps .

## 4.6 SR150 FiraLite Responder IOT Concurrency

This demo showcases concurrency, it communicates with iot applet to get random no and also secure ranging with FiRaLite between two SR150-Se051W board, using FiraLite applet. This demo will configure the device as Responder-Controlee.

In case of RTOS host platform with USB or UART, communication between initiator and responder is handled using serial communication via PC, and python application listening on UART com ports. And in case of embed linux hosts communication between initiator and responder is handled over IP, using socket, with initiator and responder devices as clients, and a python server running on a PC in the same network.

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- This will wait for the communication from initiator
- Authenticate with initiator device using FiraLite applet
- Communicates to IOT applet parallely to get random number.
- Get the wrapped ranging data set using FiraLite applet
- Set the application ranging parameters
- Perform secure ranging with dynamic STS.

### 4.6.1 Prerequisites

- SE051W should be connceted to host
- SR150 should be bound to SE051W. Refer Section 4.12 *demo_binding*
- Perform the ADF provisioning Refer Section 9.6 *FiRaLite ADF Provisioning*
- Once ADF provisioning successful we can run `demo_fl_responder_iot_concurrency` binary

### 4.6.2 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_fl_responder_iot_concurrency`

### 4.6.3 How to Run

Steps to be followed to run: Refer Section 4.4: *SR150 FiraLite Initiator*.

### 4.6.4 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV    :6200003D 1B010000 44332211 ... 00
TMLUWB  :RX < :RECV    :6200003D 1C010000 44332211 ... 00
TMLUWB  :RX < :RECV    :6200003D 1D010000 44332211 ... 00
TMLUWB  :RX < :RECV    :6200003D 1E010000 44332211 ... 00
TMLUWB  :TX > :SEND    :22010004 44332211
TMLUWB  :RX < :RECV    :60070001 0A
TMLUWB  :TX > :SEND    :22010004 44332211
TMLUWB  :RX < :RECV    :42010001 00
TMLUWB  :RX < :RECV    :61020006 44332211 0300
TMLUWB  :TX > :SEND    :21010004 44332211
TMLUWB  :RX < :RECV    :60010001 01
TMLUWB  :RX < :RECV    :41010001 00
TMLUWB  :RX < :RECV    :61020006 44332211 0100
APP     :INFO :Finished ../../demos/SR150/demo_fl_responder_iot_
↪concurrency/demo_fl_responder_iot_concurrency.c : Success!
```

If such a log is not seen, re-run the steps .

## 4.7 SR1XX OTP Storage Factory Mode

This demo showcases OTP Write/Read operation for SR1XX in factory mode.

Internally UWB is initiailized in Factory mode, and calibration data write and read is done in OTP. Following sequence of steps are handled.

- Initialize UWBD in Factory Firmware

- Write calib param in OTP (be careful while writing the data. Data written to OTP can not be reset)
- Read calib data from OTP

### 4.7.1 Prerequisites

- UWBS programmed with OTP implemented factory mode firmware

### 4.7.2 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*

### 4.7.3 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_otp_storage_factory.bin` file.
- On linux platform run the built executable.
- Source: `demo_otp_storage_factory`

### 4.7.4 Log (Success)

**Note:** This demo will success only on new device.

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                    :40040002 0000
TMLUWB  :TX > :SEND                    :20040004 01010101
TMLUWB  :RX < :RECV                    :40040002 0000
TMLUWB  :TX > :SEND                    :20040006 01E40402 F401
TMLUWB  :RX < :RECV                    :40040002 0000
TMLUWB  :TX > :SEND                    :20020000
TMLUWB  :RX < :RECV                    :4002003D 00011039 ... 00
TMLUWB  :TX > :SEND                    :2A010003 050100
TMLUWB  :RX < :RECV                    :4A010001 00
TMLUWB  :RX < :RECV                    :6A010004 00020100
```

(continues on next page)

(continued from previous page)

```
APP     :INFO :Read Calib data VCO_PLL: 0x0001
TMLUWB  :TX > :SEND                 :2A000006 05010300 0100
TMLUWB  :RX < :RECV                 :4A000001 00
TMLUWB  :RX < :RECV                 :6A000001 00
TMLUWB  :TX > :SEND                 :2A010003 050100
TMLUWB  :RX < :RECV                 :4A010001 00
TMLUWB  :RX < :RECV                 :6A010004 00020100
APP     :INFO :Read after Write VCO_PLL: 0x0001
APP     :INFO :Read and Write data are same
APP     :INFO :
Finished :<PROJECT_PATH>\uwbiot-top\demos\SR1XX\demo_otp_storage_factory\
↪demo_otp_storage_factory.c : Success!
```

If such a log is not seen, re-run the program.

## 4.8 SR1XX OTP Storage Mainline Mode

This demo showcases OTP Read from SR1XX in mainline mode.

Internally UWB is initiailized in Mainline mode, and calibration data is read from OTP. Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware
- Read calib param from OTP
- Set the calib data read from OTP
- Start Ranging

### 4.8.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_otp_storage_mainline`

### 4.8.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_otp_storage_mainline.bin` file.
- On linux platform run the built executable.

### 4.8.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :TX > :SEND                 :2A010003 050100
TMLUWB  :RX < :RECV                 :4A010001 00
TMLUWB  :RX < :RECV                 :6A010004 0002B081
TMLUWB  :TX > :SEND                 :2A010003 050108
TMLUWB  :RX < :RECV                 :4A010001 00
TMLUWB  :RX < :RECV                 :6A010004 00020000
TMLUWB  :TX > :SEND                 :2F210005 050002B0 81
TMLUWB  :RX < :RECV                 :4F210001 00
TMLUWB  :TX > :SEND                 :2F210005 05610200 00
TMLUWB  :RX < :RECV                 :4F210001 00
TMLUWB  :TX > :SEND                 :2F220002 0500
TMLUWB  :RX < :RECV                 :4F220006 00000002 B081
TMLUWB  :TX > :SEND                 :2F220002 0561
TMLUWB  :RX < :RECV                 :4F220006 00006102 0000
TMLUWB  :TX > :SEND                 :2A040000
TMLUWB  :RX < :RECV                 :4A040003 000102
APP     :INFO :Module Make ID is:
APP     :INFO :readMMId             :0102
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_otp_
↪storage_mainline/demo_otp_storage_mainline.c : Success!
```

If such a log is not seen, re-run the program.

## 4.9 TDOA Anchor UCI 2.0

This demo showcases TDOA Special use case App ranging with device configured as a Controller - Initiator and UWBS configured as a Controlee - Responder.

For details on Peer-to-Peer ranging sequence and configuration details for UWBS, refer to *Peer-to-Peer ranging*.

### 4.9.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_ultdoa_anchor`

### 4.9.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_ultdoa_anchor.bin` file.
- On linux platform run the built executable.
- Run the other demo *TDOA Tag* for normal ranging.

### 4.9.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                :6200003F 26010000 00000000 ..␣
↪00000001 012F2F
TMLUWB  :RX < :RECV                :6200003F 27010000 00000000 ..␣
↪00000001 013030
TMLUWB  :RX < :RECV                :6200003F 28010000 00000000 ..␣
↪00000001 013030
TMLUWB  :TX > :SEND                :22010004 00000000
TMLUWB  :RX < :RECV                :42010001 00
TMLUWB  :RX < :RECV                :61020006 00000000 0300
TMLUWB  :TX > :SEND                :21010004 00000000
TMLUWB  :RX < :RECV                :60010001 01
TMLUWB  :RX < :RECV                :41010001 00
TMLUWB  :RX < :RECV                :61020006 00000000 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_ultdoa_
↪anchor/demo_ultdoa_anchor.c : Success!
```

If such a log is not seen, re-run the program.

## 4.10 TDOA Tag

This demo showcases TDOA Special use case App ranging with SR150 configured TDoA Tag and sends blink packets to a TDoA Anchor.

### 4.10.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_ultdoa_tag`

### 4.10.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_ultdoa_tag.bin` file.
- On linux platform run the built executable.
- Run the other demo *TDOA Anchor UCI 2.0* for normal ranging.

### 4.10.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :TX > :SEND                 :22010004 00000000
TMLUWB  :RX < :RECV                 :42010001 00
TMLUWB  :RX < :RECV                 :61020006 00000000 0300
TMLUWB  :TX > :SEND                 :21010004 00000000
TMLUWB  :RX < :RECV                 :60010001 01
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 00000000 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_ultdoa_
↪tag/demo_ultdoa_tag.c : Success!
```

If such a log is not seen, re-run the program.

## 4.11 TDOA Sync Anchor UCI 2.0

This demo showcases TDOA Special use case App ranging with device configured as a Controller - Initiator (UT-Synchronization Anchor) and UWBS configured as a Controlee - Responder.

For details on Peer-to-Peer ranging sequence and configuration details for UWBS, refer to *Peer-to-Peer ranging*.

### 4.11.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_ultdoa_sync_anchor`

### 4.11.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_ultdoa_sync_anchor.bin` file.
- On linux platform run the built executable.
- Run the other demo *TDOA Tag* for normal ranging.

### 4.11.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                    :6200003F 92000000 00000000 00FFFFFF ..
↪ 012E2E
TMLUWB  :RX < :RECV                    :6200003F 93000000 00000000 00FFFFFF ..
↪ 012F2F
TMLUWB  :RX < :RECV                    :6200003F 94000000 00000000 00FFFFFF ..
↪ 012F2F
TMLUWB  :TX > :SEND                    :22010004 00000000
TMLUWB  :RX < :RECV                    :42010001 00
TMLUWB  :RX < :RECV                    :61020006 00000000 0300
TMLUWB  :TX > :SEND                    :21010004 00000000
TMLUWB  :RX < :RECV                    :60010001 01
TMLUWB  :RX < :RECV                    :41010001 00
TMLUWB  :RX < :RECV                    :61020006 00000000 0100
```

(continues on next page)

(continued from previous page)

```
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_ultdoa_
↪sync_anchor/demo_ultdoa_sync_anchor.c : Success!
```

If such a log is not seen, re-run the program.

## 4.12 demo_binding

This demo showcases SR150 Binding with SE051W in production process.

**Note:**

1. Binding process shall be done only once during the lifetime of the IC
2. Binding is an irresversible process. Once done, the IC state cannot be reverted to unbound state.

### 4.12.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_binding`

### 4.12.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_binding.bin` file.
- On linux platform run the built executable.

### 4.12.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :TX > :SEND                :29000000
TMLUWB  :RX < :RECV                :4900001B 008C0000 01020304 05060757␣
↪33365030 2D3934B1 31473447 001B00
TMLUWB  :TX > :SEND                :29010018 009B9B7F F5209D44 4790D321␣
↪47007474 6E6E6E62 62620000
TMLUWB  :RX < :RECV                :49010001 00
```

(continues on next page)

(continued from previous page)

```
TMLUWB  :TX > :SEND                 :29030000
TMLUWB  :RX < :RECV                 :49030010 000E8150 00000800 01020304␣
↪05060700
App     :INFO :pOutData             :00010203 04050607 0000FF03 607CD913␣
↪CC843A81 26348691 5F6C1263 68
TMLUWB  :TX > :SEND                 :29040011 107CD913 CC843A81 26348691␣
↪5F6C1263 68
TMLUWB  :RX < :RECV                 :49040018 00168582 33001046 5F12FFE8␣
↪28929FBC AEC21F9F C9425E00
App     :INFO :pOutData             :
TMLUWB  :TX > :SEND                 :29050001 30
TMLUWB  :RX < :RECV                 :49050010 000E8530 000008A9 81B6ED6C␣
↪33528E00
App     :INFO :pOutData             :AD18723E CCDEFFBF
TMLUWB  :TX > :SEND                 :2906000B 0AAD1872 3ECCDEFF BF9000
TMLUWB  :RX < :RECV                 :49060001 00
TMLUWB  :TX > :SEND                 :29020000
TMLUWB  :RX < :RECV                 :49020001 00
hostLib :INFO :rspBuf               :6F218410 A0000003 96545300 00000104␣
↪02000000 A50DBF0C 0A9F7E02 00074C03 000000
App     :INFO :Applet is Selected successfully!!!! .
App     :INFO :SUS Applet is Bound and in unlocked state
APP     :INFO :
Finished uwbiot-top\demos\SR150\demo_binding\demo_binding.c : Success!
```

If such a log is not seen, re-run the program.

## 4.13 RV4 Plug-n-Play FW

This firmware is used as a PnP FW for RhodesV4 to run applications from PC. PC would send commands over UART to the PnP firmware, which would be forwarded to SR150 and the responses and notifications are returned to PC.

### 4.13.1 How to Use

The PnP FW supports internal FW download and also from application, by default internal firmware download is disabled. to enable the internal firmware download, enable following macro.

```
/* Internal Firmware Download is DISABLED by Default.
```

(continues on next page)

(continued from previous page)

```
 *
 * Enable it if required by setting INTERNAL_FIRMWARE_DOWNLOAD to ENABLED
 */

#define INTERNAL_FIRMWARE_DOWNLOAD DISABLED
```

Flash the FW on to RhodesV4 board.

Pre-built firmware is present in `binaries/RhodesV4` directory.

On PC, set environment variable `UWBIOT_ENV_COM` to the RhodesV4 UART *COMPORT* and execute the application.

### 4.13.2 How to run PNP at higher baudrate

To run the pnp on higher baud rate than 3000000, low power mode has to be disabled on QN9090 platform which will increase the throughput

```
/*
 * For PNP Mode at higher baudrate than default one (3000000) on QN9090␣
↪(MKShield/Rhodes4)
 * low power mode has to be disabled, by setting following two macros
 * (cPWR_UsePowerDownMode and cPWR_FullPowerDownMode) to 0
 * The "clean + build re-compilation is needed for the changes to take␣
↪place in binary.
 */

/* Enable Power down modes
 * Need cPWR_UsePowerDownMode to be set to 1 first
 */

#define cPWR_UsePowerDownMode  1
#define cPWR_FullPowerDownMode 1
```

In file `demos/pnp/rhodesV4/pnp_usart.c`, set `DEMO_USART_BAUDRATE` to use higher baud rate.

### 4.13.3 How to run PNP with Hardware flow control

To run the pnp with Hardware flow control enabled

```
/*
 * Pnp support HW flow control support
 * To enable the HW flow control support set HW_FLOW_CONTROL_SUPPORT to 1
 * By default HW flow control is disabled
 * The "clean + build re-compilation is needed for the changes to take␣
↪place in binary.
*/

/* Rhodes 4 HW flow control support */
#define HW_FLOW_CONTROL_SUPPORT 0
```

### 4.13.4 Watch Dog Timer

See as well WATCHDOG_TIMEOUT_SECONDS.

```
/*
 * Reset the full board for recovery.
 *
 * If there's no communicaiton between the PnP FW
 * and host for these much time, the PnP FW would
 * auto reset.
 *
 * /!\ IMPORTANT /!\
 * /!\
 * /!\ A side effect of this Watchdog reset is that the Host has to
 * /!\ flush out and read all the previous packets from the RV4 board
 * /!\ on a new connection.
 * /!\
 * /!\ These packets are either buffered at FTDI, or due to the WatchDog
 * /!\ reset of QN9090, some garbage values are sensed/fetched by the
 * /!\ FTDI.
 * /!\
 * /!\ IMPORTANT /!\
 *
 * Set this to 0 to disable the Watch Dog Timer.
 *
 * Set this to non zero value to enable the watch dog timer behavior.
 *
 */
```

(continues on next page)

(continued from previous page)

```
#define WATCHDOG_TIMEOUT_SECONDS 0
```

## 4.14 Plug-n-Play FW Over Socket

This demo is used as a PnP socket server for embed-linux or other platform with socket, to run applications from PC. PC would send commands over socket to the PnP firmware, which would be forwarded to SR1xx/SR2xx and the responses and notifications are returned to PC.

### 4.14.1 How to Build

- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*

### 4.14.2 How to Use on linux host

This pnp demo supports internal FW download and also from application, by default internal firmware download is disabled. to enable the internal firmware download, enable following macro.

```
/* Internal Firmware Download is DISABLED by Default.
 *
 * Enable it if required by setting INTERNAL_FIRMWARE_DOWNLOAD to ENABLED
 */

#define INTERNAL_FIRMWARE_DOWNLOAD DISABLED
```

- Linux host device and your PC should be connected in same network.
- Run the demo on the linux host board.
- This demo is listening on port number 3001

## 4.15 SR1XX DLTDOA Ranging Initiator

This demo showcases DLTDOA ranging with two SR1XX configured as Initiator ( Initiator and Responder), and third SR1XX configured as a Mobile node(Receiver,Controlle) [Another demo].

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.

- Set the application ranging parameters
- Perform DLTDOA raning.

### 4.15.1 How to Build

- For RTOS based platform refer *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer build-embed-linux
- Source: `demo_dltdoa_initiator`

### 4.15.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the demo_dltdoa_initiator.bin file.
- On linux platform run the built executable.
- Run the other demos *SR1XX DLTDOA Ranging Tag* and *SR1XX DLTDOA Ranging Responder* for DLTDOA ranging.

### 4.15.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                   :62000048 6E170000 01000000 .. 92D080D0
TMLUWB  :RX < :RECV                   :62000048 6F170000 01000000 .. 94D080D0
TMLUWB  :TX > :SEND                   :22010004 01000000
TMLUWB  :RX < :RECV                   :42010001 00
TMLUWB  :RX < :RECV                   :61020006 01000000 0300
TMLUWB  :TX > :SEND                   :21010004 01000000
TMLUWB  :RX < :RECV                   :60010001 01
TMLUWB  :RX < :RECV                   :41010001 00
TMLUWB  :RX < :RECV                   :61020006 01000000 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_dltdoa_
↪initiator/demo_dltdoa_initiator.c : Success!
```

If such a log is not seen, re-run the program.

## 4.16 SR1XX DLTDOA Ranging Responder

This demo showcases DLTDOA ranging with two SR1XX configured as a Anchor( Initiator and responder), and third SR1XX configured as a Mobile node(Receiver,Controlle) [Another demo].

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Set the application ranging parameters
- Perform DLTDOA raning.

### 4.16.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_dltdoa_responder`

### 4.16.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_dltdoa_responder.bin` file.
- On linux platform run the built executable.
- Run the other demos *SR1XX DLTDOA Ranging Tag* and *SR1XX DLTDOA Ranging Initiator* for DLTDOA ranging.

### 4.16.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                :62000048 47170000 01000000 00C80000␣
↪00010000 00000000 00000000 .. 7FD080D0
TMLUWB  :RX < :RECV                :62000048 48170000 01000000 00C80000␣
↪00010000 00000000 00000000 .. 8ED080D0
TMLUWB  :RX < :RECV                :62000048 49170000 01000000 00C80000␣
↪00010000 00000000 00000000 .. 8CD080D0
TMLUWB  :RX < :RECV                :62000048 4A170000 01000000 00C80000␣
↪00010000 00000000 00000000 .. 9CD080D0
TMLUWB  :TX > :SEND                :22010004 01000000
```

(continues on next page)

(continued from previous page)

```
TMLUWB  :RX < :RECV                 :42010001 00
TMLUWB  :RX < :RECV                 :61020006 01000000 0300
TMLUWB  :TX > :SEND                 :21010004 01000000
TMLUWB  :RX < :RECV                 :60010001 01
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 01000000 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_dltdoa_
↪responder/demo_dltdoa_responder.c : Success!
```

If such a log is not seen, re-run the program.

## 4.17 SR1XX DLTDOA Ranging Tag

This demo showcases DLTDOA ranging with one SR1XX configured as a Tag node(Tag,Controlle), second SR1XX configured as Slave Anchor( Initiator and responder, Controlee) [Another demo] and third SR1XX configured as a Master anchor ( Initiator and Responder,Controller)[Another demo].

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Set the application ranging parameters
- Perform DLTDOA raning.

### 4.17.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_dltdoa_tag`

### 4.17.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_dltdoa_tag.bin` file.
- On linux platform run the built executable.
- Run the other demos *SR1XX DLTDOA Ranging Initiator* and *SR1XX DLTDOA Ranging Responder* for DLTDOA ranging.

### 4.17.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                 :6200001B 4A170000 01000000 00C80000␣
↪00020000 00000000 00000000 000000
TMLUWB  :RX < :RECV                 :6200001B 4B170000 01000000 00C80000␣
↪00020000 00000000 00000000 000000
TMLUWB  :RX < :RECV                 :6200001B 4C170000 01000000 00C80000␣
↪00020000 00000000 00000000 000000
TMLUWB  :TX > :SEND                 :22010004 01000000
TMLUWB  :RX < :RECV                 :42010001 00
TMLUWB  :RX < :RECV                 :61020006 01000000 0300
TMLUWB  :TX > :SEND                 :21010004 01000000
TMLUWB  :RX < :RECV                 :60010001 01
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 01000000 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_dltdoa_
↪tag/demo_dltdoa_tag.c : Success!
```

If such a log is not seen, re-run the program.

## 4.18 MCTT & PCTT Demo (SR1XX)

MCTT : MAC FIRA Conformance Test Tool PCTT : PHY FIRA Conformance Test Tool

This demo is used as a MCTT/PCTT FW QN9090(RV4) boards to run applications from PC. PC application has to send MCTT/PCTT compliant commands over USB/UART to the MCTT/PCTT firmware, which would be forwarded to SR1XX and the responses and notifications are returned to PC.

To run application in MCTT/PCTT mode, application has to perform `UwbApi_Init_New()` with mMcttCallback set and mCdcCallback and mAppCallback to `NULL` to activate MCTT/PCTT mode of operation. Once Init is success, Low Power Mode and NXP extended Notification configuration to be disabled if applicable. Tx Pulse shape config is set to 47 instead of default value 2. After this application can send MCTT/PCTT compliant commands to FW and response and notification are received.

### 4.18.1 How to Use

Flash the mctt_pctt firmware binary on the board. Use COMARCH tool to test MCTT feature. Use LitePoint tool to test PCTT feature

### 4.18.2 How to Build(Standalone project)

Build the project for SR1xx Using MCUXpresso IDE Project :

- Source: `demo_mctt_pctt`

### 4.18.3 How to Build(CMAKE project)

Build the mctt/pctt project for SR150 Using MCUXpresso Project

- Project: `cmake_qn9090_project`
- Source: `demo_mctt_pctt`

Build the mctt/pctt project for SR100 Using MCUXpresso Project

- Project: `cmake_rhodes3_project`
- Source: `demo_mctt_pctt`

### 4.18.4 How to Run

The project `demo_mctt_pctt` has package compiled for SR1xx located at `binaries/` in the project and can be run for `SR1XX` natively as:

```
demo_mctt_pctt.bin
```

Steps to be followed to run:

- Flash the `demo_mctt_pctt.bin` file to the SR1XX device
- Use required tool for required feature.

## 4.19 Demo nearby interaction

This demo showcases ranging via Bluetooth LE ™ with background(pairing) feature with SR150 configured as a either Controller - Initiator or Controlee - Responder.

For details on Peer-to-Peer ranging sequence and configuration details for SR150 refer to *Peer-to-Peer ranging*.

### 4.19.1 IOS

1) By default the UWB spec version is set to v1.1. To enable v1.0, we shall set the `UWB_IOS_SPEC_VERSION_MINOR` to `0x00, 0x00` in :file:UwbApi_types.h.
2) By default the Bonding and pairing capability is disabled, To enable them we shall set the below macros to 1

   `#define gAppUseBonding_d 1`

   `#define gAppUsePairing_d 1`

### 4.19.2 Android

1) By default the UWB spec version is set to v1.0 in :file:UwbApi_types.h.

- For multiple connection using ble, gAppMaxConnections_c define needs to updated to max number of connections allowed in `app_preinclude.h`, at location `boards/Host/Rhodes4/app_preinclude.h`, by default it's defined to 5. Clean build is needed after updating the `app_preinclude.h`

#### App Ranging with Bluetooth LE ™ on QN9090-SR150

In this demo, a smartphone first asks to pair the UWB device, once paired it sends commands over Bluetooth LE ™ to QN9090 to initialize and configure session and start ranging.

By Default the Low Power Mode is Enabled, To Disable the Low Power Mode we need to set the below macros as `0` inside `uwbiot-top/boards/Host/Rhodes4/app_preinclude.h`

`#define cPWR_UsePowerDownMode 0`

`#define cPWR_FullPowerDownMode 0`

For UWB and App developer specific changes refer `/*Define for App developer*/` & `/*Define for UWB developer*/` section in `TLV_Mng.c` file.

**Note:** `gatt_uuid128.h` file is moved from q9090 sdk to application as `gatt_uuid128_uwb_ble.h`, to update service UUID to match ios expection.

Similarly `gatt_db.h` file is moved to application as `gatt_db_uwb_ble.h` to update qpps rx characteristic.

---

### How to Build

Build the Bluetooth LE ™ based Ranging project for SR150 Using QN9090 MCUXpresso Project:

- Project: `RhodesV4_SE`
- Source: `demo_nearby_interaction`

Refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*.

### SR150 configuration

According to specification here are the configuration that can be changed:

- `TLV_Types_i.h` - DEVICE_ROLE:

```
#define DEMO_DEVICE_TYPE kUWB_DeviceType_Controller     // Configure␣
↪accessory as Controller
#define DEMO_DEVICE_ROLE kUWB_DeviceRole_Initiator      // Configure␣
↪accessory as Initiator
// #define DEMO_DEVICE_TYPE kUWB_DeviceType_Controlee   // Configure␣
↪accessory as Controlee
// #define DEMO_DEVICE_ROLE kUWB_DeviceRole_Responder   // Configure␣
↪accessory as Responder
```

- `TLV_Mng.c`:

```
uint8_t SpecMajorVersion[2] = {0x01, 0x00};                    // Spec␣
↪major version
uint8_t SpecMinorVersion[2] = {0x01, 0x00};                    // Spec␣
↪minor version
configData.PreferedUpdateRate                                  //␣
↪Prefered Update data rate (use PreferedUpdateRate_t definition)
```

### Log (Success)

After program execution, log message like this must be seen:

```
#################################################
## Demo Nearby Interaction (BLE tracker) : SR150
## UWBIOT_v04.05.05
#################################################
Device_MAC_ID = 00:60:37:0f:1f:cf
BLE Start adv
APP     :WARN :device init
|
|
TMLUWB  :RX < :RECV                :62000048 13000000 01000000 .. 89D080D0
TMLUWB  :RX < :RECV                :62000048 14000000 01000000 .. 9DD080D0
BLE Disconnected
TMLUWB  :TX > :SEND                :22010004 01000000
TMLUWB  :RX < :RECV                :60070001 0A
TMLUWB  :RX < :RECV                :62000048 15000000 01000000 .. 8AD080D0
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                :22010004 01000000
TMLUWB  :RX < :RECV                :42010001 00
TMLUWB  :RX < :RECV                :61020006 01000000 0300
TMLUWB  :TX > :SEND                :21010004 01000000
TMLUWB  :RX < :RECV                :60010001 01
TMLUWB  :RX < :RECV                :41010001 00
TMLUWB  :RX < :RECV                :61020006 01000000 0100
APP     :INFO :BLE Disconnected peer 1, 0 peers conenctd
APP     :WARN :device deinit
```

If such a log is not seen, re-run the program.

## 4.20 Demo nearby interaction Client

This demo showcases ranging via Bluetooth LE ™ with background(pairing) feature with SR150 configured as a Controller - Initiator. This is the counter part of the : Section 4.19 *Demo nearby interaction*

For details on Peer-to-Peer ranging sequence and configuration details for SR150 refer to *Peer-to-Peer ranging*.

### 4.20.1 Android

1) By default the UWB spec version is set to v1.0 in `UwbApi_types.h`.

### 4.20.2 IOS

1) Not Supported Yet.

#### App Ranging with Bluetooth LE ™ on QN9090-SR150

In this demo, first asks to pair the UWB device, once paired it sends commands over Bluetooth LE ™ to QN9090 (Section 4.19 *Demo nearby interaction*) to initialize and configure session and start ranging.

By Default the Low Power Mode is Enabled, to disable low power mode we need to set the below macros as `0` inside `uwbiot-top/boards/Host/Rhodes4/app_preinclude.h`

`#define cPWR_UsePowerDownMode 0`

`#define cPWR_FullPowerDownMode 0`

**Note:** `gatt_uuid128.h` file is moved from q9090 sdk to application as `gatt_uuid128_uwb_ble.h`, to update service UUID to match ios expection.

Similarly `gatt_db.h` file is moved to application as `gatt_db_uwb_ble.h` to update qpps rx characteristic.

#### How to Build

Build the Bluetooth LE ™ based Ranging project for SR150 Using QN9090 MCUXpresso Project:

- Project: `RhodesV4_SE`
- Source: `demo_nearby_interaction_client`

Refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*.

### Log (Success)

After program execution, log message like this must be seen:

```
################################################
## Demo BLE Tracker counterpart : SR150
## UWBIOT_v04.05.05
################################################
Device_MAC_ID = 00:60:37:29:bd:4c
Scanning...
Found device: Tag
Connected!
Scanning...
APP     :WARN :device init
FWDNLD  :INFO :FWDL Directly from host
HALUCI  :INFO :Starting FW download
HALUCI  :INFO :FW Download done.
TMLUWB  :RX < :RECV                :60010001 00
|
|
TMLUWB  :RX < :RECV                :62000048 11000000 01000000 00F00000 ..
↪ 98D080D0
APP     :INFO :TWR[0].distance        : 21
TMLUWB  :RX < :RECV                :62000048 12000000 01000000 00F00000 ..
↪ 51D040D0
APP     :INFO :TWR[0].distance        : 18
TMLUWB  :TX > :SEND                :22010004 01000000
TMLUWB  :RX < :RECV                :60070001 0A
TMLUWB  :RX < :RECV                :62000048 3C000000 01000000 00F00000 ..
↪ 00000000
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                :22010004 01000000
TMLUWB  :RX < :RECV                :42010001 00
TMLUWB  :RX < :RECV                :62000048 3D000000 01000000 00F00000 ..
↪ 00000000
TMLUWB  :RX < :RECV                :61020006 01000000 0300
APP     :WARN :device deinit
```

If such a log is not seen, re-run the program.

# 4.21 OWR AoA advertiser

This demo showcases OWR Special use case App ranging with SR150 configured owr aoa advertiser and sends data packets to a owr aoa observer.

## 4.21.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_owr_aoa_advertiser`

## 4.21.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_owr_aoa_advertiser.bin` file.
- On linux platform run the built executable.
- Run the counterpart demo `demo-owr-aoa-observer` demo-owr-aoa-observer for normal ranging.

## 4.21.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                 :62000038 4F3A0000 01000001 00C80000 ..
↪. 4AC12F2F
TMLUWB  :RX < :RECV                 :62000038 503A0000 01000001 00C80000 ..
↪. A6C02A2A
TMLUWB  :RX < :RECV                 :62000038 513A0000 01000001 00C80000 ..
↪. DCC03030
APP     :INFO :Data received successfully!!!
TMLUWB  :TX > :SEND                 :22010004 01000001
TMLUWB  :RX < :RECV                 :42010001 00
TMLUWB  :RX < :RECV                 :61020006 01000001 0300
TMLUWB  :TX > :SEND                 :21010004 01000001
TMLUWB  :RX < :RECV                 :60010001 01
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 01000001 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_owr_
↪aoa_observer/demo_owr_aoa_observer.c : Success!
```

If such a log is not seen, re-run the program.

## 4.22 SR150 In Band Data Transfer Rx

This demo showcases data transfer feature with one SR150 configured as a Controllee - Responder and another SR150 configured as a Controller - Initiator [Another demo].

Following sequence of steps are handled.

- Intialize UWBD in Mainline Firmware.
- Initialize the ranging session and Set the application configuration parameters
- Start the ranging session to move the device to ACTIVE state
- Invoke Send data API to receive the data and wait for Data Rx notification

Limitations:

- Maximum of 2031 bytes can be transferred.

### 4.22.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_inband_data_transfer_rx`

### 4.22.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_inband_data_transfer_rx.bin` file.
- On linux platform run the built executable.
- Run the other demo Section 4.23 *SR150 IN Band Data Transfer Tx* for normal ranging.

### 4.22.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                :02002501 01000001 00111100 00000000␣
↪00000014
|
|
APP     :INFO :Data Receive NTF Received
APP     :INFO :Data received successfully!!!
|
|
TMLUWB  :RX < :RECV                :62000048 42000000 01000001 .. 8ED080D0
TMLUWB  :RX < :RECV                :62000048 43000000 01000001 .. 5CD040D0
TMLUWB  :RX < :RECV                :61020006 01000001 0305
TMLUWB  :RX < :RECV                :60010001 01
TMLUWB  :TX > :SEND                :21010004 01000001
TMLUWB  :RX < :RECV                :60070001 0A
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                :21010004 01000001
TMLUWB  :RX < :RECV                :41010001 00
TMLUWB  :RX < :RECV                :61020006 01000001 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_inband_
↪data_transfer_rx/demo_inband_data_transfer_rx.c : Success!
```

If such a log is not seen, re-run the program.

## 4.23 SR150 IN Band Data Transfer Tx

This demo showcases data transfer feature with one SR150 configured as a Controller - Initiator and another SR150 configured as a Controlee - Responder [Another demo].

Following sequence of steps are handled.

- Intialize UWBD in Mainline Firmware.
- Initialize the ranging session and Set the application configuration parameters
- Start the ranging session to move the device to ACTIVE state
- Invoke Send data API to send the data and wait for Data Tx notification

Limitations:

- Maximum of 2031 bytes can be transferred.

## 4.23.1 Prerequisites

- SR100T (SR150) programmed configured as a Controlee - Responder

## 4.23.2 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- Source: `demo_inband_data_transfer_tx`

## 4.23.3 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_inband_data_transfer_tx.bin` file.
- On linux platform run the built executable.
- Run the other demo Section 4.22 *SR150 In Band Data Transfer Rx* for normal ranging.

## 4.23.4 Log (Success)

At the end of program execution, log message like this must be seen:

```
|
TMLUWB  :TX > :SEND                 :01002401 01000001 22220000 00000000␣
↪00001401 01020304 .. 0D0E0F10 11121314
|
TMLUWB  :RX < :RECV                 :62040005 01000001 00
TMLUWB  :RX < :RECV                 :62000055 37000000 01000001 00C80000 ..
↪ 00000000 00
TMLUWB  :RX < :RECV                 :62000055 38000000 01000001 00C80000 ..
↪ 00000000 00
TMLUWB  :TX > :SEND                 :21010004 01000001
TMLUWB  :RX < :RECV                 :60070001 0A
TMLUWB  :RX < :RECV                 :62000055 39000000 01000001 00C80000 ..
↪ 00000000 00
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                 :21010004 01000001
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :62000055 3A000000 01000001 00C80000 ..
↪ 00000000 00
```

(continues on next page)

(continued from previous page)

```
TMLUWB  :RX < :RECV                 :61020006 01000001 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_data_
↪transfer_tx/demo_tx.c : Success!
```

If such a log is not seen, re-run the program.

## 4.24 SR1XX Demo Test TX

This demo showcases how to test SR1XX in Test mode. This demo is tested with session ID `0x00000000` and used to check RF parameters. Here data is sent over RF (over the air).

Following sequence of steps are handled.

- Intialize UWBD.
- Initialize the test session and set the RF test parameters.
- Set test configuration.
- Start the RF test and check the parameters of RF test.

This test is generally used to measure characteristics like signal power, bandwith, etc.

### 4.24.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*

### 4.24.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_test_tx.bin` file.
- On linux platform run the built executable.
- Run the other demo `demos-test-rx` demos-test-rx for normal ranging.

---

**Note:** By default BPRF configuration is set. To enable HPRF configuration set `APP_INTERNAL_USE_HPRF` to 1.

---

### 4.24.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB   :TX > :SEND                 :21000005 00000000 D0
TMLUWB   :RX < :RECV                 :41000005 00010000 D0
TMLUWB   :RX < :RECV                 :61020006 010000D0 0000
TMLUWB   :TX > :SEND                 :2D000032 010000D0 09000410 27000001␣
↪04E80300 000204C2 01000003 04EE0200 00040100 05010006 04000000 00070400␣
↪00000008 0100
TMLUWB   :RX < :RECV                 :4D000002 0000
TMLUWB   :TX > :SEND                 :21030010 010000D0 03050101 06021111␣
↪07022222
TMLUWB   :RX < :RECV                 :41030002 0000
TMLUWB   :RX < :RECV                 :61020006 010000D0 0300
TMLUWB   :TX > :SEND                 :2103001A 010000D0 07040109 15010214␣
↪010A1201 00160100 1701011F 0100
TMLUWB   :RX < :RECV                 :41030002 0000
TMLUWB   :TX > :SEND                 :21030008 010000D0 01290100
TMLUWB   :RX < :RECV                 :41030002 0000
TMLUWB   :TX > :SEND                 :2D000009 010000D0 01E50301 01
TMLUWB   :RX < :RECV                 :4D000002 0000
TMLUWB   :TX > :SEND                 :2D02007F 01020304 05060708 090A0B0C␣
↪0D0E0F10 11121314 .. 7D7E7F
TMLUWB   :RX < :RECV                 :4D020001 00
APP      :INFO :Sleeping for 10s
TMLUWB   :RX < :RECV                 :60010001 02
TMLUWB   :RX < :RECV                 :61020006 010000D0 0200
TMLUWB   :RX < :RECV                 :6D020001 00
TMLUWB   :TX > :SEND                 :21010004 010000D0
APP      :INFO :pPerTxData->status  : 0
TMLUWB   :RX < :RECV                 :61020006 010000D0 0300
TMLUWB   :RX < :RECV                 :41010001 00
TMLUWB   :RX < :RECV                 :60010001 01
TMLUWB   :RX < :RECV                 :61020006 010000D0 0100
APP      :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_test_
↪tx/demo_test_tx.c : Success!
```

If such a log is not seen, re-run the program.

## 4.25 SR1XX Demo Test Rx

This demo showcases how to test SR1XX in Test mode. This demo is tested with session ID `0x00000000` and used to check RF parameters. Here data is received over RF (over the air).

Following sequence of steps are handled.

- Intialize UWBD.
- Initialize the test session and set the RF test parameters.
- Set test configuration.
- Start the RF test and check the parameters of RF test.

This test is generally used to measure characteristics like signal power, bandwith, etc.

### 4.25.1 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*

### 4.25.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the `demo_test_rx.bin` file.
- On linux platform run the built executable.
- Run the other demo `demos-test-tx` demos-test-tx for normal ranging.

**Note:** By default BPRF configuration is set. To enable HPRF configuration set `APP_INTERNAL_USE_HPRF` to 1.

### 4.25.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :TX > :SEND                :21000005 00000000 D0
TMLUWB  :RX < :RECV                :41000005 00010000 D0
TMLUWB  :RX < :RECV                :61020006 010000D0 0000
TMLUWB  :TX > :SEND                :2D000032 010000D0 09000428 00000001␣
```

(continues on next page)

(continued from previous page)

```
↪04A00F00 000204C2 01000003 04EE0200 00040100 05010006 04000000 00070400␣
↪00000008 0100
TMLUWB  :RX < :RECV                 :4D000002 0000
TMLUWB  :TX > :SEND                 :21030010 010000D0 03050101 06021111␣
↪07022222
TMLUWB  :RX < :RECV                 :41030002 0000
TMLUWB  :RX < :RECV                 :61020006 010000D0 0300
TMLUWB  :TX > :SEND                 :2103001A 010000D0 07040109 15010214␣
↪010A1201 00160100 1701011F 0100
TMLUWB  :RX < :RECV                 :41030002 0000
TMLUWB  :TX > :SEND                 :21030008 010000D0 01290100
TMLUWB  :RX < :RECV                 :41030002 0000
TMLUWB  :TX > :SEND                 :2D000009 010000D0 01E50301 01
TMLUWB  :RX < :RECV                 :4D000002 0000
TMLUWB  :TX > :SEND                 :2D030004 00000000
TMLUWB  :RX < :RECV                 :4D030001 00
TMLUWB  :RX < :RECV                 :60010001 02
TMLUWB  :RX < :RECV                 :61020006 010000D0 0200
TMLUWB  :RX < :RECV                 :6D030041 00280000 002C0000 00040000␣
↪00000000 00280000 00000000 00280000 00000000 00280000 00000000 00280000␣
↪00000000 00280000 000A0000 00010178 C8362812 00
TMLUWB  :RX < :RECV                 :61020006 010000D0 0300
TMLUWB  :RX < :RECV                 :60010001 01
TMLUWB  :TX > :SEND                 :21010004 010000D0
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                 :21010004 010000D0
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                 :21010004 010000D0
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                 :21010004 010000D0
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                 :21010004 010000D0
UWBAPI  :ERROR:sendUciCommandAndWait : event timedout
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_test_
↪rx/demo_test_rx.c : Success!
```

If such a log is not seen, re-run the program.

## 4.26 Demo Hybrid Scheduled Ranging Controlee

This demo showcases hybrid scheduled ranging with one device configured as a Controlee - Responder and another device configured as a Controller - Initiator [Another demo].

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Set the application ranging parameters
- Perform normal ranging with static STS.

### 4.26.1 How to Build

- Set `UWBFTR_DataTransfer` macro to 1
- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- Source: `demo_hybrid_ranging_controlee`

### 4.26.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the demo_hybrid_ranging_controlee.bin file.
- Run the other demo Section 4.27 *Demo Hybrid Scheduled Ranging Controller* for hybrid scheduled ranging.

### 4.26.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                         :62000048 00000000 02000003 00640000␣
↪00010000 .. 93D080D0
TMLUWB  :TX > :SEND                         :22010004 02000003
TMLUWB  :RX < :RECV                         :42010001 00
TMLUWB  :RX < :RECV                         :61020006 02000003 0300
TMLUWB  :TX > :SEND                         :22010004 03000003
TMLUWB  :RX < :RECV                         :42010001 00
TMLUWB  :RX < :RECV                         :61020006 03000003 0300
TMLUWB  :TX > :SEND                         :22010004 04000005
TMLUWB  :RX < :RECV                         :42010001 00
TMLUWB  :RX < :RECV                         :61020006 04000005 0300
```

(continues on next page)

(continued from previous page)

```
TMLUWB  :TX > :SEND                 :22010004 01000001
TMLUWB  :RX < :RECV                 :60070001 0A
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                 :22010004 01000001
TMLUWB  :RX < :RECV                 :42010001 00
TMLUWB  :RX < :RECV                 :61020006 01000001 0300
TMLUWB  :TX > :SEND                 :21010004 02000003
TMLUWB  :RX < :RECV                 :60010001 01
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 02000003 0100
TMLUWB  :TX > :SEND                 :21010004 03000003
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 03000003 0100
TMLUWB  :TX > :SEND                 :21010004 04000005
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 04000005 0100
TMLUWB  :TX > :SEND                 :21010004 01000001
TMLUWB  :RX < :RECV                 :41010001 00
TMLUWB  :RX < :RECV                 :61020006 01000001 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_hybrid_
↪ranging_controlee/demo_hybrid_ranging_controlee.c : Success!
```

If such a log is not seen, re-run the program.

## 4.27 Demo Hybrid Scheduled Ranging Controller

This demo showcases hybrid scheduled ranging with one device configured as a Controller - Initiator and another device configured as a Controlee - Responder [Another demo].

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Set the application ranging parameters
- Perform normal ranging with static STS.

### 4.27.1 How to Build

- Set `UWBFTR_DataTransfer` macro to 1
- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- Source: `demo_hybrid_ranging_controller`

### 4.27.2 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the demo_hybrid_ranging_controller.bin file.
- Run the other demo Section 4.26 *Demo Hybrid Scheduled Ranging Controlee* for hybrid scheduled ranging.

### 4.27.3 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                    :62000048 B90B0000 04000005 .. 00000000
TMLUWB  :RX < :RECV                    :62000048 BA0B0000 04000005 .. 00000000
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                    :22010004 04000005
TMLUWB  :RX < :RECV                    :42010001 00
TMLUWB  :RX < :RECV                    :61020006 04000005 0300
TMLUWB  :TX > :SEND                    :22010004 01000001
TMLUWB  :RX < :RECV                    :42010001 00
TMLUWB  :RX < :RECV                    :61020006 01000001 0300
TMLUWB  :TX > :SEND                    :21010004 02000003
TMLUWB  :RX < :RECV                    :60010001 01
TMLUWB  :RX < :RECV                    :41010001 00
TMLUWB  :RX < :RECV                    :61020006 02000003 0100
TMLUWB  :TX > :SEND                    :21010004 03000003
TMLUWB  :RX < :RECV                    :41010001 00
TMLUWB  :RX < :RECV                    :61020006 03000003 0100
TMLUWB  :TX > :SEND                    :21010004 04000005
TMLUWB  :RX < :RECV                    :41010001 00
TMLUWB  :RX < :RECV                    :61020006 04000005 0100
TMLUWB  :TX > :SEND                    :21010004 01000001
TMLUWB  :RX < :RECV                    :41010001 00
TMLUWB  :RX < :RECV                    :61020006 01000001 0100
```

(continues on next page)

(continued from previous page)

```
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_hybrid_
↪ranging_controller/demo_hybrid_ranging_controller.c : Success!
```

If such a log is not seen, re-run the program.

## 4.28 Demo CSA Controlee

This demo showcases CSA scenario with one device (SR150) configured as a Controlee - Responder and another device (SR100) configured as a Controller - Initiator.

Following sequence of steps are handled.

- Initialize UWBD in Mainline Firmware.
- Create Session
- Configure Session
- Start Session

### 4.28.1 How to Set Wrapped Rds(optional)

This section explain how to form and call the wrapped Rds. Uncomment below to enable this feature.

```
status = set_wrapred_rds(sessionHandle);
if (status != UWBAPI_STATUS_OK) {
    LOG_E("set_wrapred_rds Failed");
    goto exit;
}
```

```
/*
 * Example code to form the Wrapped Rds.
 */


tUWBAPI_STATUS set_wrapred_rds(uint32_t sessionHandle){

    tUWBAPI_STATUS status;
```

(continues on next page)

(continued from previous page)

```
    //  To Set Wrapped RDS which includes
    // Session ID(4 octets) || Random(12 octets) || Plain text Session␣
↪Key(32 octets)
    uint8_t wrappedRds[CCC_WRAPPED_RDS_LEN] = {0};
    // 12 bytes of Random Key with its 1st 4 bytes set as 0xB5 to␣
↪distinguish from normal WRAPPED_RDS command.
    uint8_t RandomKey[RANDOM_KEY_LEN] = {0xB5, 0xB5, 0xB5, 0xB5, 0x09,␣
↪0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10};
    // 32 bytes of Plain Text Session Key
    uint8_t SessionKey[CCC_SESSION_KEY_LEN] = {0x01, 0x02, 0x03, 0x04,␣
↪0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10,\
                                              0x01, 0x02, 0x03, 0x04,␣
↪0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10 \
                                             };

    // Form The Wrapped RDS
    serializeSessionHandlePayload(sessionHandle, &wrappedRds[0]);
    phOsalUwb_MemCopy(&wrappedRds[SESSION_ID_LEN], RandomKey, RANDOM_KEY_
↪LEN);
    phOsalUwb_MemCopy(&wrappedRds[SESSION_ID_LEN + RANDOM_KEY_LEN],␣
↪SessionKey, CCC_SESSION_KEY_LEN);

    status = UwbApi_SetAppConfigWrappedRDS(sessionHandle, wrappedRds, CCC_
↪WRAPPED_RDS_LEN);
    if (status != UWBAPI_STATUS_OK) {
        LOG_E("UwbApi_SetAppConfigWrappedRDS Failed");
        goto exit;
    }
exit :
    return status;

}
```

### 4.28.2 How to Build

- For RTOS based platform refer Section 3.2.1 *RhodesV4-SE Standalone MCUXpresso Project*
- For embed linux platform refer Section 3.4 *Raspberry PI + MK Shield CMake Project*
- For embed linux raspberry pi with crete setup build-rpi-crete
- Source: `demo_csa_controlee`

### 4.28.3 How to Run

Steps to be followed to run:

- For embedded RTOS device flash the demo_ranging_controlee.bin file.
- On linux platform run the built executable.

### 4.28.4 Log (Success)

At the end of program execution, log message like this must be seen:

```
TMLUWB  :RX < :RECV                :62200033 010000A0 00C34902 00000020 ..
↪ D2DFDF17 D39AE0
TMLUWB  :RX < :RECV                :62200033 010000A0 00F34902 00000049 ..
↪ D361E0D5 D19AE0
TMLUWB  :RX < :RECV                :62200033 010000A0 00234A02 0000001C ..
↪ D1A9E05D D2F1DF
TMLUWB  :TX > :SEND                :22010004 010000A0
TMLUWB  :RX < :RECV                :60070001 0A
TMLUWB  :RX < :RECV                :62200033 010000A0 00534A02 00000051 ..
↪ D0DCDF4D D258E0
TMLUWB  :RX < :RECV                :62200033 010000A0 F0534A02 ..␣
↪D0DCDF4D D258E0
UCICORE :WARN :Retrying last failed command
TMLUWB  :TX > :SEND                :22010004 010000A0
TMLUWB  :RX < :RECV                :42010001 00
TMLUWB  :RX < :RECV                :61020006 010000A0 0300
TMLUWB  :TX > :SEND                :21010004 010000A0
TMLUWB  :RX < :RECV                :60010001 01
TMLUWB  :RX < :RECV                :41010001 00
TMLUWB  :RX < :RECV                :61020006 010000A0 0100
APP     :INFO :Finished <Project_Path>/uwbiot-top/demos/SR1XX/demo_csa_
↪controlee/demo_csa_controlee.c : Success!
```

If such a log is not seen, re-run the program.

CHAPTER

FIVE

# SE051W DEMOS

## 5.1 Demo List SE051W

### 5.1.1 Standalone C Examples for SE051W

These examples are supported with SE051W With QN9090 RhodesV4 board.

| Applet Update Demos | Description |
|---|---|
| Section 5.2 *se_vcom* | This project allows the Rhodes4 boards to be used as a bridge between the PC and the secure module and enables the execution of the config tool and other utilities from the PC |

## 5.2 se_vcom

This project allows the Rhodes4 boards to be used as a bridge between the PC and the secure module and enables the execution of the config tool and other utilities from the PC

### 5.2.1 How to Build

Build the project for SR150 Using MCUXpresso IDE Project :

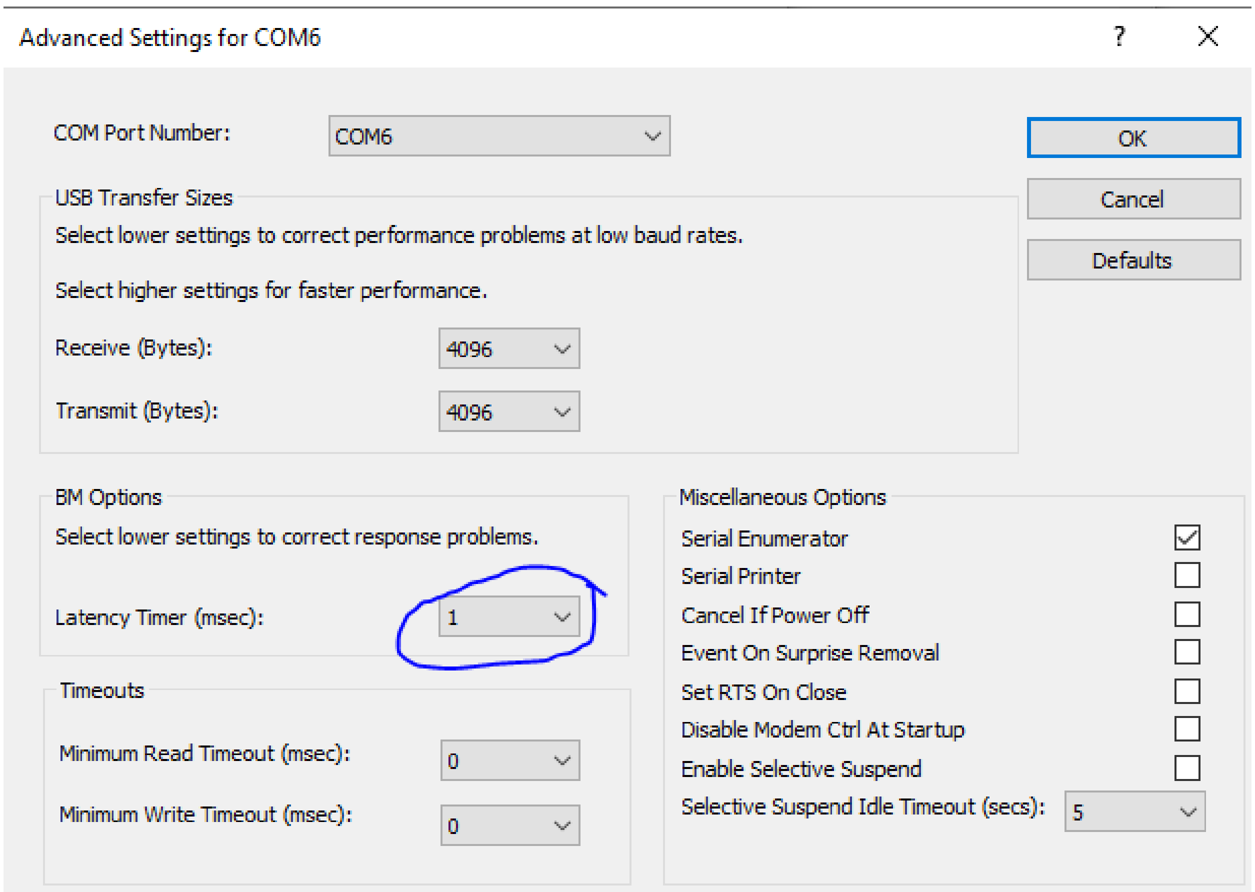
- Source: `se_vcom`

### 5.2.2 How to Run

The project `se_vcom` has package compiled for Rhodes4 board located at `binaries/Rhodes4_SE/` in the project and can be run for SE051W natively as:

```
se_vcom.bin
```

**Note:** In device manager change your COMPORT settings. By default latency timer(msec) is set as 16mesc update it to 1msec. `COM port properties` → `port settings` → `advanced` → `latency timer(msec)`



CHAPTER

SIX

# APIS : UWB IOT AND SR150

## 6.1 UWB API Status Codes

*group* **uwb_status**

Various Status cores returned by UWB APIs.

### Defines

**UWBAPI_STATUS_LOW_POWER_ERROR**

Status Codes for SR040.

Battery low error status

**UWBAPI_STATUS_OK**

Status Codes, as per UCI.

Command succeeded

**UWBAPI_STATUS_REJECTED**

Request is rejected

**UWBAPI_STATUS_FAILED**

Command Failed

**UWBAPI_STATUS_NOT_INITIALIZED**

API called without UCI being initialized

**UWBAPI_STATUS_INVALID_PARAM**

Invalid parameter provided

**UWBAPI_STATUS_INVALID_RANGE**
Invalid value range provided

**UWBAPI_STATUS_SESSION_NOT_EXIST**
Session wrt SESSION Handle Does not exist

**UWBAPI_STATUS_INVALID_PHASE_PARTICIPATION**
Invalid Phase Participation values in SESSION_SET_HUS_CONFIG_CMD

**UWBAPI_STATUS_SESSION_ACTIVE**
Session active

**UWBAPI_STATUS_MAX_SESSIONS_EXCEEDED**
MAX Sessions exceeded

**UWBAPI_STATUS_SESSION_NOT_CONFIGURED**
Operation is started with out configuring required parameters for Session

**UWBAPI_STATUS_SESSIONS_ONGOING**
Sessions ongoing

**UWBAPI_STATUS_MULTICAST_LIST_FULL**
Indicates when multicast list is full during one to many ranging

**UWBAPI_STATUS_OK_NEGATIVE_DISTANCE_REPORT**
Negative distance was reported

**UWBAPI_STATUS_ESE_RESET**
ESE Rest happened during command processing.

**UWBAPI_STATUS_DATA_TRANSFER_ERROR**
Unrecoverable data transfer error

**UWBAPI_STATUS_NO_CREDIT_AVAILABLE**
Credit not available for Data Packet

**UWBAPI_STATUS_ERROR_ROUND_INDEX_NOT_ACTIVATED**
Given round index couldn’t be activated

**UWBAPI_STATUS_ERROR_NUMBER_OF_ACTIVE_RANGING_ROUNDS_EXCEEDED**

Given round exceeds the possible number of ranging rounds

**UWBAPI_STATUS_ERROR_ROUND_INDEX_NOT_SET_AS_INITIATOR**

The role for the configured ranging round index is not Initiator and therefore RDM list cannot be set

**UWBAPI_STATUS_DLTDOA_DEVICE_ADDRESS_NOT_MATCHING_IN_REPLY_TIMELIST**

Device address not matching

**UWBAPI_STATUS_BUFFER_OVERFLOW**

Buffer overflow

**UWBAPI_STATUS_PBF_PKT_SENT**

Status PBF=1 CMD SENT

**UWBAPI_STATUS_HPD_WAKEUP**

Device is woken up from HPD

**UWBAPI_STATUS_TIMEOUT**

Command failed with time out

**UWBAPI_STATUS_ESE_ERROR**

SE Error

**UWBAPI_STATUS_SUSPEND**

Ranging suspended

**UWBAPI_STATUS_OK_DTPCM_CONFIG_SUCCESS**

DTPCM Status Ok, reported when DTPCM is configured for given MAC Address

**UWBAPI_STATUS_ERROR_INVALID_SLOT_BITMAP**

INVALID_SLOT_BITMAP Shall be reported when configured slot bit map size exceeds RANGING_DURATION

**UWBAPI_STATUS_ERROR_DUPLICATE_SLOT_ASSIGMENT**

DUPLICATE_SLOT_ASSIGMENT Shall be reported when configured slot assignments is inconsistent, i.e., one slot is assigned to more than one FiRA device

**UWBAPI_STATUS_ERROR_INVALID_MAC_ADDRESS**
INVALID_MAC_ADDRESS Shall be reported when given MAC address is not found

**UWBAPI_STATUS_UNKNOWN**
It is not known whether the intended operation was failed or successful.

**UWBAPI_UCI_DEV_INIT**
Device Status codes, as per UCI.

Device State - INIT

**UWBAPI_UCI_DEV_READY**
Device State - READY

**UWBAPI_UCI_DEV_ACTIVE**
Device State - ACTIVE

**UWBAPI_UCI_DEV_ERROR**
Device State - ERROR

**UWBAPI_SESSION_INIT_SUCCESS**
Session State codes, as per UCI.

Session State - Session Init Success

**UWBAPI_SESSION_INIT_SUCCESS**
Session State codes, as per UCI.

Session State - Session Init Success

**UWBAPI_SESSION_DEINIT_SUCCESS**
Session State - Session DeInit Success

**UWBAPI_SESSION_ACTIVATED**
Session State - Session Activated

**UWBAPI_SESSION_IDLE**
Session State - Session Idle

**UWBAPI_SESSION_ERROR**

Session State - Error

**UWBAPI_DATA_TRANSFER_STATUS_REPETITION_OK**

Data Transfer related status codes.

Data transmission is ongoing.

**UWBAPI_DATA_TRANSFER_STATUS_OK**

Data transmission is completed.

**UWBAPI_DATA_TRANSFER_STATUS_ERROR**

Application Data could not be sent due to an unrecoverable error

**UWBAPI_DATA_TRANSFER_STATUS_NO_CREDIT_AVAILABLE**

DATA_MESSAGE_SND is not accepted as no credit is available

**UWBAPI_DATA_TRANSFER_STATUS_REJECTED**

DATA_MESSAGE_SND packet sent in wrong state or Application Data Size exceeds the maximum size that can be sent in one Ranging Round.

**UWBAPI_DATA_TRANSFER_STATUS_TYPE_NOT_SUPPORTED**

Data transfer is not supported for given session type.

**UWBAPI_DATA_TRANSFER_DATA_TRANSFER_IS_ONGOING**

Application Data is being transmitted and the number of configured DATA_REPETITION_COUNT transmissions is not yet completed

**UWBAPI_DATA_TRANSFER_STATUS_INVALID_FORMAT**

The format of the command DATA_MESSAGE_SND associated with this notification is incorrect (e.g, a parameter is missing, a parameter value is invalid).

**ULTDOA_MESSAGE_CONTROL_MASK**

UL-TDoA related definitions.

UL-TDoA Message Control Mask

**ULTDOA_DEVICE_ID_16BIT_VALUE**

16-bit UL-TDoA Device ID

`ULTDOA_DEVICE_ID_32BIT_VALUE`
32-bit UL-TDoA Device ID

`ULTDOA_DEVICE_ID_64BIT_VALUE`
64-bit UL-TDoA Device ID

`ULTDOA_DEVICE_ID_MASK`
MASK for above values.

`ULTDOA_40BIT_TX_TIMESTAMP_MASK`
40-bit TX timestamp

`ULTDOA_64BIT_TX_TIMESTAMP_MASK`
64-bit TX timestamp

`ULTDOA_64BIT_RX_TIMESTAMP_MASK`
64-bit RX timestamp

`ULTDOA_16BIT_IN_BYTES`
Length in bytes for 16-bits

`ULTDOA_32BIT_IN_BYTES`
Length in bytes for 32-bits

`ULTDOA_40BIT_IN_BYTES`
Length in bytes for 40-bits

`ULTDOA_64BIT_IN_BYTES`
Length in bytes for 64-bits

`AUTH_TAG_IN_16BYTES`
Length in Bytes for 128-bits

`MAX_NUM_ANTENNA_PAIR`
Max number of supported rx antenna ID/pair

`FOV_SPECIFIC_DATA_TYPE`
FoV specific Vendor Data type

## Enums

enum **eSESSION_STATUS_REASON_CODES_t**

Session status error reason code 0x06 - 0x1C : RFU 0x40 - 0x7F : RFU 0x80 - 0xFF : Reserved for Vendor Specific Use 0xA0 - 0xAF : Reserved for CCC Specific Use.

*Values:*

enumerator **UWB_SESSION_STATE_CHANGED**

enumerator **UWB_SESSION_MAX_RR_RETRY_COUNT_REACHED**

enumerator **UWB_SESSION_MAX_RANGING_BLOCKS_REACHED**

enumerator **UWB_SESSION_SUSPENDED_DUE_TO_INBAND_SIGNAL**

enumerator **UWB_SESSION_RESUMED_DUE_TO_INBAND_SIGNAL**

enumerator **UWB_SESSION_STOPPED_DUE_TO_INBAND_SIGNAL**

enumerator **UWB_SESSION_INVALID_UL_TDOA_RANDOM_WINDOW**

enumerator **UWB_SESSION_MIN_RFRAMES_PER_RR_NOT_SUPPORTED**

enumerator **UWB_SESSION_TX_DELAY_NOT_SUPPORTED**

enumerator **UWB_SESSION_SLOT_LENTGH_NOT_SUPPORTED**

enumerator **UWB_SESSION_SLOTS_PER_RR_NOT_SUFFICIENT**

enumerator **UWB_SESSION_MAC_ADDRESS_MODE_NOT_SUPPORTED**

enumerator **UWB_SESSION_INVALID_RANGING_DURATION**

enumerator **UWB_SESSION_INVALID_STS_CONFIG**

enumerator **UWB_SESSION_HUS_INVALID_RFRAME_CONFIG** enumerator UWB_SESSION_HUS_NOT_ENOUGH_SLOTS

enumerator UWB_SESSION_HUS_CFP_PHASE_TOO_SHORT

enumerator UWB_SESSION_HUS_CAP_PHASE_TOO_SHORT

enumerator UWB_SESSION_HUS_OTHERS

enumerator UWB_SESSION_STATUS_SESSION_KEY_NOT_FOUND

enumerator UWB_SESSION_STATUS_SUB_SESSION_KEY_NOT_FOUND

enumerator UWB_SESSION_INVALID_PREAMBLE_CODE_INDEX

enumerator UWB_SESSION_INVALID_SFD_ID

enumerator UWB_SESSION_INVALID_PSDU_DATA_RATE

enumerator UWB_SESSION_INVALID_PHR_DATA_RATE

enumerator UWB_SESSION_INVALID_PREAMBLE_DURATION

enumerator UWB_SESSION_INVALID_STS_LENGTH

enumerator UWB_SESSION_INVALID_NUM_OF_STS_SEGMENTS

enumerator UWB_SESSION_INVALID_NUM_OF_CONTROLEES

enumerator UWB_SESSION_MAX_RANGING_REPLY_TIME_EXCEEDED

enumerator UWB_SESSION_INVALID_DST_ADDRESS_LIST

enumerator UWB_SESSION_INVALID_OR_NOT_FOUND_SUB_SESSION_ID

enumerator UWB_SESSION_INVALID_RESULT_REPORT_CONFIG enumerator UWB_SESSION_INVALID_RANGING_ROUND_CONTROL_CONFIG

enumerator UWB_SESSION_INVALID_RANGING_ROUND_USAGE

enumerator UWB_SESSION_INVALID_MULTI_NODE_MODE

enumerator UWB_SESSION_RDS_FETCH_FAILURE

enumerator UWB_SESSION_DOES_NOT_EXIST

enumerator UWB_SESSION_RANGING_DURATION_MISMATCH

enumerator UWB_SESSION_INVALID_OFFSET_TIME

enumerator UWB_SESSION_LOST

enumerator UWB_SESSION_DT_ANCHOR_RANGING_ROUNDS_NOT_CONFIGURED

enumerator UWB_SESSION_DT_TAG_RANGING_ROUNDS_NOT_CONFIGURED

enumerator UWB_SESSION_ERROR_INVALID_ANTENNA_CFG

enumerator UWB_SESSION_BASEBAND_ERROR

enumerator UWB_SESSION_TESTMODE_TERMINATED

enumerator UWB_SESSION_INVALID_DATA_TRANSFER_MODE

enumerator UWB_SESSION_INVALID_MAC_CFG

enumerator UWB_SESSION_INVALID_ADAPTIVE_HOPPING_THRESHOLD

enumerator UWB_SESSION_UNSUPPORTED_RANGING_LIMIT

enumerator UWB_SESSION_RNG_INVALID_DEVICE_ROLE enumerator **UWB_SESSION_INVALID_ANTENNA_PAIR_SWAP_CONFIGURATION**

enumerator **UWB_SESSION_URSK_EXPIRED**

enumerator **UWB_SESSION_TERMINATION_ON_MAX_STS**

enumerator **UWB_SESSION_RADAR_FCC_LIMIT_REACHED**

enumerator **UWB_SESSION_CSA_INVALID_CFG**

## 6.2 UWB API Consts

*group* **uwb_const**

Some helper constants for UWB APIs.

## 6.3 UWB API Data Types

*group* **uwb_types**

Various structrues and data types used by UWB.

### Defines

**MODELID_BOARD_TYPE**

**MODELID_BOARD_TYPE**

**MODELID_CHIP_TYPE**

**MODELID_CHIP_TYPE**

**MODELID_RFU**

**CLOCK_DRIFT**

**kUWB_RangingMethod_TDoA**

**kUWB_RangingMethod_SS_TWR**

**kUWB_RangingMethod_DS_TWR**

**kUWB_RangingMethod_SS_TWR_ND**

**kUWB_RangingMethod_DS_TWR_ND**

## Typedefs

typedef uint8_t **tUWBAPI_STATUS**

Types used by Uwb APIs.

Status used by UWB APIs.

typedef UWB_RangingRoundUsage_t **UWB_RangingMethod_t**

**void() tUwbApi_AppCallback (eNotificationType opType, void *pData)**

Generic callback function registered by Application to receive data from Rhodes SDK. This is a common callback for All session types.

a. Ranging
b. App Data Transfer
c. PER Exchange
d. Generic Error Notification
e. Device Reset Notification
f. RFrame Data Notification
g. Recovery Notification
h. Scheduler status Notification

This data is provided by the stack, to the user. The user should implement this callback if they want to receive the NTF information.

- If the callback function pointer is NULL, the user will not receive the notifications.

**Param opType**
Type of Notification

**Param pData**
Data received

`void() tUwbApi_AppDataCallback (uint8_t *recvData, uint16_t recvDataLen)`

Generic callback function registered by Application to receive data packet from Rhodes SDK. If the callback function pointer is NULL, the user will not receive the data packet.

**Param recvData**
pointer to received data

**Param recvDataLen**
len of data received

### Enums

enum **resetType**

RESET Modes.

*Values:*

enumerator **UWBD_RESET**

UWB Device Reset

enumerator **RFU**

RFU

enum **session_type**

UWBD Session Type.

*Values:*

enumerator **UWBD_RANGING_SESSION**

Ranging Session

enumerator **UWBD_RANGING_WITH_INBAND_DATA_TRANSFER**

Inband data transfer

enumerator **UWBD_DATA_TRANSFER**

Data Transfer Session

enumerator **UWBD_RANGING_ONLY_PHASE**

Ranging Only Phase

enumerator **UWBD_INBAND_DATA_PHASE**

Inband Data Phase

enumerator **UWBD_RANGING_WITH_DATA_PHASE**

Ranging With Data Phase

enumerator **UWBD_CCC_SESSION**

CCC Session

enumerator **UWBD_CSA_SESSION**

Aliro Session

enumerator **UWBD_RFTEST**

Test Session

enumerator **UWBD_RADAR_TRANSFER**

Radar Trasnfer Session

enumerator **UWBD_RFU**

RFU

enum **linkLayerModes**

Modes for setting LINK_LAYER_MODE app config, supported in UWB API layer.

*Values:*

enumerator **Link_Layer_Mode_Bypass**

Bypass mode. (Default)

enumerator **Link_Layer_Mode_ConnectionLess**

Connection-Less mode.

enumerator **Link_Layer_Mode_ConnectionOriented**

Connection mode.

enum **notification_type**

UWBD notification Type.

*Values:*

enumerator **UWBD_RANGING_DATA**

Ranging Data Notification

enumerator **UWBD_DATA_TRANSMIT_NTF**

Data Transmit Notification

enumerator **UWBD_PER_SEND**

PER Packet Sent Notification

enumerator **UWBD_PER_RCV**

PER receive operation completed notification

enumerator **UWBD_GENERIC_ERROR_NTF**

Generic Error Notification

enumerator **UWBD_DEVICE_RESET**

Upon Receiving Device Reset, Application needs to clear all session context and re-initiate all the sessions

enumerator **UWBD_RFRAME_DATA**

RFRAME Data Notification

enumerator **UWBD_RECOVERY_NTF**

Upon receiving Recovery Notification, Application has to invoke Recovery API non main/application thread context.

enumerator **UWBD_SCHEDULER_STATUS_NTF**

UWBS shall send SCHED_STATUS_NTF notification when scheduler meets either SESSION_SCHEDULER_ATTEMPTS or SESSION_SYNC_ATTEMPTS configuration criteria.

enumerator **UWBD_SESSION_DATA**

Session Data Notification

enumerator **UWBD_MULTICAST_LIST_NTF**

Multicast list notification

enumerator **UWBD_OVER_TEMP_REACHED**

Over Temperature reached notification

enumerator **UWBD_BLINK_DATA_TX_NTF**

Blink data tx notification

enumerator **UWBD_DATA_TRANSFER_PHASE_CONFIG_NTF**

Data Tx Phase Configuration Notification

enumerator **UWBD_ACTION_APP_CLEANUP**

TEST RX notification Perform Application Clean Up & Restart upon receiving this notification

enumerator **UWBD_TEST_MODE_LOOP_BACK_NTF**

Loopback Status Data

enumerator **UWB_TEST_PHY_LOG_NTF**

PHY LOG NTF

enumerator **UWBD_TEST_RX_RCV**

TEST RX notification

enumerator **UWBD_DATA_RECV_NTF**

Data receive

enumerator **UWBD_CIR_DATA_NTF**

CIR notification data

enumerator **UWBD_DATA_LOGGER_NTF**

Data logger ntf

enumerator **UWBD_PSDU_DATA_NTF**
PSDU log ntf

enumerator **UWBD_RANGING_TIMESTAMP_NTF**
RANGING timestamp ntf

enumerator **UWBD_DATA_RCV_NTF**
Data receive notification

enumerator **UWBD_RADAR_RCV_NTF**
Radar rcv ntf

enumerator **UWBD_TEST_RADAR_ISO_NTF**
Test radar Isolation ntf

enumerator **UWBD_WIFI_COEX_IND_NTF**

enumerator **UWBD_MAX_ACTIVE_GRANT_DURATION_EXCEEDED_WAR_NTF**
Max Active Grant Duration Exceeded Warning NTF

enum **demo_chip_type_t**
Enumeration lists out the chip specific type used.

*Values:*

enumerator **kChipType_NA**

enumerator **kChipType_SR150**

enumerator **kChipType_SR040**

enumerator **kChipType_SR160**

enum **demo_board_type_t**
Enumeration lists out the board specific type used.

*Values:*

enumerator **kBoardType_NA**

enumerator **kBoardType_Shield**

enumerator **kBoardType_FinderV3**

enum **PreferedUpdateRate_t**

Enumeration lists out the Preferred specific rate type used.

*Values:*

enumerator **kUpdateRate_Automatic**

enumerator **kUpdateRate_Infrequent**

enumerator **kUpdateRate_UserInteractive**

enum **eAntennaSelect_t**

Enumeration lists out the Preferred Antenna used.

*Values:*

enumerator **kAntennaSecect_AntDefault**

enumerator **kAntennaSecect_AntTop**

enumerator **kAntennaSecect_AntBottom**

enum **UWB_ChannelNumber**

Enumeration lists out the channel numbers used.

*Values:*

enumerator **kUWB_ChannelNumber_5**

enumerator **kUWB_ChannelNumber_6**

enumerator **kUWB_ChannelNumber_8**

enumerator **kUWB_ChannelNumber_9**

enum **UWB_MacFcsType**

Enumeration lists out the MAC FCS type used.

*Values:*

enumerator **kUWB_MacFcsType_CRC16**

enumerator **kUWB_MacFcsType_CRC32**

enum **UWB_SfdId**

Enumeration lists out the SFD ID's used.

*Values:*

enumerator **kUWB_SfdId_BPRF_0**

enumerator **kUWB_SfdId_BPRF_2**

enumerator **kUWB_SfdId_HPRF_1**

enumerator **kUWB_SfdId_HPRF_2**

enumerator **kUWB_SfdId_HPRF_3**

enum **UWB_PreambleIndxCode**

Enumeration lists out the Preamble Index codes used.

*Values:*

enumerator **kUWB_PreambleIndxCode_BPRF_09**

enumerator **kUWB_PreambleIndxCode_BPRF_10**

enumerator **kUWB_PreambleIndxCode_BPRF_11**

enumerator **kUWB_PreambleIndxCode_BPRF_12** enumerator **kUWB_PreambleIndxCode_HPRF_25**

enumerator **kUWB_PreambleIndxCode_HPRF_26**

enumerator **kUWB_PreambleIndxCode_HPRF_27**

enumerator **kUWB_PreambleIndxCode_HPRF_28**

enumerator **kUWB_PreambleIndxCode_HPRF_29**

enumerator **kUWB_PreambleIndxCode_HPRF_30**

enumerator **kUWB_PreambleIndxCode_HPRF_31**

enumerator **kUWB_PreambleIndxCode_HPRF_32**

enum **UWB_PreambleDuration**

Enumeration lists out the Preamble duration used.

*Values:*

enumerator **kUWB_PreambleDuration_32Symbols**

enumerator **kUWB_PreambleDuration_64Symbols**

enum **UWB_Session_InfoNtf**

Enumeration lists out the Ranging Data Notifications used.

*Values:*

enumerator **kUWB_DisableSession_Info_Ntf**

enumerator **kUWB_EnableSession_Info_Ntf**

enumerator **kUWB_SessionInfo_Ntf_Proximity**

enumerator **kUWB_SessionInfo_Ntf_AOABounds** enumerator **kUWB_SessionInfo_Ntf_AOABounds_n_Proximity**

enumerator **kUWB_SessionInfo_Ntf_Enter_Leave_Proximity**

enumerator **kUWB_SessionInfo_Ntf_Enter_Leave_AOABounds**

enumerator **kUWB_SessionInfo_Ntf_Enter_Leave_AOABounds_n_Proximity**

enum **UWB_CoEx_InterfaceSelection**

Enumeration lists out the CoEx Interface to be selected.

*Values:*

enumerator **kUWB_CoEx_Gpio_Interface**

UWB_WiFiCoEx_AdvGrantDuration

enumerator **kUWB_CoEx_Uart_Interface**

Set CoEx Interface as Uart Interface

enumerator **kUWB_CoEx_OneWire_Interface**

Set CoEx Interface as One Wire Interface.

enum **UWB_CoEx_NtfSelection**

Enumeration lists out the configurations for the WiFi CoEx feature Notifications.

*Values:*

enumerator **kUWB_CoEx_Disable**

Disable Wifi CoEx Feature (default)

enumerator **kUWB_CoEx_En_WoDebug_WoWarning**

Enable CoEx Interface without Debug and without Warning Verbose.

enumerator **kUWB_CoEx_En_Debug**

Enable CoEx Interface with Debug Verbose only

enumerator **kUWB_CoEx_En_Warning**

Enable CoEx Interface with Warnings Verbose only enumerator **kUWB_CoEx_En_Debug_Warning**

Enable CoEx Interface with both Debug and Warning Verbose

enum **UWB_CoEx_ChannelCfg**

Enumeration lists out the configurations for the WiFi CoEx feature.

*Values:*

enumerator **kUWB_CoEx_CH5**

Enable Wifi CoEx on Channel 5

enumerator **kUWB_CoEx_CH6**

Enable Wifi CoEx on Channel 6

enumerator **kUWB_CoEx_CH8**

Enable Wifi CoEx on Channel 8

enumerator **kUWB_CoEx_CH9**

Enable Wifi CoEx on Channel 9

enumerator **kUWB_CoEx_AllCH**

Enable Wifi CoEx for all channels i.e ch5,ch6,ch8 and ch9

enum **UWB_PrfMode**

Enumeration lists out the PRF modes used.

*Values:*

enumerator **kUWB_PrfMode_62_4MHz**

BPRF

enumerator **kUWB_PrfMode_124_8MHz**

HPRF

enumerator **kUWB_PrfMode_249_6MHz**

HPRF mode with data rate 27.2 and 31.2 Mbps

enum **UWB_PsduDataRate**

Enumeration lists out the PSDU data rates used.

*Values:*

enumerator **kUWB_PsduDataRate_6_81Mbps**

enumerator **kUWB_PsduDataRate_7_80Mbps**

enumerator **kUWB_PsduDataRate_27_2Mbps**

enumerator **kUWB_PsduDataRate_31_2Mbps**

enumerator **kUWB_PsduDataRate_850Kbps**

enum **UWB_RfFrameConfig**

Enumeration lists out the RFrame Config sts values.

*Values:*

enumerator **kUWB_RfFrameConfig_No_Sts**

enumerator **kUWB_RfFrameConfig_SP0**

enumerator **kUWB_RfFrameConfig_Sfd_Sts**

enumerator **kUWB_RfFrameConfig_SP1**

enumerator **kUWB_RfFrameConfig_Psdu_Sts**

enumerator **kUWB_RfFrameConfig_Sfd_Sts_NoPhr_NoPsdu**

enumerator **kUWB_RfFrameConfig_SP3**

enum **UWB_DeviceRole_t**

Enumeration lists out the Device role values.

*Values:*

enumerator **kUWB_DeviceRole_Responder** enumerator **kUWB_DeviceRole_Initiator**

enumerator **kUWB_DeviceRole_UT_Sync_Anchor**

enumerator **kUWB_DeviceRole_UT_Anchor**

enumerator **kUWB_DeviceRole_UT_Tag**

enumerator **kUWB_DeviceRole_Advertiser**

enumerator **kUWB_DeviceRole_Observer**

enumerator **kUWB_DeviceRole_DlTDoA_Anchor**

enumerator **kUWB_DeviceRole_DlTDoA_Tag**

enum **UWB_MultiNodeMode**

Enumeration lists out the Multicast mode values.

*Values:*

enumerator **kUWB_MultiNodeMode_UniCast**

enumerator **kUWB_MultiNodeMode_OnetoMany**

enumerator **kUWB_MultiNodeMode_ManytoMany**

enum **UWB_DeviceType**

Enumeration lists out the Device type values.

*Values:*

enumerator **kUWB_DeviceType_Controlee**

enumerator **kUWB_DeviceType_Controller** enum **UWB_RangingRoundUsage**

Enumeration lists out the Ranging Round Usage values.

*Values:*

enumerator **kUWB_RangingRoundUsage_TDoA**

enumerator **kUWB_RangingRoundUsage_SS_TWR**

enumerator **kUWB_RangingRoundUsage_DS_TWR**

enumerator **kUWB_RangingRoundUsage_SS_TWR_nd**

enumerator **kUWB_RangingRoundUsage_DS_TWR_nd**

enumerator **kUWB_RangingRoundUsage_DL_TDOA**

enumerator **kUWB_RangingRoundUsage_OWR_AOA**

enumerator **kUWB_RangingRoundUsage_eSS_TWR**

enumerator **kUWB_RangingRoundUsage_aDS_TWR**

enumerator **kUWB_RangingRoundUsage_DTx**

enum **UWB_AOA_Result_Req**

*Values:*

enumerator **kUWB_AOA_Result_Req_Disable**

enumerator **kUWB_AOA_Result_Req_Enable**

enumerator **kUWB_AOA_Result_Req_Azimuth**

enumerator **kUWB_AOA_Result_Req_Elevation** enum **UWB_StsConfig**

Enumeration lists out the STS Config values.

*Values:*

enumerator **kUWB_StsConfig_StaticSts**

enumerator **kUWB_StsConfig_DynamicSts**

enumerator **kUWB_StsConfig_DynamicSts_Ctrlee_key**

enumerator **kUWB_StsConfig_ProvisionSts**

enumerator **kUWB_StsConfig_ProvisionSts_Ctrlee_key**

enum **UWB_MacAddressMode_t**

Enumeration lists out the MAC address mode.

*Values:*

enumerator **kUWB_MacAddressMode_2bytes**

enumerator **kUWB_MacAddressMode_8bytes**

enum **UWB_ScheduledMode**

Enumeration lists out the scheduled Mode values.

*Values:*

enumerator **kUWB_ScheduledMode_ContentionBased**

Contention based Ranging Scheduling

enumerator **kUWB_ScheduledMode_TimeScheduled**

Time based Ranging Scheduling

enumerator **kUWB_ScheduledMode_HybridBased**

Hybrid based Scheduling

enum **UWB_RadarMode**

Enumeration lists out the Radar Mode values.

0x01: Medium distance, e.g. used for vital sign detection; 0x02: Close distance, e.g. used for hand gesture recognition 0x03: Far distance, e.g. used for presence detection 0x04 - RFU 0x05: Medium distance, e.g. used for static object support 0x06: Far distance, e.g. used for static object support

*Values:*

enumerator **kUWB_RadarMode_Medium_Distance**

enumerator **kUWB_RadarMode_Close_Distance**

enumerator **kUWB_RadarMode_Far_Distance**

enumerator **kUWB_RadarMode_Static_Medium_Distance**

enumerator **kUWB_RadarMode_Static_Far_Distance**

enumerator **kUWB_RadarMode_Test_Isolation**

enum **UWB_phaseParticipationCtrl**

Enumeration lists out the Phase Participation values of Controller.

0x00 = No participation in the phase 0x01 = Participate in the phase as a device role configured by DEVICE_ROLE 0x02 = The UWBS shall transmit DTPCM in this phase 0x03 = The UWBS shall Receive DTPCM in this phase Note : 0x02 and 0x03 are applicable only for Data Transfer Phase.

*Values:*

enumerator **kUWB_CtrlNoParticipation**

enumerator **kUWB_CtrlDeviceRoleParticipation**

enumerator **kUWB_CtrlDtpcmTxParticipation**

enumerator **kUWB_CtrlDtpcmRxParticipation** enum **UWB_phaseParticipationClee**

Enumeration lists out the Phase Participation values of Controlee.

0x00 = No participation in the phase. 0x01 = Participate as a Responder Role if MAC Address of this phase is not present in the RMM. 0x02 = Participate in the phase as a device role configured by DEVICE_ROLE.

*Values:*

enumerator **kUWB_CleeNoParticipation**

enumerator **kUWB_CleeResponderRoleParticipation**

enumerator **kUWB_CleeDeviceRoleParticipation**

enum **phSlotBitmap**

Set the Slot Bitmap [b3-b1] 0 = 8 ranging slots 1 = 16 ranging slots 2 = 32 ranging slots 3 = 64 ranging slots 4 = 128 ranging slots 5 = 256 ranging slots 6-7 = RFU

*Values:*

enumerator **ranging_slots_8**

enumerator **ranging_slots_16**

enumerator **ranging_slots_32**

enumerator **ranging_slots_64**

enumerator **ranging_slots_128**

enumerator **ranging_slots_256**

enum **multicastControllerActions**

Enumeration to list out the Actions of the Controller's Multicast List.

*Values:*

enumerator **kUWB_AddControlee**

enumerator **kUWB_DelControlee**

enumerator **KUWB_Add16BSubSessionKey**

enumerator **KUWB_Add32BSubSessionKey**

enum **UWB_ProfileID**

Enumeration lists out the Vendor specific Profile ID.

*Values:*

enumerator **kUWB_Profile_1**

struct **phUwbSessionData**

*#include <UwbApi_Types.h>* Structure for storing Session Data.

### Public Members

uint32_t **sessionHandle**

Session Handle

uint8_t **session_type**

Session Type

uint8_t **session_state**

Session State

struct **phUwbSessionsContext**

*#include <UwbApi_Types.h>* Structure for storing Session Context.

### Public Members

uint8_t **status**

Status

uint8_t **sessioncnt**

[Input/Output] Session Count

phUwbSessionData_t ***pUwbSessionData**

Pointer to Session Data

struct **phRangingMesr**

*#include <UwbApi_Types.h>* Structure lists out the ranging measurement information. Ranging measurements array — TWR.

### Public Members

uint8_t **mac_addr**[8]

Mac address of the participating device, addr can be of short 2 byte or extended 8 byte modes

uint8_t **status**

Status

uint8_t **nLos**

Indicates if the ranging measurement was in Line of sight or non-line of sight

uint8_t **aoa_azimuth_FOM**

AOA Azimuth FOM

uint8_t **aoa_elevation_FOM**

AOA elevation FOM

uint8_t **aoa_dest_azimuth_FOM**

AOA destination azimuth FOM

uint8_t **aoa_dest_elevation_FOM**

AOA destination elevation FOM

uint8_t **slot_index**

Status to the slot number within the ranging round In time schedule mode, in case of a failure, this field indicates the slot number within the ranging round where the failure has occurred In contention-based ranging mode, this field can be used to indicate the slot selected by the controlee device. If the Slot Index field is not used, then it shall be set to 0.

uint8_t **rssi**

Rssi

uint16_t **distance**

Distance in centimeters

int16_t **aoa_azimuth**

AOA Azimuth

int16_t **aoa_elevation**

AOA Elevation

int16_t **aoa_dest_azimuth**

AOA destination azimuth

int16_t **aoa_dest_elevation**

AOA destination elevation

struct **phRxInfoMesr**

*#include <UwbApi_Types.h>* Structure lists out the Vendor speicifc Rx Antenna Info for AoA Measurements.

### Public Members

uint8_t **rxMode**

RX mode

uint8_t **num_of_rx_antennaRxInfo**

Number of RX antenna to follow

uint8_t **rx_antennaIdRxInfo**[8]

RX antenna pair

struct **phRxInfoDebugNtf**

*#include <UwbApi_Types.h>* Structure lists out the Vendor speicifc Rx Antenna Info For Debug Notifications.

### Public Members

uint8_t **rxModeDebugNtf**

RX mode

uint8_t **num_of_rx_antennaDebugNtf**

Number of RX antenna to follow

uint8_t **rx_antennaIdDebugNtf**[8]

RX antenna pair

struct **phAoaPdoaMesr**

*#include <UwbApi_Types.h>* Structure lists out the Vendor speicifc information for AoA / PDoA measurements per RX.

### Public Members

int16_t **angleOfArrival**

Angle of arrival

int16_t **pdoa**

Phase difference of arrival

uint16_t **pdoaIndex**

Phase difference of arrival index in the whole CIR

uint8_t **aoaFovFlag**

This parameter indicates the presence or absence of the peer device in FoV for user configured Horizontal Rx Antenna Pair as defined in 'AZIMUTH_FIELD_OF_VIEW' 0x00: Peer device is not present in FoV 0x01: Peer device is present in FoV This field would be displayed only when NXP Specific Data Type is 0xA0

struct **phSnrPathIndexMesr**

*#include <UwbApi_Types.h>* Structure lists out the Vendor speicifc Rx Antenna information SNRFirst / SNRMain /FirstIndex : Main Index measurements per RX entry.

### Public Members

uint8_t **rxSnrFirstPath**

Signal-to-Noise (SNR) of the First Path in dB

uint8_t **rxSnrMainPath**

Signal-to-Noise (SNR) of the Main Path in dB

int16_t **rx_FirstPathIndex**

First path index in the whole CIR

int16_t **rx_MainPathIndex**

Main path index in the whole CIR

struct **phTwoWayRangingVsMesr**

*#include <UwbApi_Types.h>* Structure lists out the TWR Vendor speicifc measurements for each responder.

### Public Members

phAoaPdoaMesr_t **aoaPdoaMesr_twr**[8]

AoA / PDoA measurements per RX

phSnrPathIndexMesr_t **snrPathIndexMesr_twr**[8]

SNRFirst / SNRMain / FirstIndex / Main Index measurements per RX

uint16_t **distance_2**

Range or distance between the device and target Set to 0 for Initiator: distance-2 is reported using the RFM pair on responder only in AOA_RFM mode

struct **phTwoWayRangingVsData**

*#include <UwbApi_Types.h>* Structure lists out the Vendor speicifc information for TWR.

### Public Members

uint8_t **wifiCoExStatus**

Vendor Specific Data WiFi co-existence status

phRxInfoMesr_t **rxInfoMesr_twr**

Rx Antenna Info for AoA Measurements

phRxInfoDebugNtf_t **rxInfoDebugNtf_twr**

Rx Antenna Info For Debug Notifications

phTwoWayRangingVsMesr_t **vsMesr**[12]

twr vs measurment for each responders

struct **phOneWayRangingVsMesr**

*#include <UwbApi_Types.h>* Structure lists out the OWR Vendor speicifc measurements for each responder.

### Public Members

phAoaPdoaMesr_t **aoaPdoaMesr_owr**[8]

AoA / PDoA measurements per RX

int16_t **rssi**

RSSI in dB

struct **phOneWayRangingVsData**

*#include <UwbApi_Types.h>* Structure lists out the Vendor speicifc information for OWR.

### Public Members

phRxInfoMesr_t **rxInfoMesr_owr**

Vendor Specific Data Rx Antenna Info for AoA Measurements

phOneWayRangingVsMesr_t **vsMesr**[MAX_NUM_OF_TDOA_MEASURES]

owr vs measurment for each responders

struct **phRangingMesrTdoa**

*#include <UwbApi_Types.h>* Structure lists out the TDoA ranging measurement information — one way ranging.

### Public Members

uint8_t **mac_addr**[8]

Mac address of the participating device, addr can be of short 2 byte or extended 8 byte modes

uint8_t **status**

Status

uint8_t **message_control**

This signifies the presence of Device ID, Rx and Tx timestamps

uint8_t **frame_type**

Type of frame

uint8_t **nLos**

Indicates if the ranging measurement was in Line of sight or non-line of sight

int16_t **aoa_azimuth**

AOA Azimuth

uint8_t **aoa_azimuth_FOM**

AOA Azimuth FOM

int16_t **aoa_elevation**

AOA elevation

uint8_t **aoa_elevation_FOM**

AOA elevation FOM

uint32_t **frame_number**

Number received in the Payload of the blink UTM from the UT-Tag or UTM from the UT-Synchronization Anchor.

uint8_t **rx_timestamp**[(0x08)]

Local RX timestamp of the received UWB RFRAME as measured by the device that is sending this SESSION_INFO_NTF. The unit is 2^(-7) of the 499.2 MHz chipping period, which is approximately 15.65ps.

uint8_t **ul_tdoa_device_id**[(0x08)]

UL-TDoA Device ID of the sender of the received UTM as listed in the Blink UTM from the UT-Tag or the Synchronization UTM from the UT-Synchronization Anchor.

uint8_t **tx_timestamp**[(0x08)]

TX timestamp of the UWB RFRAME as listed in the Blink UTM from the UT-Tag or the Synchronization UTM from UT-Synchronization Anchor. The unit is 2^(-7) of the 499.2 MHz chipping period, which is approximately 15.65ps.

struct **phTdoaRangingVsData**

*#include <UwbApi_Types.h>* Structure lists out the Vendor speicifc information for tdoa.

### Public Members

uint8_t **vendorExtLength**

Vendor Specific Data Vendor extension length for TDOA Measurements

int16_t **rssi_rx1**

RSSI RX1

int16_t **rssi_rx2**

RSSI RX2

uint8_t **noOfPdoaMeasures**

No of pdoA measurement

int16_t **pdoa**[8]

Estimation of phase difference for antenna pair N

uint16_t **pdoaIndex**[8]

CIR Index estimate at which pdoa has been detected struct **phRangingMesrOwrAoa_t**

*#include <UwbApi_Types.h>* Structure lists out the OWR with AoA ranging measurement information.

### Public Members

uint8_t **mac_addr**[8]

Mac address of the participating device, addr can be of short 2 byte or extended 8 byte modes

uint8_t **status**

Status

uint8_t **nLos**

Indicates if the ranging measurement was in Line of sight or non-line of sight

uint8_t **frame_seq_num**

frame sequence number

uint16_t **block_index**

block index

int16_t **aoa_azimuth**

AOA Azimuth

uint8_t **aoa_azimuth_FOM**

AOA Azimuth FOM

int16_t **aoa_elevation**

AOA Elevation

uint8_t **aoa_elevation_FOM**

AOA elevation FOM

union **RANGING_MEAS**

*#include <UwbApi_Types.h>* Union for TWR ranging and TDoA ranging measurement information.

### Public Members

phRangingMesr_t **range_meas_twr**[12]

Ranging measurements array

phRangingMesrTdoa_t **range_meas_tdoa**[MAX_NUM_OF_TDOA_MEASURES]

Ranging measurements TDoA array One Way Ranging Ntf

phRangingMesrDlTdoa_t **range_meas_dltdoa**[MAX_NUM_OF_TDOA_MEASURES]

Ranging measurements DLTDoA array

*phRangingMesrOwrAoa_t* **range_meas_owr_aoa**[MAX_NUM_OWR_AOA_MEASURES]

Ranging measurements OWR with AoA

union **VENDORSPECIFIC_MEAS**

*#include <UwbApi_Types.h>* Union for vendor speific Information of TWR ranging and TDoA ranging.

### Public Members

phTwoWayRangingVsData_t **twr**

Ranging measurements array

phOneWayRangingVsData_t **owr_aoa**

One Way Ranging Ntf

phTdoaRangingVsData_t **tdoa**

One Way Ranging TDoA Ntf

struct **phRangingData**

*#include <UwbApi_Types.h>* Structure lists out the ranging notification information.

### Public Members

uint8_t **rcr_indication**

RCR Indication

uint8_t **ranging_measure_type**

Ranging Measurement Type

uint8_t **mac_addr_mode_indicator**

Mac addr mode indicator,

- 0: short 2 byte,
- 1: extended 8 byte mode

uint8_t **no_of_measurements**

Number of ranging measurements

uint32_t **seq_ctr**

keep track of the notifications

uint32_t **sessionHandle**

Session Handle of the ranging round

uint32_t **curr_range_interval**

Current ranging interval setting in milli seconds

uint32_t **sessionHandle_of_primary_session**

Session Handle of Primary Session

*RANGING_MEAS* **ranging_meas**

Ranging measures array

uint8_t **vs_data_type**

Vendor specific data type

uint16_t **vs_length**

Vendor specific data Length

*VENDORSPECIFIC_MEAS* **vs_data**

Vendor specific data

struct **phRangingParams**

*#include <UwbApi_Types.h>* Structure lists out the mandatory configurations to be set for ranging.

### Public Members

uint8_t **deviceRole**

Device Role

kUWB_DeviceRole_Responder = 0, kUWB_DeviceRole_Initiator = 1, kUWB_DeviceRole_UT_Sync_Anchor = 2, kUWB_DeviceRole_UT_Anchor = 3, kUWB_DeviceRole_UT_Tag = 4, kUWB_DeviceRole_Advertiser = 5, kUWB_DeviceRole_Observer = 6, kUWB_DeviceRole_DlTDoA_Anchor = 7, kUWB_DeviceRole_DlTDoA_Tag = 8,

uint8_t **multiNodeMode**

Multi Node Mode,

- 0x00: Single device to Single device (Unicast),
- 0x01: One to Many,
- 0x02: Many to Many,
- 0x03: Reserved

uint8_t **macAddrMode**

Device Mac Address mode 0:2 bytes,1:8 bytes mac addr with 2 bytes in header, 2: 8 bytes in mac addr and header

uint8_t **deviceMacAddr**[8]

Device Mac Address, 2 bytes or 8 bytes addr is supported.

uint8_t **deviceType**

Device Type, 0x00: Controlee, 0x01: Controller, 0x02: Advertiser, 0x03: Observer

uint8_t **rangingRoundUsage**

Ranging Round Usage

uint8_t **scheduledMode**

Scheduled Mode

struct **phTxTelecConfig**

*#include <UwbApi_Types.h>* Structure lists out the preamble pulse shape config.

### Public Members

uint8_t **shape_id**

Preamble pulse shape id

uint8_t **payload_tx_shape_id**

Payload tx pulse shape id

uint8_t **sts_shape_id**

STS Tx pulse shape id

uint8_t **dac_stage_cofig**

DAC Stage config

struct **AccessoryConfigDataContent**

*#include <UwbApi_Types.h>* Structure lists out the configurations to be get from Accessory Config Data.

### Public Members

uint8_t **length**

Total length

uint8_t **uwb_spec_ver_major**[2]

Specification Major version

uint8_t **uwb_spec_ver_minor**[2]

Specification Minor version

uint8_t **manufacturer_id**[4]

Manufacture id for device specific

uint8_t **model_id**[4]

Model id for device specific

uint8_t **mw_version**[4]

MW version

uint8_t **ranging_role**

Device Role

uint8_t **device_mac_addr**[2]

Source mac address

uint8_t **clock_drift**[2]

CLock frequency drift value

struct **UwbPhoneConfigData**

*#include <UwbApi_Types.h>* Structure lists our the configurations to be loaded on the device sent from the phone counterpart.

struct **UwbDeviceConfigData**

*#include <UwbApi_Types.h>* Structure lists out the configurations to be get from Accessory Config Data.

### Public Members

uint8_t **spec_ver_major**[2]

Specification Major version

uint8_t **spec_ver_minor**[2]

Specification Minor version

uint8_t **chip_id**[2]

UWB Chip identifier

uint8_t **chip_fw_version**[2]

UWB Chip firmware version

uint8_t **mw_version**[3]

MW version

uint32_t **supported_config_ids**

Range of supported profiles by the device

uint8_t **ranging_role**

Device Role

uint8_t **device_mac_addr**[2]

Source mac address

struct **UserAccessoryConfigData_iOS**

*#include <UwbApi_Types.h>* Structure lists out the mandatory configurations to be set for User Accessory Config Data.

struct **phDataTransmit**

*#include <UwbApi_Types.h>* Structure lists out the data control transmit notification.

### Public Members

uint32_t **transmitNtf_sessionHandle**

Session Handle

uint16_t **transmitNtf_sequence_number**

Sequence number

uint8_t **transmitNtf_status**

Status

uint8_t **transmitNtf_txcount**

Tx count

struct **phDataCredit**

*#include <UwbApi_Types.h>* Structure lists out the data credit notification.

### Public Members

uint32_t **sessionHandle**

Session Handle

uint8_t **credit_availability**

Credit availability

struct **phUwbRadarcirNtf**

*#include <UwbApi_Types.h>* Structure lists out the Radar Cir notification.

### Public Members

uint16_t **num_cirs**

Num of Cirs

uint8_t **cir_taps**

Cir Taps

uint8_t **rfu**

Rfu

uint16_t **cir_len**

Length of the Cir

uint8_t ***cirdata**

Cir data

struct **phUwbRadarTestIsoNtf**

*#include <UwbApi_Types.h>* Structure lists out the Test Radar Antenna Isolation.

### Public Members

uint8_t **antenna_tx**

Tx Antenna select

uint8_t **antenna_rx**

Rx Antenna select

uint16_t **anteena_isolation**

Radar type

union **RADAR_MEAS**

*#include <UwbApi_Types.h>* Union for Radar and Anteena isolation measurement information.

### Public Members

phUwbRadarcirNtf_t **radr_cir**

Radar CIR notification Structure

phUwbRadarTestIsoNtf_t **radar_tst_ntf**

Radar CIR Antenna test isolation Structure

struct **phUwbRadarNtf**

*#include <UwbApi_Types.h>* Structure lists out the Radar notificaiton.

### Public Members

uint32_t **sessionHandle**

Session Handle

uint8_t **radar_status**

Status of the radar

uint8_t **radar_type**

Radar type

*RADAR_MEAS* **radar_ntf**

Radar measures

struct **UWB_CoEx_IndNtf**

*#include <UwbApi_Types.h>* Structure for storing WiFiCoEx IND Ntf Context.

### Public Members

uint8_t **status**

UWB_WLAN_IND Status

uint32_t **slot_index**

Slot Index where the GPIO change occured

uint32_t **sessionHandle**

Session Handle to which UWB_WIFI_COEX_IND_NTF belongs

struct **phCtlrPhaseList**

*#include <UwbApi_Types.h>* Structure for storing List of Phases of Controller.

### Public Members

uint32_t **phase_sessionHandle**

Secondary Session Handle

uint16_t **start_slot_index**

Start Slot Index

uint16_t **end_slot_index**

End Slot Index

uint8_t **phase_participation**

Phase Participation to indicate whether the device shall participate or not in the phase

uint8_t **mac_addr**[8]

MAC address of the participating device in the current phase struct **phControllerHusSessionConfig**

*#include <UwbApi_Types.h>* Structure for storing Controller HUS Session Configurations.

### Public Members

uint32_t **sessionHandle**

Primary Session Handle

uint8_t **phase_count**

Number of Phases

uint8_t **update_time**[8]

Update Time

phCtlrPhaseList_t **phase_list**[6]

Phase List

struct **phCleePhaseList**

*#include <UwbApi_Types.h>* Structure for storing List of Phases of Controlee.

### Public Members

uint32_t **phase_sessionHandle**

Secondary Session Handle

uint8_t **phase_participation**

Phase Participation to indicate whether the device shall participate or not in the phase

struct **phControleeHusSessionConfig**

*#include <UwbApi_Types.h>* Structure for storing Controlee HUS Session Configurations.

### Public Members

uint32_t **sessionHandle**

Primary Session Handle

uint8_t **phase_count**

Number of Phases

phCleePhaseList_t **phase_list**[6]

Phase List

struct **phDataTxPhaseMngList**

*#include <UwbApi_Types.h>* Structure to store Data Transfer Phase Management List.

### Public Members

uint8_t **mac_addr**[8]

MAC address for which Data Tx slots are configured

uint8_t **slot_bitmap**[32]

Slot Bitmap

struct **phDataTxPhaseConfig**

*#include <UwbApi_Types.h>* structure to store Data Transfer Phase Configurations.

### Public Members

uint32_t **dtpcm_SessionHandle**

Session Handle to which Data Transfer phase to be configured

uint8_t **dtpcm_Repetition**

Data Transfer Phase Control Message Repetition

uint8_t **dataTransferCtrl**

Data Transfer Control

uint8_t **dtpml_size**

Data Transfer Phase Management List Size

phDataTxPhaseMngList_t **dtpml**[6]

Data Transfer Phase Management List

struct **phDataTxPhaseCfgNtf**

*#include <UwbApi_Types.h>* Structure to store Data Transfer Phase Config notification values.

### Public Members

uint32_t **sessionHandle**

Session Handle to which the DataTx phase is configured

uint8_t **status**

Data Tx phase Status

struct **phUwbappContext**

*#include <UwbApi_Types.h>* Structure lists out the UWB-IoT init context.

### Public Members

tUwbApi_AppCallback ***pCallback**

Standalone mode callback function

tUwbApi_AppDataCallback ***pMcttCallback**

Application specific callback function

tUwbApi_AppCallback ***pCdcCallback**

CDC mode callback function

phUwbFWImageContext_t **fwImageCtx**

FW Image context

void ***seHandle**

SE communication handle Can be set to Null if SE is not present struct **phGenericError**

*#include <UwbApi_Types.h>* Structure lists out the Generic Error notification.

### Public Members

uint8_t **status**

Status

struct **phUwbSessionInfo**

*#include <UwbApi_Types.h>* Structure lists out session information.

### Public Members

uint32_t **sessionHandle**

Session Handle

uint8_t **state**

Session state

uint8_t **reason_code**

Reason code

struct **phControleeList**

*#include <UwbApi_Types.h>* Structure List out Controlee info.

### Public Members

uint16_t **short_address**

Short address

uint32_t **subsession_id**

Sub Session Handle

uint8_t **subsession_key**[32]

Controlee specific Sub-session Key 16/32 Bytes struct **phMulticastControleeListContext**

*#include <UwbApi_Types.h>* Structure for storing Multicast Controlee List Context.

### Public Members

uint8_t **action**

Action

uint8_t **no_of_controlees**

Controlee Count

phControleeList_t **controlee_list**[8]

Controlee List

struct **phMulticastControleeStatusList**

*#include <UwbApi_Types.h>* Structure for storing Multicast Controlee Status List Context.

struct **phMulticastControleeListNtfContext**

*#include <UwbApi_Types.h>* Structure for storing Multicast Controlee List Ntf Context.

struct **phBlinkDataTxNtfContext**

*#include <UwbApi_Types.h>* Structure for storing Blink Data Tx Ntf Context.

### Public Members

uint8_t **repetition_count_status**

Repetition Count Status

struct **phUwbDataPkt**

*#include <UwbApi_Types.h>* Structure for SendData.

### Public Members

uint32_t **sessionHandle**
Session Handle

uint8_t **mac_address**[8]
MAC Address

uint16_t **sequence_number**
Sequence Number

uint16_t **data_size**
Data Size

uint8_t ***data**
Application Data

struct **phUwbRcvDataPkt**
*#include <UwbApi_Types.h>* Structure for RcvData.

### Public Members

uint32_t **sessionHandle**
Session Handle

uint8_t **status**
Status

uint8_t **src_address**[8]
MAC Address

uint16_t **sequence_number**
Sequence Number

uint16_t **data_size**
Data Size

uint8_t **data**[MAX_RESPONSE_DATA_RCV]

Application Data

struct **phDataControlTransmitNtfContext**

*#include <UwbApi_Types.h>* Structure for storing Data Control Transmit Ntf Context.

### Public Members

uint32_t **sessionHandle**

Session Handle

uint8_t **status**

Status

struct **phUwbProfileInfo**

*#include <UwbApi_Types.h>* Structure lists out Profile information.

### Public Members

uint32_t **sessionHandle**

Session Handle is out param

uint8_t **mac_addr**[2]

Source mac address

eUWB_ProfileID_t **profileId**

Profile ID

uint8_t **deviceRole**

Device Role

- 0x00: Responder
- 0x01: Initiator
- 0x02: Master Anchor

uint8_t **deviceType**

Device Type, 0x00: Controlee, 0x01: Controller struct **phUwbCapInfo**

*#include <UwbApi_Types.h>* Structure lists out the UWB Device Info Parameters.

### Public Members

uint8_t **firaPhyLowerRangeMajorVersion**

FIRA phy lower range major version

uint8_t **firaPhyLowerRangeMinorMaintenanceVersion**

FIRA phy lower range minor maintenance version

uint8_t **firaPhyHigherRangeMajorVersion**

FIRA phy higher range major version

uint8_t **firaPhyHigherRangeMinorMaintenanceVersion**

FIRA phy higher range minor maintenance version

uint8_t **firaMacLowerRangeMajorVersion**

FIRA mac lower range major version

uint8_t **firaMacLowerRangeMinorMaintenanceVersion**

FIRA mac lower range minor maintenance version

uint8_t **firaMacHigherRangeMajorVersion**

FIRA mac higher range major version

uint8_t **firaMacHigherRangeMinorMaintenanceVersion**

FIRA mac higher range minor maintenance version

uint8_t **deviceTypes**

Device types

uint16_t **deviceRoles**

Device roles

uint16_t **rangingMethod**

Ranging method

uint8_t **stsConfig**
STS config

uint8_t **multiNodeMode**
Multinode mode

uint8_t **rangingTimeStruct**
Ranging time struct

uint8_t **scheduledMode**
Scheduled mode

uint8_t **hoppingMode**
Hopping mode

uint8_t **blockStriding**
Block striding

uint8_t **uwbInitiationTime**
UWB initiation time

uint8_t **channels**
FIRA phy ver range

uint8_t **rframeConfig**
RFRAME config

uint8_t **ccConstraintLength**
CC constraint length

uint8_t **bprfParameterSets**
BPRF parameter sets

uint8_t **hprfParameterSets**[5]
HPRF parameter sets

uint8_t **aoaSupport**
AOA support

uint8_t **extendedMacAddress**

Extended mac address

uint16_t **maxMessageSize**

Max message size

uint16_t **maxDataPacketPayloadSize**

Max data packet payload size

uint8_t **slotBitmask**

Slot bitmask

uint8_t **syncCodeIndexBitmask**[4]

Sync code index bitmask

uint8_t **hoppingConfigBitmask**

Hopping config bitmask

uint8_t **channelBitmask**

Channel bitmask

uint16_t **supportedProtocolVersion**

Supported protocol version

uint16_t **supportedUWBConfigID**

Supported UWB config ID

uint8_t **supportedPulseShapeCombo**[9]

Supported pulse shape combo

uint16_t **maxUciPayloadLength**

Maximum UCI payload size that can handle by UWBS

uint8_t **inbandDataBlockSize**

Data buffer block size in bytes which the UWBS manages for the overall in-band data transfer memory pool.

uint8_t **inbandDataMaxBlock**

Parameter to indicate the number of blocks available in the overall in-band data transfer memory pool.

struct **phUwbRframeLogNtf**

*#include <UwbApi_Types.h>* Structure lists out the RFRAME Log Notification Parameters.

struct **phUwbQueryDataSize**

*#include <UwbApi_Types.h>* Structure lists out the Query Data Size Parameters.

## 6.4 UWB Functional APIs

*group* **uwb_apis**

Various UWB APIs.

APIs exposed to application to access UWB Functionality.

### Defines

**UWBAPI_SETAPPCONFIGMULTIPLEPARAMS**(SESSION_HANDLE, PARMS_LIST)

Helper macro to limit the parameters and avoid error case

### Functions

*tUWBAPI_STATUS* **UwbApi_Init_New**(phUwbappContext_t *pAppCtx)

Initialize the UWB Device stack in the required mode. Operating mode will be set as per the Callback functions. Operating Modes supported include Standalone mode [default mode], CDC mode and MCTT mode. Atleast one of the call backs shall not be NULL. When all the callbacks are set then "Standalone" mode will take precedence.

**Parameters**

**pAppCtx** – **[in]** Pointer to phUwbappContext_t structure

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SwitchToMCTTMode**(phUwbappContext_t *pAppCtx)

To switch the Operating mode to MCTT.

**Parameters**

**pAppCtx** – **[in]** Pointer to phUwbappContext_t strucutre

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_Init**(tUwbApi_AppCallback *pCallback)

Initialize the UWB Middleware stack in standalone mode.

**Parameters**

**pCallback** – **[in]** Pointer to *tUwbApi_AppCallback* (Callback function to receive notifications (Ranging data/App Data/Per Tx & Rx) at application layer.)

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_ShutDown**()

De-initializes the UWB Middleware stack Sequence of task deinitialization must be maintained -> Deinit client thread -> Deinit reader thread -> Deinit uwb_task thread.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_UwbdReset**(uint8_t resetConfig)

(UWBIOT_UWBD_SR040)

Resets UWBD device to Ready State

**Parameters**

**resetConfig** – **[in]** Supported Value: UWBD_RESET

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetUwbDevState**(uint8_t *pDeviceState)

Gets UWB Device State.

**Parameters**

**pDeviceState** – **[out]** pointer to uint8_t to get Device State. Valid only if API status is success.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_INVALID_PARAM** – if parameter is invalid
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SessionInit**(uint32_t sessionId, eSessionType sessionType, uint32_t *sessionHandle)

Initializes session for a Type(Ranging/Data/Per)

**Parameters**

- **sessionId** – **[in]** Session ID.
- **sessionType** – **[in]** Type of Session(Ranging/Data/Per).
- **sessionHandle** – **[out]** Session Handle.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_MAX_SESSIONS_EXCEEDED** – if more than 5 sessions are exceeded
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SessionDeinit**(uint32_t sessionHandle)

De-initialize based on Session Handle.

**Parameters**

**sessionHandle** – **[in]** Initialized Session Handle

**Return values**

- **UWBAPI_STATUS_OK** – on success

- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetDeviceInfo**(phUwbDevInfo_t *pdevInfo)

Returns UCI, FW and MW version.

**Parameters**

**pdevInfo** – **[out]** Pointer to phUwbDevInfo_t

**Return values**

- **UWBAPI_STATUS_OK** – if successful
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UCI stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_SetRangingParams**(uint32_t sessionHandle, const phRangingParams_t *pRangingParam)

Set session specific ranging parameters.

For contention based ranging DST_MAC_ADDRESS and NO_OF_CONTROLEES parameter is not required both should be set to zero.

Example: For time based and contention based configuration given below.

```
// Time based Ranging :

phRangingParams_t inRangingParams = {0};
inRangingParams.noOfControlees = 1;
inRangingParams.dstMacAddr[] = {0x11,0x22};

// Contention based Ranging :

phRangingParams_t inRangingParams = {0};
inRangingParams.noOfControlees = 0;
inRangingParams.dstMacAddr[] = {0x00,0x00};
```

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle

- **pRangingParam** – **[in]** Pointer to phRangingParams_t

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetRangingParams**(uint32_t sessionHandle, phRangingParams_t *pRangingParams)

Get session specific ranging parameters.

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **pRangingParams** – **[out]** Pointer to phRangingParams_t .Valid only if API status is success

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetAppConfig**(uint32_t sessionHandle, eAppConfig param_id, uint32_t param_value)

Set session specific app config parameters.

Note : FOR SR1XXT and SR2XXT this API can only be used to set FIRA-specific AppCfgs.

> **Warning:** For setting STATIC_STS_IV and UWB_INITIATION_TIME, use UwbApi_SetAppConfigMultipleParams API.

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **param_id** – **[in]** App Config Parameter Id
- **param_value** – **[in]** Param value for App config param id

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetAppConfigMultipleParams**(uint32_t sessionHandle, uint8_t noOfparams, const UWB_AppParams_List_t *AppParams_List)

Host shall use this API to set multiple application configuration parameters. Number of Parameters also needs to be indicated.

To easily set the AppParams list, following macros have been defined.

*UWB_SET_APP_PARAM_VALUE(Parameter, Value)*: This macro sets the value of the corresponding parameter with the given Value.This shall be used to set all types of values of 8 or 16 or 32 bit wide.For more than 32-bit values, following macro shall be used.

*UWB_SET_APP_PARAM_ARRAY(Parameter, ArrayValue, Length)*: This macro sets the value of the corresponding parameter as an array of 8bit. Length parameter contains the total length of the array.

Example: To set SFD Id to zero and static sts iv, macro shall be invoked as given below.

```
UWB_AppParams_List_t SetAppParamsList[] = {UWB_SET_APP_PARAM_
↪VALUE(SFD_ID, 0)};
uint8_t static_sts_iv[] = {1,2,3,4,5,6};
```

(continues on next page)

(continued from previous page)

```
UWB_AppParams_List_t SetAppParamsList[] = {UWB_SET_APP_PARAM_
↪ARRAY(STATIC_STS_IV, static_sts_iv, sizeof(static_sts_iv))};
```

Note : FOR SR1XXT and SR2XXT this API can only be used to set FIRA-specific AppCfgs.

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **noOfparams** – **[in]** Number of App Config Parameters
- **AppParams_List** – **[in]** Application parameters values in tlv format

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetVendorAppConfigs**(uint32_t sessionHandle, uint8_t noOfparams, const UWB_VendorAppParams_List_t *vendorAppParams_List)

Host shall use this API to set multiple Vendor specific application configuration parameters. Number of Parameters also needs to be indicated.

To easily set the AppParams list, following macros have been defined.

UWB_SET_VENDOR_APP_PARAM_VALUE(Parameter, Value): This macro sets the value of the corresponding parameter with the given Value.This shall be used to set all types of values of 8 or 16 or 32 bit wide.For more than 32-bit values, following macro shall be used.

UWB_SET_VENDOR_APP_PARAM_ARRAY(Parameter, ArrayValue, Length): This macro sets the value of the corresponding parameter as an array of 8bit. Length parameter contains the total length of the array.

Example: To set MAC Palyoad encription Id to zero and antenna config tx, macro shall be invoked as given below.

```
UWB_VendorAppParams_List_t SetAppParamsList[] = {UWB_SET_VENDOR_
↪APP_PARAM_VALUE(MAC_PAYLOAD_ENCRYPTION, 0)};
uint8_t antennas_configuration_tx[] = {1,2,3,4,5,6};
UWB_VendorAppParams_List_t SetAppParamsList[] = {UWB_SET_VENDOR_
↪APP_PARAM_ARRAY(ANTENNAS_CONFIGURATION_TX, antennas_
↪configuration_tx, sizeof(antennas_configuration_tx))};
```

**Parameters**

- **`sessionHandle`** – **[in]** Initialized Session Handle
- **`noOfparams`** – **[in]** Number of App Config Parameters
- **`vendorAppParams_List`** – **[in]** vendor specific Application parameters values in tlv format

**Return values**

- **`UWBAPI_STATUS_OK`** – on success
- **`UWBAPI_STATUS_NOT_INITIALIZED`** – if UWB stack is not initialized
- **`UWBAPI_STATUS_INVALID_PARAM`** – if invalid parameters are passed
- **`UWBAPI_STATUS_SESSION_NOT_EXIST`** – if session is not initialized with sessionHandle
- **`UWBAPI_STATUS_TIMEOUT`** – if command is timeout
- **`UWBAPI_STATUS_FAILED`** – otherwise

*tUWBAPI_STATUS* **`UwbApi_GetAppConfig`**(uint32_t sessionHandle, eAppConfig param_id, uint32_t *param_value)

Get session specific app config parameters.

Note : FOR SR1XXT and SR2XXT this API can only be used to get FIRA-specific AppCfgs.

**Parameters**

- **`sessionHandle`** – **[in]** Initialized Session Handle
- **`param_id`** – **[in]** App Config Parameter Id
- **`param_value`** – **[out]** Param value for App config param id

**Return values**

- **`UWBAPI_STATUS_OK`** – on success
- **`UWBAPI_STATUS_NOT_INITIALIZED`** – if UWB stack is not initialized

- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetAppConfigMultipleParams**(uint32_t sessionHandle, uint8_t noOfparams, UWB_AppParams_List_t *AppParams_List)

Host shall use this API to get multiple application configuration parameters. Number of Parameters also needs to be indicated.

Following macros can be used, to easily set the AppParams list

*UWB_SET_GETAPP_PARAM(Parameter)*: This macro sets parameter, in UWB_AppParams_List_t structure. This shall be used to get all types of values of 8 or 16 or 32 bit wide. For more than 32-bit values, following macro shall be used.

*UWB_SET_APP_PARAM_ARRAY(Parameter, ArrayValue, Length)*: This macro sets parameter and array of 8bit to store the configuration, in UWB_AppParams_List_t structure Length parameter contains the total length of the array.

Example: To get SFD Id and static sts iv, macro shall be invoked as given below.

```
uint8_t static_sts_iv[6];
UWB_AppParams_List_t SetAppParamsList[] = {UWB_SET_GETAPP_PARAM_
↪VALUE(SFD_ID),
                                          UWB_SET_APP_PARAM_
↪ARRAY(STATIC_STS_IV, static_sts_iv, sizeof(static_sts_iv)),};
```

Note : FOR SR1XXT and SR2XXT this API can only be used to get FIRA-specific AppCfgs.

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **noOfparams** – **[in]** Number of App Config Parameters
- **AppParams_List** – **[in]** Application parameters

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized

- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetVendorAppConfigs**(uint32_t sessionHandle, uint8_t noOfparams, UWB_VendorAppParams_List_t *vendorAppParams_List)

Host shall use this API to get multiple vendor application configuration parameters. Number of Parameters also needs to be indicated.

Following macros can be used, to easily set the AppParams list

UWB_SET_GETVENDOR_APP_PARAM_VALUE(Parameter): This macro sets parameter, in UWB_VendorAppParams_List_t structure. This shall be used to get all types of values of 8 or 16 or 32 bit wide. For more than 32-bit values, following macro shall be used.

UWB_VENDOR_SET_APP_PARAM_ARRAY(Parameter, ArrayValue, Length): This macro sets parameter and array of 8bit to store the configuration, in UWB_VendorAppParams_List_t structure Length parameter contains the total length of the array.

Example: To get MAC Palyoad encription Id and antenna config tx, macro shall be invoked as given below.

```
uint8_t antennas_configuration_tx[6];
UWB_VendorAppParams_List_t SetAppParamsList[] = {UWB_SET_
↪GETVENDOR_APP_PARAM_VALUE(MAC_PAYLOAD_ENCRYPTION),
                                                     UWB_VENDOR_SET_APP_PARAM_
↪ARRAY(ANTENNAS_CONFIGURATION_TX, antennas_configuration_tx,␣
↪sizeof(antennas_configuration_tx)),};
```

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **noOfparams** – **[in]** Number of App Config Parameters
- **vendorAppParams_List** – **[in]** Vendor Application parameters

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized

- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetStaticSts**(uint32_t sessionHandle, uint16_t vendorId, uint8_t const *const staticStsIv)

Sets session specific app config parameters Vendor ID and Static STS IV.

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **vendorId** – **[in]** App Config Parameter Vendor Id
- **staticStsIv** – **[in]** Param value for App config param static Sts Iv It is the responsibility of the caller that STS IV is exactly UCI_PARAM_LEN_STATIC_STS_IV bytes long.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_StartRangingSession**(uint32_t sessionHandle)

Start Ranging for a session. Before Invoking Start ranging its mandatory to set all the ranging configurations.

**Parameters**

**sessionHandle** – **[in]** Initialized Session Handle

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetDeviceConfig**(eDeviceConfig param_id, uint8_t param_len, phDeviceConfigData_t *param_value)

Sets device configuration.

**Parameters**

- **param_id** – **[in]** device configuration param id
- **param_len** – **[in]** Parameter length
- **param_value** – **[in]** Param value

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetDeviceConfig**(eDeviceConfig param_id, phDeviceConfigData_t *param_value)

Get device config parameters.

**Parameters**

- **param_id** – **[in]** Device Config Parameter Id
- **param_value** – **[inout]** Param value structure for device config param id

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_StopRangingSession**(uint32_t sessionHandle)

Stop Ranging for a session.

**Parameters**

**sessionHandle** – **[in]** Initialized Session Handle

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_EnableRangingDataNtf**(uint32_t sessionHandle, uint8_t enableRangingDataNtf, uint16_t proximityNear, uint16_t proximityFar)

Enable Ranging Data Notifications different options.

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **enableRangingDataNtf** – **[in]** Enable Ranging data notification 0/1/2. option 2 is not allowed when ranging is ongoing.
- **proximityNear** – **[in]** Proximity Near value valid if enableRangingDataNtf sets to 2
- **proximityFar** – **[in]** Proximity far value valid if enableRangingDataNtf sets to 2

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetSessionState**(uint32_t sessionHandle, uint8_t *sessionState)

Get Session State.

if API returns *UWBAPI_STATUS_OK*, Session State would be one of the below values UWBAPI_SESSION_INIT_SUCCESS - Session is Initialized *UWBAPI_SESSION_DEINIT_SUCCESS* - Session is De-initialized *UWBAPI_SESSION_ACTIVATED* - Session is Busy *UWBAPI_SESSION_IDLE* - Session is Idle *UWBAPI_SESSION_ERROR* - Session Not Found if API returns not *UWBAPI_STATUS_OK*, Session State is set to *UWBAPI_SESSION_ERROR*

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **sessionState** – **[out]** Session Status

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UCI stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SendRawCommand**(uint8_t data[], uint16_t data_len, uint8_t *pResp, uint16_t *pRespLen)

Send UCI RAW command.

**Parameters**

- **data** – **[in]** UCI Command to be sent
- **data_len** – **[in]** Length of the UCI Command
- **pResp** – **[out]** Response Received
- **pRespLen** – **[out]** Response length

**Return values**

- **UWBAPI_STATUS_OK** – if successful
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UCI stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if wrong parameter is passed
- **UWBAPI_STATUS_BUFFER_OVERFLOW** – if response buffer is not sufficient to hold the response
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_UpdateControllerMulticastList**(phMulticastControleeListContext_t *pControleeContext)

Update Controller Multicast List.

**Parameters**

**pControleeContext** – **[in]** Controlee Context

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetTrng**(uint8_t trng_size, uint8_t *ptrng)

Get TRNG api.

> **Warning:** On SR040, maximum 4 bytes can be drawn during an active ranging session

**Parameters**

- **trng_size** – **[in]** Size of ptrng buffer and number of bytes expected
- **ptrng** – **[out]** : the size of this buffer shall be equal to trng size.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetProfileParams**(uint8_t *pProfileBlob, uint16_t blobSize, phUwbProfileInfo_t *pProfileInfo)

Setting up Profile blob.

**Parameters**

- **pProfileBlob** – **[in]** : Profile Blob which contain all information.
- **blobSize** – **[in]** : Size of Blob
- **pProfileInfo** – **[inout]** : contains profile information.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized

- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetUwbConfigData_iOS**(*UWB_DeviceRole_t* device_role, AccessoryUwbConfigDataContent_t *uwb_data_content)

Fill Accessory UWB related configuration data.

**Parameters**

- **device_role** – -[in] device role
- **uwb_data_content** – -[Out] Pointer to structure of AccessoryUwbConfigDataContent_t

**Return values**

- **UWBAPI_STATUS_OK** – - on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – - if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – - if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – - otherwise

*tUWBAPI_STATUS* **UwbApi_GetUwbConfigData_Android**(UwbDeviceConfigData_t *uwb_data_content)

Fill Accessory UWB related configuration data.

**Parameters**

**uwb_data_content** – **[out]** : Pointer to structure of UwbDeviceConfigData_t

**Return values**

- **UWBAPI_STATUS_OK** – - on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – - if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – - if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – - otherwise

*tUWBAPI_STATUS* **UwbApi_GetDeviceCapability**(phUwbCapInfo_t *pDevCap)

Frames the device capabilities in TLV format.

**Parameters**

**pDevCap** – - [Out] Pointer to structure of device capability data

**Return values**

- **UWBAPI_STATUS_OK** – - on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – - if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – - if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – - otherwise

*tUWBAPI_STATUS* **UwbApi_SendData**(phUwbDataPkt_t *pSendData)

Host shall use this API to send data over UWB interface. If SESSION_DATA_TRANSFER_STATUS_NTF is disabled, then the UWBS shall not send SESSION_DATA_TRANSFER_STATUS_NTF for every Application Data transmission except for last.

EX: SESSION_DATA_TRANSFER_STATUS_NTF=0(Disable) and DATA_REPETITION_COUNT=5 and RANGING_ROUND_USAGE=200 ms

UWA_DM_DATA_TRANSMIT_STATUS_EVT will receive after 1,000 ms (DATA_REPETITION_COUNT * RANGING_ROUND_USAGE )

Limitation:

Notification Read Timeout : Reading UWA_DM_DATA_TRANSMIT_STATUS_EVT Notfication from the UWB will result in a read timeout if UWB Notification time exceeds the define limit.

EX: if DATA_REPETITION_COUNT=60 and SESSION_DATA_TRANSFER_STATUS_NTF=0 and RANGING_ROUND_USAGE=200 ms then UWA_DM_DATA_TRANSMIT_STATUS_EVT will receive after ~12 secs (DATA_REPETITION_COUNT * RANGING_ROUND_USAGE) by that time UwbApi_SendData API time out will happen .

> **Warning:** : There is a possibility of UwbApi_SendData API returning Timeout (UWBAPI_STATUS_FAILED status) in the following sceanrio. Although API status is failed but the outcome of testing scenario to be treated as SUCCESS only.

**Parameters**

**phUwbDataPkt_t** – **[in]** Send Data Content

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized

- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_REJECTED** – if session is not established when data packet sent
- **UWBAPI_STATUS_NO_CREDIT_AVAILABLE** – if buffer is not available to accept data
- **UWBAPI_STATUS_DATA_TRANSFER_ERROR** – if data is not sent due to an unrecoverable error
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetControllerHusSession**(phControllerHusSessionConfig_t *pHusSessionCfg)

Frames the HUS session config in TLV format for Controller.

**Parameters**

**pHusSessionCfg** – **[in]** : Pointer to structure of device HUS Controller session config data.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if Session is not existing or not created.
- **UWBAPI_STATUS_INVALID_PHASE_PARTICIPATION** – if Invalid Phase Participation values in HUS CONFIG CMD.
- **UWBAPI_STATUS_SESSION_NOT_CONFIGURED** – if Session is not configured with required app configurations.
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetControleeHusSession**(phControleeHusSessionConfig_t *pHusSessionCfg)

Frames the HUS session config in TLV format for Controlee.

**Parameters**

**pHusSessionCfg** – **[in]** : Pointer to structure of device HUS Controlee session config data.

**Return values**

- **UWBAPI_STATUS_OK** – on success

- **UWBAPI_STATUS_REJECTED** – if UWB stack is not initialized
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if Session is not existing or not created.
- **UWBAPI_STATUS_INVALID_PHASE_PARTICIPATION** – if Invalid Phase Participation values in HUS CONFIG CMD.
- **UWBAPI_STATUS_SESSION_NOT_CONFIGURED** – if command is time-out
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SessionDataTxPhaseConfig**(phDataTxPhaseConfig_t *phDataTxPhaseCfg)

Frames the Data Transfer Phase Control Message in TLV format.

**Parameters**

**phDataTxPhaseCfg** – - Pointer to structure of Data Transfer Phase Configuration.

**Return values**

- **UWBAPI_STATUS_OK_DTPCM_CONFIG_SUCCESS** – - if DTPCM is configured for given MAC Address.
- **UWBAPI_STATUS_ERROR_INVALID_SLOT_BITMAP** – - if configured slot bit map size exceeds RANGING_DURATION.
- **UWBAPI_STATUS_ERROR_DUPLICATE_SLOT_ASSIGMENT** – - if configured slot assignments is inconsistent.
- **UWBAPI_STATUS_ERROR_INVALID_MAC_ADDRESS** – - if DTPCM is configured for given MAC Address.
- **UWBAPI_STATUS_INVALID_PARAM** – - if given MAC address is not found.
- **UWBAPI_STATUS_FAILED** – - otherwise

## 6.5 UWB Functional APIs (SR100/SR150 Specific)

*group* **uwb_apis_sr1xx**

Various SR100/SR150 Specific APIs.

APIs for SR100 and SR150

### Defines

**UWBD_DPD_TIMEOUT_MIN**
Minimum timeout Supported by UWBS

**UWBD_DPD_TIMEOUT_MAX**
Maximum timeout Supported by UWBS

**UWBD_VERSION_LENGTH_MAX**
Maximum length of version Supported by UWBS

**UWB_TAG_CMAC_LENGTH**

**UWBS_MAX_UCI_PACKET_SIZE**
UWBS MAX UCI packet size

**HOST_MAX_UCI_PACKET_SIZE**
HOST MAX UCI packet size

**MIN_DEVICE_PACKET_SIZE**
MIN DEVICE packet size

**EXTENDED_PARAM_ID_MASK**
EXTENDED PARAM ID MASK

**CCC_INFO_ID**
CCC PARAM ID

**MAX_UCI_CCC_VERSION_LEN**
MAX UCI CCC Version Length

**MAX_CCC_VERSION_LEN**
MAX CCC Version Length

**MODULE_MAKER_ID_MAX_SIZE**
MODULE MAKER ID MAX SIZE

**MODULE_MAKER_ID_MAX_SIZE_FW**

**MAX_UWB_CHIP_ID_LEN**
MAX UWB CHID ID Length

**MAX_PPM_VALUE_LEN**
MAX UWBS PPM VALUE Length

**MAX_TX_POWER_LEN**
MAX TX Power Lenght

**FW_BOOT_MODE_LEN**
MAX FW BOOT Length

**RANDOM_KEY_LEN**

**SESSION_KEY_LEN**

**WRAPPED_RDS_LEN**

**MAX_RDS_LIST_SIZE**
Max RDS List Size

**VCO_PLL_POS**

**TX_POWER_POS**

**XTAL_CAP_VALUES_POS**

**RSSI_CONSTANT1_POS**

**RSSI_CONSTANT2_POS**

**TX_POWER_PARAMS_POS**

**PAPPPA_CALIB_CTRL_POS**

**TX_TEMP_COMP_POS**

`DELAY_CALIB_POS`

`PDOA_MFG_OFFSET_POS`

`PDOA_AOA_ANT_MULTIPOINT_CALIB`

`UWB_ANTENNA_SELECTION_GPIO_BIT_EF1_MASK`

antenna selection gpio bit mask

Antenna selection gpio bit mask for EF1

`UWB_ANTENNA_SELECTION_GPIO_BIT_EF2_MASK`

Antenna selection gpio bit mask for EF2

`UWB_ANTENNA_SELECTION_GPIO_BIT_GPIO6_MASK`

Antenna selection gpio bit mask for GPIO6

`UWB_ANTENNA_SELECTION_GPIO_BIT_GPIO7_MASK`

Antenna selection gpio bit mask for GPIO7

`UWB_ANTENNA_SELECTION_GPIO_BIT_GPIO9_MASK`

Antenna selection gpio bit mask for GPIO9

`UWB_ANTENNA_SELECTION_GPIO_BIT_GPIO10_MASK`

Antenna selection gpio bit mask for GPIO10

`UWB_ANTENNA_SELECTION_GPIO_BIT_GPIO11_MASK`

Antenna selection gpio bit mask for GPIO11

`UWB_ANTENNA_SELECTION_GPIO_BIT_GPIO14_MASK`

Antenna selection gpio bit mask for GPIO14

`MAX_RFRAME_MEAS`

MAX RFRAME Measurements

`MAX_RFRAME_PACKET_SIZE`

MAX RFRAME packet size

**RANGE_DATA_NTF_BOUND_AOA**

**RNG_DATA_NTF_PROXIMITY_FAR**

**RNG_DATA_NTF_PROXIMITY_NEAR**

**RNG_DATA_NTF**

**kAPPPARAMS_Type_Unknown**

**kAPPPARAMS_Type_u32**

**kAPPPARAMS_Type_au8**

**UWB_SET_APP_PARAM_VALUE**(PARAM, VALUE)

Macro to set SetApp Configuration parameters value supported in UWB API layer.

**UWB_SET_GETAPP_PARAM**(PARAM)

Macro to set GetApp Configuration parameters value supported in UWB API layer.

**UWB_SET_APP_PARAM_ARRAY**(PARAM, ARRAY, LENGTH)

Macro to set SetApp/GetApp Configuration parameters array value supported in UWB API layer.

**UWB_SET_VENDOR_APP_PARAM_VALUE**

**UWB_SET_GETVENDOR_APP_PARAM**

**UWB_SET_VENDOR_APP_PARAM_ARRAY**

**UWB_SET_DEBUG_PARAM_VALUE_u8**(PARAM, VALUE)

Macro to set Set Debug Configuration parameters value supported in UWB API layer.

**UWB_SET_DEBUG_PARAM_VALUE_u16**(PARAM, VALUE)

**UWB_SET_DEBUG_PARAM_VALUE_u32**(PARAM, VALUE)

**UWB_SET_GETDEBUG_PARAM_u8**(PARAM)

Macro to set Get Debug Configuration parameters value supported in UWB API layer.

**UWB_SET_GETDEBUG_PARAM_u16**(PARAM)

**UWB_SET_GETDEBUG_PARAM_u32**(PARAM)

**AOA_ANTENNAE_PDOA_CALIB**

**UwbApi_DoCalibration**(channel, paramId, calibResp)

### Typedefs

typedef UWB_AppParams_type_t **SetAppParams_type_t**

typedef UWB_AppParams_value_au8_t **SetAppParams_value_au8_t**

typedef UWB_AppParams_value_t **SetAppParams_value_t**

typedef UWB_AppParams_List_t **SetAppParams_List_t**

### Enums

enum **otpRWOption**

OTP Read Write Options.

*Values:*

enumerator **CALIB_PARAM**

enum **appConfig**

SetApp Configuration parameters supported in UWB API layer.

*Values:*

enumerator **RANGING_ROUND_USAGE**

Ranging Method

```
- 0:TDoA
- 1:SS-TWR with Deferred Mode
- 2:DS-TWR with Deferred Mode
- 3:SS-TWR with Non-Deferred Mode
- 4:DS-TWR with Non-Deferred Mode
- 6:TDoA with Advertisement
```

enumerator **STS_CONFIG**

STS Config
- 0:Static STS
- 1:Dynamic STS
- 2:Dynamic STS for controlee individual key
- 3:Provisioned STS
- 4:Provisioned STS for Responder specific Sub-session Key
- 5 to 0xFF : RFU

enumerator **CHANNEL_NUMBER**

5,6,8,9

enumerator **NO_OF_CONTROLEES**

Number of Controlees

enumerator **DST_MAC_ADDRESS**

Destination MAC Addr

enumerator **SLOT_DURATION**

Slot duration in RSTU (Ranging/Radar Standard Time Unit)

enumerator **RANGING_DURATION**

Ranging Duration in ms

enumerator **STS_INDEX**

STS index init value

enumerator **MAC_FCS_TYPE**

0:CRC16, 1:CRC32

enumerator **RANGING_ROUND_CONTROL**

- 1:Enable,
- 0:Disable
- b0 - Measurement Report Phase,
- b1 - Ranging Control Phase
- b2 : b7 - RFU (default = 0x03)

enumerator **AOA_RESULT_REQ**

0: AOA Disable, 1: AOA Enable

enumerator **SESSION_INFO_NTF**

0:Disable, 1:Enable, 2:Enable in proximity range

enumerator **NEAR_PROXIMITY_CONFIG**

Proximity near in cm

enumerator **FAR_PROXIMITY_CONFIG**

Proximity far in cm

enumerator **DEVICE_ROLE**

0x00 = Responder 0x01 = Initiator 0x02 = UT-Synchronization Anchor 0x03 = UT-Anchor 0x04 = UT-Tag Values 0x05 to 0xFF = RFU

enumerator **RFRAME_CONFIG**

0:No STS, 1:STS follows SFD, 2:STS follows PSDU, 3:STS follows SFD

enumerator **PREAMBLE_CODE_INDEX**

[9-12]:BPRF, [25-32]:HPRF

enumerator **SFD_ID**

[0,2]:BPRF, [1,3]:HPRF

enumerator **PSDU_DATA_RATE**

0:6.81MBPS, 1:7.80MBPS

enumerator **PREAMBLE_DURATION**

0:32 symbols, 1:64 symbols

enumerator **LINK_LAYER_MODE**

0x00 – Bypass Mode (Default), 0x01 – Connection-Less data transfer

enumerator **DATA_REPETITION_COUNT**

0x00 – No repetition (Default), 0xFF – Repeat infinite number of times

enumerator **RANGING_TIME_STRUCT**

1:Block based & 0:[2-255]:RFU

enumerator **SLOTS_PER_RR**

Slot per Ranging Round, Not applicable for Contention Based Ranging

enumerator **TX_ADAPTIVE_PAYLOAD_POWER**

0:disable, 1:Enable

enumerator **PRF_MODE**

0: BPRF, 1:HPRF

enumerator **KEY_ROTATION**

1: Enable, 0:Disable [Default]

enumerator **SESSION_PRIORITY**

Session Priority 1-100, default : 50

enumerator **MAC_ADDRESS_MODE**

MAC address mode Default 0 [SHORT]

enumerator **VENDOR_ID**

Vendor ID

enumerator **STATIC_STS_IV**

Static STS IV

enumerator **NUMBER_OF_STS_SEGMENTS**

Number of STS segments in the frame. 0x00:No STS Segments (if RFRAME_CONFIG is 0).

If RFRAME_CONFIG Config is set to 1 or 3 then

- 0x01:1 STS Segment(default)
- 0x02:2 STS Segments

enumerator **MAX_RR_RETRY**

Maximum Ranging Round Retry

enumerator **HOPPING_MODE**

Ranging round hopping, 1=Enable, 0=Disable [Default] enumerator **RESULT_REPORT_CONFIG**

Config to enable result report, 0: Disable, This is applicable only RANGING_ROUND_CONTROL enabled

enumerator **MAX_NUMBER_OF_MEASUREMENTS**

Maximum Number of ranging blocks to executed in a session Default : 0

enumerator **UL_TDOA_TX_INTERVAL**

This parameter specifies the average transmission interval of Blink UTMs from UT-Tags and/or Synchronization UTMs from UT-Synchronization Anchors, as defined by the UL-TDoA TX Interval MAC configuration parameter.

By default, UL_TDOA_TX_INTERVAL = 2000ms.

enumerator **UL_TDOA_RANDOM_WINDOW**

Length of the randomization window within which Blink and Synchronization UTMs may be transmitted.

enumerator **STS_LENGTH**

STS length 0: 32 symbols , 1: 64 symbols(Default), 2: 128 symbols

enumerator **UL_TDOA_DEVICE_ID**

This value shall be used to specify the length and presence of the UL-TDoA Device ID in UTMs.

enumerator **UL_TDOA_TX_TIMESTAMP**

Presence and length of TX timestamps in UTMs. 0x00: TX timestamp shall not be included (default). 0x01: 40-bit TX timestamp shall be included. 0x02: 64-bit TX timestamp shall be included. Values 0x03 to 0xFF = RFU

By default, TX timestamps shall not be included.

enumerator **SESSION_KEY**

Session Key provided for Provisioned STS mode (STS_CONFIG equal to 0x03 or 0x04) If the Session Key is not provided by the Host in Provisioned STS mode, the UWBS shall fetch the Session Key from the Secure Component. This parameter is valid only in Provisioned STS mode and shall be ignored otherwise.

enumerator **SUB_SESSION_KEY**

Sub-session Key provided for Provisioned STS for Responder specific Key mode (STS_CONFIG equal to 0x04). If the Sub-session Key is provided by the Host, the

Host shall also provide the SESSION_KEY. If the Sub-session Key is not provided by the Host for Provisioned STS for Responder specific Key mode, the UWBS shall fetch the Sub-session Key from the Secure Component. This parameter is valid only in Provisioned STS for Responder specific Key mode and shall be ignored otherwise.

enumerator **END_OF_SUPPORTED_APP_CONFIGS**

enumerator **ADAPTIVE_HOPPING_THRESHOLD**

This parameter can be used to configure the required number of successful responses(T) from Responders to conclude a successful ranging round. Default : NUMBER_OF_CONTROLEES

enumerator **MAC_CFG**

b0: MAC header present, b1:MAC footer present

enumerator **ANTENNA_CONFIG**

Antenna configiguration for Antenna selection, 0: Default. ANT_DEFAULT selected. No GPIO Toggle required. Legacy antenna selection. 1: ANT-TOP selected for Tx/Rx . GPIO set to output, value= 0. 2: ANT-BOTTOM selected for Tx/Rx. GPIO set to output, value =1.

enumerator **ULTDOA_MAC_FRAME_FORMAT**

Parameter to select MAC frame format for UL-TDOA Tag device 0x00: FIRA(default) 0x01: Vendor-specific MAC format Note: This parameter is only applicable when RANGING_ROUND_USAGE = 0x00 (One Way Ranging UL-TDoA) and DEVICE_ROLE = 0x04 (UT-Tag)

enumerator **RANGING_START_OFFSET**

Parameter to configure the time offset of first ranging block (in ms) 0..200 ms default: 0 ms (No offset)

enumerator **RX_START_MARGIN**

receiver window start margin

enumerator **RX_TIMEOUT**

receiver window length

enumerator **SR040_NBIC_CONF**

PHSCA_UCIMSG_SESSION_NBIC_CONF_ID - 0xF2u

NBIC disabled - 0u, NBIC enabled - 1u, NBIC enabled FS mode enabled - 2,

enumerator **SALTED_HASH**

enumerator **D_URSK**

enumerator **TX_POWER_ID**

TX Power ID

- 0: max power (14db)
- 104: least power(-12db)

Range 0 to 104: 0.25db per step

enumerator **DEBUG_LOG_LEVEL**

enumerator **RX_PHY_LOGGING_ENABLE**

Phy logging for Test Receive Mode, 1=Enable, 0=Disable [Default]

enumerator **TX_PHY_LOGGING_ENABLE**

Phy logging for Test Transmit Mode, 1=Enable, 0=Disable [Default]

enumerator **LOG_PARAMS_CONF**

enumerator **STS_INDEX_RESTART**

Sts Index Restart

enumerator **RX_RADIO_CFG_IDXS**

RX radio configuration slot index in flash

- Low byte indicates slot index for SP0 type
- High byte indicates slot index for SP3 type Minimum slot index: 0x10, Maximum slot index: 0x1F Default: 0x0100 [SP3(High Byte): index 0x01, SP0(Low Byte): index 0x00]

enumerator **TX_RADIO_CFG_IDXS**

TX radio configuration slot index in flash

- Low byte indicates slot index for SP0 type
- High byte indicates slot index for SP3 type Minimum slot index: 0x10, Maximum slot index: 0x1F Default: 0x1110 [SP3(High Byte): index 0x11, SP0(Low Byte): index 0x10]

enumerator **SR040_INTERNAL_0xFD**

enumerator **END_OF_SUPPORTED_EXT_CONFIGS**

enumerator **UWB_DEVICE_TYPE**

Devce Type 0x00 = Controlee 0x01 = Controller

enumerator **RANGING_ROUND_USAGE**

Ranging Round Usage 0x00 = One Way Ranging UL-TDoA 0x01 = SS-TWR with Deferred Mode 0x02 = DS-TWR with Deferred Mode 0x03 = SS-TWR with Non-deferred Mode 0x04 = DS-TWR with Non-deferred Mode 0x05 = One Way Ranging DL-TDOA 0x06 = OWR for AoA Measurement 0x07 = eSS-TWR with Non-deferred Mode for Contention-based ranging 0x08 = aDS-TWR for Contention-based ranging 0x09 = Data Transfer Mode

enumerator **STS_CONFIG**

STS Config 0x00:Static STS 0x01:Dynamic STS 0x02:Dynamic STS for controlee individual key 0x03:Provisioned STS 0x04:Provisioned STS for Responder specific Sub-session Key 0xA0:To be set at Anchor and User device to distinguish the transition from Static STS to Dynamic STS 0x05 to 0xFF except 0xA0 : RFU

enumerator **MULTI_NODE_MODE**

0x00 = Single device to Single device (Unicast) 0x01 = One to Many 0x02 = Many to Many Values 0x03 to 0xFF = RFU

enumerator **CHANNEL_NUMBER**

Possible values are {5, 6, 8, 9, 10, 12, 13, 14} (default = 9)

enumerator **NO_OF_CONTROLEES**

Number of Controlees To be configured by Host when MULTI_NODE_MODE is set other than 0x00. Number of Controlees(N) 1<=N<=8 (Default is 1)

enumerator **DEV_MAC_ADDRESS**

Device MAC Address MAC Address of the UWBS itself participating in UWB session. Size of this config is based on the MAC_ADDRESS_MODE.

Note : In case of Extended MAC Addr mode, this config is to be set through UwbApi_SetAppConfigMultipleParams.

enumerator **DST_MAC_ADDRESS**

Destination MAC Addr MAC Address of the UWBS itself participating in UWB session. Size of this config is based on the MAC_ADDRESS_MODE.

Note : In case of Extended MAC Addr mode, this config is to be set through UwbApi_SetAppConfigMultipleParams.

enumerator **SLOT_DURATION**

Unsigned integer that specifies duration of a ranging slot in the unit of RSTU (Ranging/Radar Standard Time Unit) (default = 2400)

enumerator **RANGING_DURATION**

Ranging Duration in ms Ranging duration in the unit of 1200 RSTU which is 1 ms. (default = 200)

enumerator **STS_INDEX**

STS index init value

enumerator **MAC_FCS_TYPE**

MAC FCS TYPE 0x00 = CRC 16 (default) 0x01 = CRC 32

enumerator **RANGING_ROUND_CONTROL**

1:Enable, 0:Disable Below bits are applicable when SCHEDULED_MODE is set to 0x01(Time scheduled ranging) b0 - Measurement Report Phase b1 - Control Phase b2 - Configuration of RCP in Non-deferred Mode TWR b3 : b5 - RFU b6 - Measurement Report Phase (MRP) [UWBS shall ignore this bit] b7 - Measurement Report Message (MRM) (default = 0x03) Below bits are applicable when SCHEDULED_MODE is set to 0x00(Contention-based ranging) b0 - Ranging Result Report Message (RRRM) UWBS shall ignore this bit b1 - 1 (Controller shall send a CM in-band and a Controlee shall expect a CM in-band) b2 - 1 (RCP is excluded in Ranging Round) b5 : b3 = RFU b6 - Measurement Report Phase (MRP) ; If set to 0, MRP is not present (default) ; If set to 1, MRP is present b7 - Measurement Report Message (MRM) UWBS shall ignore this bit. (default = 0x06)

enumerator **AOA_RESULT_REQ**

AOA RESULT REQ 0x00 = AoA results are disabled. 0x01 = AoA results are enabled(default), return all the AOA type supported by the device 0x02 = Only AoA Azimuth is enabled 0x03 = Only AOA Elevation is enabled

enumerator **SESSION_INFO_NTF**

SESSION_INFO_NTF 0x00 = Disable range data SESSION_INFO_NTF 0x01 =

Enable range data SESSION_INFO_NTF (default) 0x02 = Enable range data SESSION_INFO_NTF while inside proximity range 0x03 = Enable range data SESSION_INFO_NTF while inside AoA (upper and lower) bounds 0x04 = Enable range data SESSION_INFO_NTF while inside AoA bounds as well as inside proximity range 0x05 = Enable range data SESSION_INFO_NTF only when entering and leaving proximity range. 0x06 = Enable range data SESSION_INFO_NTF only when entering and leaving AoA (upper and lower) bound 0x07 = Enable range data SESSION_INFO_NTF only when entering and leaving AoA bounds as well as entering and leaving proximity range.

enumerator **NEAR_PROXIMITY_CONFIG**

Proximity near in cm (default = 0)

enumerator **FAR_PROXIMITY_CONFIG**

Proximity far in cm (default = 20000)

enumerator **DEVICE_ROLE**

Devive Role 0x00 = Responder 0x01 = Initiator 0x02 = Assigned 0x03 = Assigned 0x04 = Assigned 0x05 = Advertiser 0x06 = Observer 0x07 = DT-Anchor 0x08 = DT-Tag

enumerator **RFRAME_CONFIG**

Activate or deactivate RSSI reporting 0x00 = SP0 (Reserved value for test purpose) 0x01 = SP1 0x02 = RFU 0x03 = SP3 Values 0x04 to 0xFF = RFU (default = 0x03)

enumerator **RSSI_REPORTING**

Activate or deactivate RSSI reporting 0x00 : Disable(default) 0x01 : Enable 0x02-0xFF :RFU

enumerator **PREAMBLE_CODE_INDEX**

Preamble Code Index [9-12] - BPRF, [25-32] - HPRF, used for RADAR_MODE 1, 2 and 3 [95-102] - LPRF used for RADAR_MODE 5 and6 (default = 10)

enumerator **SFD_ID**

[0,2]:BPRF, [1,2,3,4]:HPRF (default = 2)

enumerator **PSDU_DATA_RATE**

PSDU DATA RATE 0x00 = 6.81 Mbps (default) 0x01 = 7.80 Mbps 0x02 = 27.2 Mbps 0x03 = 31.2 Mbps 0x04 = 850 Kbps Values 0x00, 0x02, 0x04 map to K=3 and 0x01, 0x03 map to K=7.

enumerator **PREAMBLE_DURATION**

0:32 symbols, 1:64 symbols, 0xA0:1024 symbols

enumerator **LINK_LAYER_MODE**

0x00 – Bypass Mode (Default), 0x01 – Connection-Less data transfer

enumerator **DATA_REPETITION_COUNT**

0x00 – No repetition (Default), 0xFF – Repeat infinite number of times

enumerator **RANGING_TIME_STRUCT**

1:Block based (default) & 0:[2-255]:RFU

enumerator **SLOTS_PER_RR**

Number of slots for per ranging round (default = 25)

enumerator **AOA_BOUND_CONFIG**

AOA_BOUND_CONFIG 1:0 AOA_BOUND_CONFIG_LOWER_BOUND_AOA_AZIMUTH range -180 (default) to 180 3:2 AOA_BOUND_CONFIG_UPPER_BOUND_AOA_AZIMUTH range -180 to 180 (default) 5:4 AOA_BOUND_CONFIG_LOWER_BOUND_AOA_ELEVATION -90 (default) to 90 7:6 AOA_BOUND_CONFIG_UPPER_BOUND_AOA_ELEVATION -90 to 90 (default)

enumerator **PRF_MODE**

PRF MODE 0x00 = 62.4 MHz PRF. BPRF mode (default) 0x01 = 124.8 MHz PRF. HPRF mode. 0x02 = 249.6 MHz PRF. HPRF mode with data rate 27.2 and 31.2 Mbps

enumerator **CAP_SIZE_RANGE**

configuration parameter sets the minimum and maximum CAP size to be used by the Controller/Initiator in the Contention-based ranging session, in the units of Ranging Slots.

Octet [0] - represents the maximum CAP size. Default = SLOTS_PER_RR-1. Octet [1] - represents the minimum CAP size. Default = 5.

enumerator **SCHEDULED_MODE**

Parameter is used to set the multi-node Ranging Type. 0x00 = Contention-based ranging 0x01 = Time scheduled ranging (default) 0x02 = Hybrid-based ranging Values 0x03 to 0xFF = RFU enumerator **KEY_ROTATION**

1: Enable, 0:Disable [Default]

enumerator **KEY_ROTATION_RATE**

Key Rotation rate 2^n where 0<=n<=15

enumerator **SESSION_PRIORITY**

Session Priority 1-100, default : 50

enumerator **MAC_ADDRESS_MODE**

MAC address mode Default 0 [SHORT]

enumerator **VENDOR_ID**

Unique vendor Id

enumerator **STATIC_STS_IV**

Vendor defined static sts

enumerator **NUMBER_OF_STS_SEGMENTS**

Number of STS segments in the frame. 0x00:No STS Segments(if PPDU_COFIG is 0). If PPDU Config is set to 1 or 3 then 0x01:1 STS Segment(default), 0x02:2 STS Segments

enumerator **MAX_RR_RETRY**

Number of Failed Ranging Round attempts before terminating the session. Default : 0

enumerator **UWB_INITIATION_TIME**

Indicates when ranging operation shall start after ranging start request is issued from AP. Default : 0

enumerator **HOPPING_MODE**

Modes for the hopping. 0x00: No hopping 0x02: Adaptive hopping using MODULO 0x03: continuous hopping using MODULO 0x04: adaptive hopping using AES 0x05: continuous hopping using AES Default : 0

enumerator **BLOCK_STRIDE_LENGTH**

Block Stride Length. 0x00:Default, [0x01-0xFF]:Application use case specific value enumerator **RESULT_REPORT_CONFIG**

Config to enable result report, 0: Disable, This is applicable only RANGING_ROUND_CONTROL enabled

enumerator **IN_BAND_TERMINATION_ATTEMPT_COUNT**

Indicates how many times in-band termination signal needs to be sent by controller/initiator to a controlee device. Default : 1

enumerator **SUB_SESSION_ID**

Sub-Session Handle for the controlee device. This config is mandatory and it is applicable if STS_CONFIG is set to 2 for controlee device

enumerator **BPRF_PHR_DATA_RATE**

PHR coding rate Default : 0(850kpbs)

enumerator **MAX_NUMBER_OF_MEASUREMENTS**

Maximum Number of ranging blocks to executed in a session Default : 0

enumerator **UL_TDOA_TX_INTERVAL**

This parameter specifies the average transmission interval of Blink UTMs from UT-Tags and/or Synchronization UTMs from UT-Synchronization Anchors, as defined by the UL-TDoA TX Interval MAC configuration parameter.

By default, UL_TDOA_TX_INTERVAL = 2000ms.

enumerator **UL_TDOA_RANDOM_WINDOW**

Length of the randomization window within which Blink and Synchronization UTMs may be transmitted.

enumerator **STS_LENGTH**

STS length 0: 32 symbols , 1: 64 symbols(Default), 2: 128 symbols

enumerator **SUSPEND_RANGING_ROUNDS**

configuration allows the Ranging Rounds to be suspended

enumerator **UL_TDOA_NTF_REPORT_CONFIG**

UT-Anchor configuration to specify if UL-TDoA related SESSION_INFO_NTF

enumerator **UL_TDOA_DEVICE_ID**

This value shall be used to specify the length and presence of the UL-TDoA Device ID in UTMs.

enumerator **UL_TDOA_TX_TIMESTAMP**

Presence and length of TX timestamps in UTMs.

enumerator **MIN_FRAMES_PER_RR**

Minimum number of frames to be transmitted in a ranging round.(default = 4)

enumerator **MTU_SIZE**

Maximum Transfer Unit (MTU) Size represents the maximum size of allowed payload size to be transmitted in a frame

enumerator **INTER_FRAME_INTERVAL**

The configuration in units of 1200 RSTU to configure the interval between the RFRAMES transmitted in the "OWR for AoA Measurement" ranging round.(default = 1)

enumerator **DLTDOA_RANGING_METHOD**

DL-TdoA ranging round 0: SS-TWR, 1:DS-TWR(default)

enumerator **DLTDOA_TX_TIMESTAMP_CONF**

DL-TdoA Tx timestamp conf b0: TX timestamp type b1-2: TX timestamp length , Default: 00000011

enumerator **DL_TDOA_HOP_COUNT**

controls cluster sync field 1: inter cluster sync filed in every poll DTM

enumerator **DLTDOA_ANCHOR_CFO**

DL-TdoA anchor CFO 0: not included, 1:Anchor CFO included(default)

enumerator **DLTDOA_ANCHOR_LOCATION**

DL-TdoA anchor location

enumerator **DLTDOA_TX_ACTIVE_RANGING_ROUNDS**

DL-TdoA tx active ranging rounds 0: not present (default) 1: present

enumerator **DL_TDOA_BLOCK_STRIDING**

To configure number of blocks that shall be skipped by a DT-Tag between two active ranging blocks, 0x00 : No blocks striding(default) 0x01-0xFF : no. of blocks to be skipped by DT-Tag

enumerator **DLTDOA_TIME_REF_ANCHOR**

global time rference of dl-tdoa network 0: Disable 1: Set global metric time

enumerator **SESSION_KEY**

Session Key provided for Provisioned STS mode (STS_CONFIG equal to 0x03 or 0x04) If the Session Key is not provided by the Host in Provisioned STS mode, the UWBS shall fetch the Session Key from the Secure Component. This parameter is valid only in Provisioned STS mode and shall be ignored otherwise.

enumerator **SUB_SESSION_KEY**

Sub-session Key provided for Provisioned STS for Responder specific Key mode (STS_CONFIG equal to 0x04). If the Sub-session Key is provided by the Host, the Host shall also provide the SESSION_KEY. If the Sub-session Key is not provided by the Host for Provisioned STS for Responder specific Key mode, the UWBS shall fetch the Sub-session Key from the Secure Component. This parameter is valid only in Provisioned STS for Responder specific Key mode and shall be ignored otherwise.

enumerator **SESSION_DATA_TRANSFER_STATUS_NTF_CONFIG**

This parameter is used to configure the SESSION_DATA_TRANSFER_STATUS_NTF. 0x00 = Disable SESSION_DATA_TRANSFER_STATUS_NTF (Default). 0x01 = Enable SESSION_DATA_TRANSFER_STATUS_NTF. If SESSION_DATA_TRANSFER_STATUS_NTF is disabled, then the UWBS shall not send SESSION_DATA_TRANSFER_STATUS_NTF for every Application Data transmission except for last transmission.

enumerator **SESSION_TIME_BASE**

Configures a reference time base for the given session. Octet 0: b0: Reference time base feature 0b0 = Enable 0b1 = Disable (default) b1: continue/stop the session(s) when reference session is not in SESSION_STATE_ACTIVE Session State 0b0 = stop (default) 0b1 = continue b2: Resync time grid in case the reference session will become active again after it has been inactive. 0b0 = No resync (default) 0b1 = Resync Octet 1-5: Session Handle of the reference session Octet 5-8: Session offset time in microseconds

enumerator **DL_TDOA_RESPONDER_TOF**

This parameter specifies whether a DT-Anchor with the Responder role in a given ranging round shall include the estimated Responder ToF Result in a Response DTM. 0x00: Responder ToF Result shall not be added to Response DTMs (default). 0x01: Responder ToF Result shall be added to Response DTMs.

enumerator **APPLICATION_DATA_ENDPOINT**

Local endpoint configuration of the session It defines which endpoint is used by the UWBS for Application data exchange using the non-secure or secure message connection. When using the Bypass mode, all data shall be exchanged using the Non-secure endpoint Values b3-b0: Non-secure end point configuration 0x0 : Host (default) 0x1: Secure Component 0xF- 0x2 : RFU b7-b4: Secure end point configuration 0x0 : Host (default) 0x1: Secure Component 0xF- 0x2 : RFU

enumerator **END_OF_SUPPORTED_APP_CONFIGS**

End of App Configs

enum **vendorAppConfig**

Set Get Vendor App Configuration parameters supported in UWB API layer.

*Values:*

enumerator **MAC_PAYLOAD_ENCRYPTION**

This parameter shall enable disable encryption of Payload data 0x00 - Plain Text 0x01 - Encrypted(default)

enumerator **ANTENNAE_CONFIGURATION_TX**

The antenna used for TX. If Octet[0] of ANTENNAE_CONFIGURATION_TX is 0 and Octet[1] of ANTENNAE_CONFIGURATION_RX is 0, FW automatically enters Scan Phase. Octet[0] - Define number of TX Antennas to follow (default value is 1). Octet[1] - Tx Antennas ID as defined by ANTENNA_TX_DEFINE (default value is 1). So we transmit by default with Antennas ID 1 (As a pre requisite: an antenna with ID 1 using ANTENNA_TX_IDX_DEFINE must be explicitly pre-defined). Octet[2] - Tx Antennas ID as defined by ANTENNA_TX_DEFINE Must be 0 for SR100T, SR150 & SR160..

enumerator **ANTENNAE_CONFIGURATION_RX**

The session specific antenna configuration for Rx If Octet[0] of ANTENNAE_CONFIGURATION_TX is 0, 0, and Octet[0] of ANTENNAE_CONFIGURATION_RX is 0,0 FW automatically enters Scan Phase. Octet [0] : Mode of RX operation – 0 : Configuration ToA Mode – ToF Only Mode – 1 : Configuration AoA Mode – Dual / Single AoA usecase – ToA

Mode with implicit Rx mode as per ANTENNAS_RX_PAIR_DEFINE – Test / Loopback mode usecase – 2 : Configuration Mode 2: Radar Mode – Default 0x01 (Note: SR100S only) – 3 : Configuration Mode 3: ToA usecase using different Rx antenna pair for RFM – 4 : Configuration Mode 4: AoA usecase using different Rx antenna pair for RFM – 5 : Configuration Mode 5: ToA mode for CSA – 6 : Configuration Mode 6: AoA mode for CSA Octet [1] : Number of Antennas or Antenna pairs to follow Default Value: Generic Session will have Octet[0] : 0x00 Octet[1] : 0x01 Test Session will have Octet[0] : 0x01 Octet[1] : 0x01

Note : Please refer the spec for more details on each configuration mode.

enumerator **RAN_MULTIPLIER**

Return the possible RAN multiplier value for a new session

enumerator **STS_LAST_INDEX_USED**

Parameter used to get the STS index of the UWB session. When GET_VENDOR_APP_CONFIG_CMD issued for this config during SESSION_STATE_ACTIVE the UWBS shall return the last

enumerator **CIR_LOG_NTF**

0x00: Disable (default) 0x01: Enable

enumerator **PSDU_LOG_NTF**

0x00: Disable (default) 0x01: Enable

enumerator **RSSI_AVG_FILT_CNT**

This parameter is used to filter out the outliers in RSSI measurements in PER RX Test. If the RSSI filtering count is set to N and total packet count is set to M then UWBS shall report the average of (M-2N) RSSI values in TEST_PER_RX_NTF excluding(N) maximum and (N) minimum RSSI values. Note: M is the total packet count to be received in PER Rx test (default = 0)

enumerator **CIR_CAPTURE_MODE**

CIR sampling position for incoming UWB packet bits[7:4] - CIR1 capture mode bits[3:0] - CIR0 capture mode CIR capture modes: 0x0 - Pre SYNC RX1 0x1 - Pre SYNC RX2 0x2 - Pre STS RX1 0x3 - Pre STS RX2 0x4 - Post SYNC RX1 0x5 - Post SYNC RX2 0x6 - Post STS RX1 0x7 - Post STS RX2 0x8 - 0xF - RFU (default = 0x76)

enumerator **RX_ANTENNA_POLARIZATION_OPTION**

This parameter is used to choose the antenna polarization option when AOA measurement is enabled (AOA_ RESULT_REQ = 1). This parameter is not applicable when AOA_RESULT_REQ = 0. bit[0]: Polarization option to be used for the first AoA computation

bit[1]: Polarization option to be used for the second AoA computation when enabled either via DUAL_AOA_ PREAMBLE_STS or NUMBER_OF_STS_SEGMENTS = 2. The value is not applicable when second AoA computation is not enabled

bit[7:2]: RFU

enumerator **SESSION_SYNC_ATTEMPTS**

Number of times scheduler shall attempt to sync in controlee session before reporting error notification. This config is applicable for controlee session only. Range: [3 : 255] (Default: 3)

enumerator **SESSION_SCHED_ATTEMPTS**

Number of times scheduler shall attempt to schedule ranging round before reporting error notification Range: [1 : 255] (Default: 3)

enumerator **SCHED_STATUS_NTF**

Enable/disable SCHEDULER_STATUS_NTF 0x00 - Disable (default) 0x01 - Enable for including all the sessions information in notification 0x02 - Enable for include only failure sessions information in notification 0x03-0xFF: RFU

enumerator **TX_POWER_DELTA_FCC**

Session specific Tx power ID offset applied on top of Tx POWER calibration parameter. configured via SET_ DEVICE_CALIBRATION_CMD. 0:No offset (default) 1 to 127: Attenuation (0.25 dB per steps) 128 to 255: RFU

enumerator **TEST_KDF_FEATURE**

This parameter is used to enable/disable KDF notification generation. 0x00: Disable (default) 0x01: Enable

enumerator **TX_POWER_TEMP_COMPENSATION**

This parameter is used to enable/disable Tx power temperature compensation 0x00: Disable (Default) 0x01: Enable

enumerator **WIFI_COEX_MAX_TOLERANCE_COUNT**

WiFi-CoEx maximum tolerance count, after the expiry of the number of count the UWBS shall make the “Medium Grant Request” with priority field set to “Critical”. This parameter can be modified when session is in SESSION_STATE_ACTIVE Session State. Range: [1 : 25] (Default: 3) enumerator **ADAPTIVE_HOPPING_THRESHOLD**

This parameter can be used to configure the required number of successful responses(T) from Responders to conclude a successful ranging round. If numbers of responses is less than this given threshold(T) when Initiator device acting as Controller then initiator device triggers a hop to a different round index within the next block. Range: [0<T<= NUMBER_OF_CONTROLEES] (Default: NUMBER_OF_CONTROLEES) Note: This parameter is applicable when HOPPING_MODE = 0xA0 (NXP Adaptive Hopping mode is Enabled)

enumerator **AUTHENTICITY_TAG**

Config to enable/disable authenticity tag in RANGE_DATA_NTF. This config is applicable when STS_CONFIG is set to Dynamic STS. 0x00 = Disable (Default) 0x01 = Enable

enumerator **RX_NBIC_CONFIG**

Octet [0] : Used to configure NBIC settings b[0]: Enable / Disable NBIC. 0 = Disable (default) 1 = Enable b[1:2]: Content of register MA_FILTER_BW_SET. Filter bandwidth setting (default: 0x3) b[3:4]: Content of register MA_FILTER_BW_START_SET. Starting filter bandwidth setting for estimation (default: 0x3) b[5:7]: RFU Octet [1]: Content of register PSD_WEIGHT_SET (Default: 0x14 ) (Default: 0x40 only applicable when NBIC is enabled )

enumerator **MAC_CFG**

Config is used to configure the MAC Header and MAC Footer b[0]: MAC Header present b[1]: MAC Footer present b[7:2]: RFU (Default value: 0x03 for FIRA Session) (Default value: 0x00 for Test Mode Session)

enumerator **SESSION_INBAND_DATA_TX_BLOCKS**

Amount of blocks which should be reserved for the given session for inband data transfer for transmitting data. If set to 0, transmitting inband data is not allowed for this session. Note: The sum of this value for all active session must not exceed UWBS_INBAND_DATA_MAX_BLOCKS and will prevent the first session which is causing to exceed the limit to get started. Default Value: 0 for Ranging Session UWBS_INBAND_DATA_BUFFER_BLOCK_SIZE for Data Session with DEVICE_TYPE set to Controlee 0 for Data Session with DEVICE_TYPE set to Controller

enumerator **SESSION_INBAND_DATA_RX_BLOCKS**

Amount of blocks which should be reserved for the given session for inband data transfer for receiving data. If set to 0, receiving inband data is not allowed for this session. Note: The sum of this value for all active session must not exceed UWBS_INBAND_DATA_MAX_BLOCKS and will prevent the first session which is causing to exceed the limit to get started. Default Value: 0 for Ranging Session UWBS_INBAND_DATA_BUFFER_BLOCK_SIZE for Data Session with DEVICE_TYPE set to Controller 0 for Data Session with DEVICE_TYPE set to Controlee

enumerator **ANTENNAE_SCAN_CONFIGURATION**

List of scanning pairs for Antennas. Assuming we have Anteannes East, West, North, South(E, W, N, S) Octet[0 + 0] E-TX for 1st Round Octet[0 + 1] E-RX1 for 1st Round Octet[0 + 2] W-RX2 for 1st Round Octet[3 + 0] N-TX for 2nd Round Octet[3 + 1] N-RX1 for 2nd Round Octet[3 + 2] S-RX2 for 2nd Round More entries as needed. Antennas IDs as defined by ANTENNA_TX_IDX_DEFINE and ANTENNA_RX_IDX_DEFINE. Once FW detects which Antenna has In case of 3D AoA, Even IDs of RX Pair are for H and Odd IDs are for V Configuration as an enforced convention.

enumerator **DATA_TRANSFER_TX_STATUS_CONFIG**

This configuration shall be used to configure DATA_TRANSMISSION_STATUS_NTF indication 0x00 : Always ON 0x01 : Always OFF 0x02 : Notify when error (Default: 0x00) Note: The UWBS shall always send DATA_TRANSMISSION_STATUS_NTF whenever it receives DATA_MESSAGE_SND, the subsequent DATA_TRANSFER_TX_STATUS_NTF on RF transmit shall be sent based on DATA_TRANSFER_TX_STATUS_CONFIG configuration

enumerator **ULTDOA_MAC_FRAME_FORMAT**

Parameter to select MAC frame format for UL-TDOA Tag device 0x00: FIRA (Default) 0x01: Vendor-specific MAC format Note: This parameter is only applicable when RANGING_ROUND_USAGE = 0x00 (One Way Ranging UL-TDoA) and DEVICE_ROLE = 0x04 (UT-Tag)

enumerator **RFRAME_LOG_NTF**

This configuration is used to enable/disable RFRAME LOG NTF. 0x00 = Disable (default) 0x01 = Enable Values 0x02 to 0xFF = RFU

enumerator **TX_ADAPTIVE_PAYLOAD_POWER**

This configuration is used to enable/disable adaptive payload power for TX. 0x00 = Disable 0x01 = Enable (default) Values 0x02 to 0xFF = RFU

enumerator **SWAP_ANTENNA_PAIR_3D_AOA**

Session specific configuration parameter is used to swap the antenna pair for RFM reception.

0x00 = not swap ( Same pairs are used for all message reception) (Default) 0x01 = Pair1 and Pair 2 are swapped for RFM reception

When SWAP_ANTENNA_PAIR_3D_AOA is set to 0x01 then RSSI measurements shall be report for all pairs in the RANGE_DATA_NTF. Applicable only for Responder.

enumerator **RML_PROXIMITY_CONFIG**

Octet 0: RML_NEAR_PROXIMITY (default = 0) This parameter sets the lower bound in meters where the discovered devices are added into the RML list. Should be less than or equal to RML_FAR_ _CONFIG value.

Octet 1: RML_FAR_PROXIMITY (default = 5) This parameter sets the upper bound in meters above which the RML list is not added with the discovered devices. Should be greater than or equal to RML_NEAR_CONFIG value.

enumerator **CSA_MAC_MODE**

This configuration is used to configure: 1.The number of active ranging rounds in a RANGING_DURATION. 2.Offset between two active ranging rounds in a RANGING_DURATION. [b7-b6]: Number of active ranging round(s) 0 = One active ranging round (default). 1 = Two active ranging rounds (CSA use case). 2 and 3 = RFU. [b5-b0]: Offset between two active ranging rounds. 1 to (Nround - 1)

Note: Bits [b5, b0] SHALL be set if [b7, b6] set to decimal value 1. Otherwise, bits [b5, b0] will be ignored. Nround is calculated based on RANGING_DURATION, SLOTS_PER_RR and SLOT_DURATION.

enumerator **FOV_ENABLE**

2D AoA FoV Processing Enable/Disable This parameter decides whether to enable or disable 2D AoA FoV Processing. 0x00 : 2D AoA FoV Processing Disabled (Default) 0x01 : 2D AoA FoV Processing Enabled

enumerator **AZIMUTH_FIELD_OF_VIEW**

Field of View for Horizontal Antenna Pair. This parameter indicates if the peer device is in the configured FoV of the device or not. Octet[0] : Horizontal RX Antenna Pair ID as defined in 'ANTENNAS_RX_PAIR_DEFINE UCI parameter'. Value 0 shall be rejected. Octet[1] : FoV Coverage in degrees.

enumerator **CSA_FINAL_DATA2_CONFIG**

Configuration to enable/disable transmission of Final Data 2 from Responder to

Initiator. 0x00 = Responder shall not transmit Final Data 2 message (default). 0x01 = Responder shall transmit the Final Data 2 message

enum **UWB_SR1XX_DBG_CFG**

Debug Configuration parameters supported in UWB API layer.

*Values:*

enumerator **kUWB_SR1XX_DBG_CFG_DATA_LOGGER_NTF**

enumerator **kUWB_SR1XX_DBG_CFG_TEST_CONTENTION_RANGING_FEATURE**

enumerator **kUWB_SR1XX_DBG_CFG_CIR_CAPTURE_WINDOW**

enumerator **kUWB_SR1XX_DBG_CFG_RANGING_TIMESTAMP_NTF**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_SECURE**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_SECURE_ISR**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_NON_SECURE_ISR**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_SHELL**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_PHY**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_RANGING**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_SECURE_ELEMENT**

enumerator **kUWB_SR1XX_DBG_CFG_THREAD_UWB_WLAN_COEX**

enumerator **END_OF_SUPPORTED_EXT_DEBUG_CONFIGS**

End of Ext Debug Configs

enum **UWB_AppParams_type**

Set/Get App Configuration parameters type supported in UWB API layer.

*Values:*

enumerator **kUWB_APPPARAMS_Type_Unknown**

We don't know the type

enumerator **kUWB_APPPARAMS_Type_u32**

It's a 32 bit value

enumerator **kUWB_APPPARAMS_Type_au8**

It's an array of 8 bit values

enumerator **kUWB_APPPARAMS_Type_Unknown**

We don't know the type

enumerator **kUWB_APPPARAMS_Type_u32**

It's a 32 bit value

enumerator **kUWB_APPPARAMS_Type_au8**

It's an array of 8 bit values

enum **UWB_DebugParams_type**

Set/Get Debug Configuration parameters type supported in UWB API layer.

*Values:*

enumerator **kUWB_DEBUGPARAMS_Type_u8**

It's a 8 bit value

enumerator **kUWB_DEBUGPARAMS_Type_u16**

It's a 16 bit value

enumerator **kUWB_DEBUGPARAMS_Type_u32**

It's a 32 bit value

enum **deviceConfig**

Device Configuration parameters supported in UWB API layer.

*Values:*

enumerator **LOW_POWER_MODE**

0:DISABLE, 1:ENABLE

enumerator **DPD_ENTRY_TIMEOUT**

DPD entry timeout in ms (default = 300ms)

enumerator **HPD_ENTRY_TIMEOUT**

DPD entry timeout in ms (default = 300ms)

enumerator **MHR_IN_CCM**

enumerator **DDFS_TONE_CONFIG_ENABLE**

enumerator **NXP_EXTENDED_NTF_CONFIG**

0x00 = FIRA generic Response/Notification (Default) 0x01 = Vendor extended Response/Notification

enumerator **END_OF_SUPPORTED_DEVICE_CONFIGS**

enumerator **LOW_POWER_MODE**

0:DISABLE, 1:ENABLE

enumerator **DPD_WAKEUP_SRC**

DPD wakeup source bit1: GPIO1, bit3: GPIO3

enumerator **WTX_COUNT_CONFIG**

WTX count, 20>= wtx count <=120

enumerator **DPD_ENTRY_TIMEOUT**

DPD entry timeout in ms (default = 500ms)

enumerator **WIFI_COEX_FEATURE**

This configuration is used to configure the wifi co-ex feature.

Octet[0]: Enable/Disable wifi co-ex feature 0x00 : To Disable(default) b3-b0: Enable/Disable functionality CoEx 1: Enable CoEx Interface without Debug and without Warning Verbose 2: Enable CoEx Interface with Debug Verbose only GPIO2 toggle status before start of ranging round and end of ranging round will be indicated via UWB_WIFI_COEX_IND_NTF 3: Enable CoEx Interface with Warnings Verbose only UWB_WLAN_COEX_MAX_ACTIVE_GRANT_DUARTION_EXCEEDED_WAR_NTF will be send when WLAN max grant duration is exceeded 4: Enable CoEx Interface with both Debug and Warning Verbose b7-b4: CoEx Interface (GPIO/UART/One

Wire) selection: 0: GPIO Interface 1: Uart Interface 2: One Wire Interface(Proposal new) Octet[1]: MIN_GUARD_DURATION. RFU for one wire interface, FW force this to 0 to manage internal scheduler logic. Octet[2]: MAX_GRANT_DURATION / MAX_WIFI_BLOCK_DURATION(will be renamed for One Wire Co-ex) Maximum duration for which the UWB can request for medium access Default is 30ms, Maximum = 255ms Octet[3]: ADVANCED GRANT DURATION / GUARD DURATION(will be renamed for One Wire Co-ex). OneWire Co-ex - Guard Duration before which UWBS can perform Tx/Rx, applied before start of every co-ex session, default = 1ms, Minimum =1ms, Maximum = 255ms

enumerator **RX_GPIO_ANTENNA_SELECTION**

This configuration is used to indicate whether the RX Antenna selection should take place w.r.t EF2 or GPIO14 (for Antenna switching) 0 : EF2 based Antenna selection (Default) 1 : GPIO14 based Antenna selection

enumerator **TX_BASE_BAND_CONFIG**

0:DISABLE, 1:ENABLE

enumerator **DDFS_TONE_CONFIG**

DDFS tone config (4*8 bytes repeated for 4 channels) 18 bytes value description: Octet[0]: channel number Octet[1]: Tx antenna selection. Possible values are 1 or 2. Octet[5:2]: Content of register TX_DDFS_TONE_0 Octet[9:6]: Content of register TX_DDFS_TONE_1 Octet[13:10]: Duration of the spur, in 124.8 MHz resolution (~ 8 ns) Octet[14]: Content of register GAINVAL_SET Octet[15]: Content of register DDFSGAINBYPASS_ENBL Octet[17:16]: Periodicity of spur in terms of gap interval in the PER command. 4 Blocks: 18 Octets repeated for each block Octets[17:0] correspond to Block1 Octets[35:18] correspond to Block2 Octets[53:36] correspond to Block3 Octets[71:54] correspond to Block4

enumerator **TX_PULSE_SHAPE_CONFIG**

Preamble pulse shape setting Octet[0]: Preamble pulse shape id Octet[1]: Payload Tx pulse shape id Octet[2]: STS Tx pulse shape ID Octet[3]: DAC Stage Config Values: Octet[0-2] = [2(default), 30, 34, 36 and 37] Octet[3] = Value is defined as below bit0: To set the DAC gain 0: Unchanged( UWBS Keeps previous assigned value) . 1: UWBS set to 0x24 bit1: To set LPF( Tx DAC C) 0: UWBS shall set to 0 1: UWBS shall set to 0x5F bit7-bit2: RFU

enumerator **CLK_CONFIG_CTRL**

Octet[0]: Clock source option

- b[0]: RF Clock option
- [0]: Use on board crystal (default)

- [1]: Use external Clock
- b[1]: Slow clock option
- [0]: Use on board crystal
- [1]: Use external 32.768 KHz Clock
- b[2:7]: RFU Octet[1]: Crystal Option for RF clock
- b[0]: Crystal Option
- [0]: Use 38.4MHz crystal (default)
- [1]: Use 26 MHz crystal
- b[1:7]: RFU

enumerator **HOST_MAX_UCI_PAYLOAD_LENGTH**

Parameter is used to set host capability of handling max UCI payload. FW shall use this parameter to send the UCI responses/notification to host. FW shall use PBF bit if UCI payload goes more than HOST_MAX_UCI_PAYLOAD_LENGTH size. 255<= HOST_MAX_UCI_PAYLOAD_LENGTH<= UWBS_MAX_UCI_PAYLOAD_LENGTH(capability parameter) (Default = 255)

enumerator **NXP_EXTENDED_NTF_CONFIG**

0x00 = FIRA generic Response/Notification (Default) 0x01 = Vendor extended Response/Notification

enumerator **CLOCK_PRESENT_WAITING_TIME**

Maximum waiting time until clock is present. The value is given in microseconds and defaults to 1000us. If Octet [0] of CLK_CONFIG_CTRL is set to 0 (on board crystal), this time is indicating the maximum waiting time until XTAL oscillator becomes stable. If Octet [0] of CLK_CONFIG_CTRL is set to 1 (external clock), this time is indicating the guard time between clock request GPIO (GPIO1) going high until the platform has to provide a stable clock.

enumerator **INITIAL_RX_ON_OFFSET_ABS**

Negative offset when Rx should be enabled to receive the first message of a ranging round on Controlee compared to expected reception time. Default = 100 us

enumerator **INITIAL_RX_ON_OFFSET_REL**

Negative offset when Rx should be enabled to receive the first message of a ranging round on Controlee compared to expected reception time. Default = 100 ppm

enumerator **WIFI_COEX_UART_USER_CFG**

UART based WiFi-CoEx Interface User Configuration. Default UWB WLAN Coex-Config : 0x01 Valid ranging 1 to 255 (seconds): The Home channel information request and Band channel information requests will be sent by UWB after specified duration.

enumerator **PDOA_CALIB_TABLE_DEFINE**

PdoA Calibration Table Definition : Octet[0] - Calibration Step Size : The calibration table step size in degrees which indicates the step size between two consecutive points. Allowed Range : 10° to 15° (default = 12°)

Example : For maximum span of 60°, with step size 12° then the calibration table would look like[-60°, -48°, -36° , ..., 0°, ..., 36°, 48°, 60°]. Octet [1] - Number of Steps: The number of calibration steps needs to be an odd number.

Allowed Range : 3<=M<=21(to include 0°)(default = 11) Example : With number of step size as 10° and number of steps as 13, the achieved calibration span is -60, -50, -40, ..., -10, 0, 10, ..., 40, 50, 60 Note:

Total Calibration Span = ((Number of Steps - 1) * Step Size). Example : If steps are 11 and step size is 12° then total span is 120°(-60° to 60° including 0°). If steps are 13 and step size is 15° then total span is 180°(-90° to 90° including 0°)

enumerator **ANTENNA_RX_IDX_DEFINE**

To define/create antenna identifier for RX Octet[0] : Number of Entries. (N). This must be equal to "MAX_N".

Array of entries: Octet[X + 0] : RX Antennae ID. Index of the Antennae. Value 0 shall be Invalid. Value shall from 1 to MAX_N

Octet[X + 2, X + 1]: GPIO Filter Mask This mask defines which GPIOs shall be changed during the state transition, if a GPIO bit is set to 0 the GPIO shall not change its state. It's a 2 byte value in Little Endian format.

Octet[X + 4, X + 3]: GPIO State / Value This mask defines the GPIO state, if the corresponding GPIO bit is 0 in the GPIO Filter mask, the state shall be ignored and not changed It's a 2 byte value in Little Endian format.

enumerator **ANTENNA_TX_IDX_DEFINE**

To define/create antenna identifier for TX

Octet[0] : N Entries.

Array of entries: In case, same Antenna is used for Tx/Rx Use the same ID. In case few GPIOs have to be switched for toggling, Octet[1] and Octet[2] would be different when TX/RX antennae are defined.

Octet[0] : Antennae ID. Index of the Antennae. Value 0 shall be Invalid. Value shall from 1 to N.

Octet[1]: GPIO Filter Mask This mask defines which GPIOs shall be changed during the state transition, if a GPIO bit is set to 0 the GPIO shall not change its state.

Octet[2]: GPIO State / Value This mask defines the GPIO state, if the corresponding GPIO bit is 0 in the GPIO Filter mask, the state shall be ignored and not changed

Octet[6:3] Group Delay

Max number of entries per product variant for max value of N is same as ANTENNA_RX_IDX_DEFINE For Calibration of TX Power, use TX_POWER from Calibration Parameters

enumerator **ANTENNAE_RX_PAIR_DEFINE**

To define/create antenna identifier for RX Pair

Octet[0] : N Entries.

Array of entries: This may be an H or V Combination. Repeat of [ Octet[0] : Antennae PAIR ID This IDx is used along with ANTENNAE_CONFIGURATION(Session Config) and for reporting. ID 0 shall not be used. For non-scanning mode/for backward compatibility, Default ID for Horizontal has to be 1. And ID for Vertical has to be 2. Use Odd Values for Horizonal Pair And Even Values for Vertical Paris. So, FW can decide to move to lower or higher Pair ID. FW can also use IDs X, X+1 to identify which ID is making a group for 3D Configuration.

Octet[1] : RX1 Antennae ID as defined by ANTENNAE_RX_IDX_DEFINE

Octet[2] : RX2 Antennae ID as defined by ANTENNAE_RX_IDX_DEFINE.

Octet[4:3]: PDOA Zero Offset

- 2 bytes PDOA1_OFFSET, PDOA2_OFFSET

Octet[6:5] : Relative Angle of View / Field of View. Assuming we have 4 Antennae pairs symmetrically placed on a UWB System. For a given selected Antennae pair, the peer object may be at 0°, but reported AoA has to be either 0°, 90°, 180° or 270° depending on this relative Angle of View.

When this is set to 0, then Host has to use RX Antennae info and derive the relative angle. When this is set to 0, the FW has no IDEA which antennae group to switch to in case of antennae moves out of some configuration.

Max number of entries per product variant for max value of N is same as ANTENNA_RX_IDX_DEFINE

enumerator **WIFI_CO_EX_CH_CFG**

To select the WIFI Co-ex channel

b0: Channel 5, Set to 1 enable Wifi Co-ex on Channel 5 b1: Channel 6, Set to 1 enable Wifi Co-ex on Channel 6 b2: Channel 8, Set to 1 enable Wifi Co-ex on Channel 8 b3: Channel 9, Set to 1 enable Wifi Co-ex on Channel 9 b4:b7 - RFU 0x0F, enable co-ex for all channels i.e ch5,ch6,ch8 and ch9 enumerator **`END_OF_SUPPORTED_DEVICE_CONFIGS`**

End of device Configs

enum **`kUWBAntCfgRxMode_t`**

Antenna Configuration and slection mode

*Values:*

enumerator **`kUWBAntCfgRxMode_ToA_Mode`**

enumerator **`kUWBAntCfgRxMode_AoA_Mode`**

enumerator **`kUWBAntCfgRxMode_Radar_Mode`**

For Rx RADAR

enumerator **`kUWBAntCfgRxMode_ToA_Rfm_Mode`**

For Rx TOA RFM

enumerator **`kUWBAntCfgRxMode_AoA_Rfm_Mode`**

For Rx AOA RFM

enumerator **`kUWBAntCfgRxMode_CSA_ToA_Mode`**

ToA mode for CSA

enumerator **`kUWBAntCfgRxMode_CSA_AoA_Mode`**

AoA mode for CSA

enum **`otpParam_Type`**

OTP Read Write Configuration parameters supported in UWB API layer.

*Values:*

enumerator **`kUWB_OTP_ModuleMakerInfo`**

2 bytes of Module maker info

enum **`calibParam`**

Calibrations Configuration parameters supported in UWB API layer.

*Values:*

enumerator **TX_POWER_DIFF**

A difference between received and expected TX power

enumerator **FREQ_DIFF**

A difference (in Hz) between received and expected UWB frequency

enumerator **ANTENNA_DELAY**

Antenna delay in 15.65ps resolution

enumerator **CURRENT_LIMIT_VALUE**

Current Limiter Value, 0:minimum 20:maximum, 20:default

enumerator **GROUP_DELAY**

Groupdelays in trim page are calculated with SP3 frames flashed in radio indexes. Length - 6 Octet 0: Recalculate (144 Octets ForRead trim value) Groupdelay index according to radio configuration index Groupdelay index second byte is a NBIC type (0: NBIC disabled, 1: NBIC enabled, 2: NBIC enabled with Low frequency config following 4 bytes represent an group delay in 15.65ps resolution 0x0: 0 ps (no delay); 0xFFFFFFFF: 67 216 238 166,75 ps (max) By default group delay is calculated at the very first startup.

enumerator **TEMP_COMPENS_FLAG**

Temp compens flag Value

enumerator **TX_ADAPTIVE_POWER_CALC**

Tx Adaptive power calc Value

enumerator **DDFS_TONE_VALUES**

DDFS Tone Values

enumerator **DPD_TIMER_PENALTY_US**

DPD Timer Penalty in us

enumerator **WAKEUP_SENSOR_ENABLE**

Enable/Disable sensor-based wakeup. 1 : Enable 2 : Disable(Default) Any other value is Invalid.

enumerator **SLOW_BLINK_INTERVAL**

Blink interval when no movement is detected (in seconds) Min: 1 second. Max: 56 minutes.

enumerator **TEMPERATURE_SOURCE**

Source of Thermistor, internal or external. 0 : External Thermister (Default) 1 : Internal. Only applicable when TEMP_COMPENS_FLAG is enabled

enumerator **THERMISTOR_RP**

Value of "RP" (resistor) in Ohms present in externaltemperature sensor measurement unit Only applicable when TEMP_COMPENS_FLAG is enabled

enumerator **LUT_XTAL**

LUT_XTAL First 2 bytes, start address Second 2 bytes, Length N Even Octets Value to be written

enumerator **LUT_THERMISTER**

LUT_TEMPRATURE First 2 bytes, start address Second 2 bytes, Length N Even Octets Value to be written

enumerator **CUSTOM_LUTS_ENBL**

Enable/Disable usage of customer specific LUTs 1 : Enable 2 : Disable(Default) Any other value is Invalid.

enumerator **LUT_TEMP_RT_OFFSET**

Offset for custom temperature LUT (Thermistor Resistance value in Ohms corresponding to 1st TEMP LUT entry). Default value 600

enumerator **LUT_TEMP_RT_SLOPE**

slope for custom temperature LUT (Thermistor Resistance increment in Ohms between 2 successive TEMP LUT entries). Default value 100

enumerator **TRIMVALUES_GET_TEMPERATURE_X10**

Read only parameter This configuration returns calculated temperature after applying applicable lookup tables.

enumerator **TRIM_POWER_CTRL_PREAMBLE_10**

Trim Power control of preamble 10. Default value 2

enumerator **TRIM_POWER_CTRL_PREAMBLE_27**

Trim Power control of preamble 27. Default value 2

enumerator **TRIM_PREAMBLE_10_27_PAYLOAD_PS_COEFFS_OVERRIDE**

Enable/Disable usage of trim preamble 10 and 27 coefficents. 1 : Enable(Default) 2 : Disable Any other value is Invalid.

enumerator **MODULE_MAKER_ID**

enumerator **TX_POWER_DIFF_ANT_BOTTOM**

A difference between received and expected TX power for Bottom Antenna

enumerator **ANTENNA_DELAY_ANT_BOTTOM**

Antenna delay in 15.65ps resolution for Bottom Antenna

enumerator **TX_ADAPTIVE_POWER_CALC_ANT_BOTTOM**

Tx Adaptive power calc Value for Bottom Antenna

enumerator **TRIM_LOCK**

Bit field to determine what is locked what change is no longer allowed. 0- locks FREQ_DIFF. 1 - locks ANTENNA_DELAY 2 - locks GROUP_DELAY 3 - locks TEMP_COMPENS_FLAG 4 - locks TRIMVALUES_DDFS_TONE_VALUES 5 - locks all parameters related to XTAL.I.e TRIMVALUES_ TEMPERATURE_SOURCE, TRIMVALUES_THERMISTOR_RP, TRIMVALUES_LUT_XTAL,TRIMVALUES_LUT_TEMPERATURE, TRIMVALUES_CUSTOM_LUTS_ENBL 6-31 : RFU 0xFFFFFFFF - Lock Everything

enumerator **VCO_PLL**

VCO_PLL calibration - Channel dependent

enumerator **RF_CLK_ACCURACY_CALIB**

RF_CLK_ACCURACY_CALIB - Channel independent

enumerator **RX_ANT_DELAY_CALIB**

Delay Calibration for each RX Antenna - Channel dependent

enumerator **PDOA_OFFSET_CALIB**

PDOA Offset Calibration - Channel dependent

enumerator **TX_POWER_PER_ANTENNA**

TX_POWER - Channel dependent

enumerator **MANUAL_TX_POW_CTRL**
MANUAL_TX_POW_CTRL

enumerator **PA_PPA_CALIB_CTRL**
TX PA and PPA setting

enumerator **AOA_ANTENNAS_PDOA_CALIB**
PDOA Calibration tables

enumerator **AOA_ANTENNAS_MULTIPOINT_CALIB**
Multi point ('N') PDoA manufacturing offset calibration

enumerator **PDOA_MANUFACT_ZERO_OFFSET_CALIB**
Zero offset Manufacturing PDOA Calibration

enumerator **AOA_THRESHOLD_PDOA**
AoA Threshold PDOA

enumerator **TX_TEMPERATURE_COMP_PER_ANTENNA**
TX temperature compensation per antenna

enumerator **SNR_CALIB_CONSTANT_PER_ANTENNA**
SNR Calibration per RX Antenna - Channel dependent

enumerator **RSSI_CALIB_CONSTANT_HIGH_PWR**
RSSI Offset per RX Antenna for High Power - Channel dependent

enumerator **RSSI_CALIB_CONSTANT_LOW_PWR**
RSSI Offset per RX Antenna for Lower Power - Channel dependent

enumerator **AOA_ANTENNAS_PDOA_CALIB_EXTENDED_SUPPORT**
This parameter is used to set the PDoA Calibration Table for 90 Fov

enum **otpCalibParam**
Calibrations Configuration parameters supported in UWB API layer.

*Values:*

enumerator **OTP_CALIB_VCO_PLL**

VCO PLL Calibration, channel dependent

enumerator **OTP_CALIB_TX_POWER**

TX POWER Calibration, channel and antenna dependent

enumerator **OTP_CALIB_RF_XTAL_CAP**

38.4MHz XTAL, channel and antenna Independent

enumerator **OTP_CALIB_RSSI_CALIB_CONST1**

RSSI Calibration, channel and antenna dependent

enumerator **OTP_CALIB_RSSI_CALIB_CONST2**

RSSI Calibration, channel and antenna dependent

enumerator **OTP_CALIB_MANUAL_TX_POW_CTRL**

power control parameters

enumerator **OTP_CALIB_PAPPPA_CALIB_CTRL**

PA_PPA Calibration control

enumerator **OTP_CALIB_TX_TEMPARATURE_COMP**

Tx Temperature Compensation

enumerator **OTP_CALIB_DELAY_CALIB**

Delay calibration value to adjust the distance measurement from ranging

enumerator **OTP_PDOA_MFG_ZERO_OFFSET_CALIB**

PDOA mfg zero offset calibration

enumerator **OTP_AOA_ANT_MULTIPOINT_CALIB**

Point calibration entries

enum **calibTagOption**

Calibration integrity protection tag options.

*Values:*

enumerator **DEVICE_SPECIFIC**

Device Specific tag option

enumerator **MODEL_SPECIFIC**

Model Specific tag option

enum **eCalibState**

Calibration Parameter States.

*Values:*

enumerator **DEFAULT**

Calibration parameter carries default value by UWBS

enumerator **CUSTOM_NOT_INTEGRITY_PROTECTED**

Calibration parameter is not integrity protected either by a Device specific or Model specific authentication tag. The parameter is applied in UWBS for usage.

enumerator **CUSTOM_AUTH_PENDING**

Calibration parameter integrity check is pending either by a Device specific or Model specific tag. Mainline FW: Until the tag verification finishes, the parameter will not be applied

enumerator **CUSTOM_DEVICE_SPECIFIC_TAG_AUTHENTICATED**

Calibration parameter integrity check is verified successfully by a Device specific tag.

enumerator **CUSTOM_MODEL_SPECIFIC_TAG_AUTHENTICATED**

Calibration parameter integrity check is verified successfully by a Model specific tag

enum **phChannel**

channel for aoa fine calibration, supported parameters

*Values:*

enumerator **CH_5**

channel 5

enumerator **CH_9**

channel 9

enum **antPair**

Antenna pair for aoa fine calibration, supported parameters

*Values:*

enumerator **ANT_1**

enumerator **ANT_2**

enum **localizationZone**

Enumurator lists out the possible values of Localization Zones.

*Values:*

enumerator **eLocZone_Undefined**

enumerator **eLocZone_Frontside**

enumerator **eLocZone_Backside**

## Functions

*tUWBAPI_STATUS* **UwbApi_GetFwCrashLog**(phFwCrashLogInfo_t *pLogInfo)

APIs exposed to application to access UWB Baord Specific Functionality.

Get Firmware Crash Log

**Parameters**

**pLogInfo** – **[out]** Pointer to phFwCrashLogInfo_t

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_SetDefaultCoreConfigs**()

Set Default Core configs.

**Return values**

- **UWBAPI_STATUS_OK** – on success

- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_DoVcoPllCalibration**(uint8_t channel, phCalibRespStatus_t *calibResp)

Do calibration parameters.

**Parameters**

- **channel** – **[in]** Channel
- **calibResp** – **[out]** Pointer to phCalibRespStatus_t

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_SetCalibration**(uint8_t channel, eCalibParam paramId, uint8_t *calibrationValue, uint16_t length)

Set calibration parameters.

**Parameters**

- **channel** – **[in]** channel
- **paramId** – **[in]** Calibration parameter ID
- **calibrationValue** – **[in]** Calibration value
- **length** – **[in]** Calibration value array length

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_GetCalibration**(uint8_t channel, eCalibParam paramId, phCalibRespStatus_t *calibResp)

Get calibration parameters.

Note: For *AOA_ANTENNAS_PDOA_CALIB*, calibResp acts as an in/out param To get the caliberation values of the specified antennaId, the antennaId needs to be passed from application, by accessing the member of phCalibRespStatus_t i.e. inRxAntennaPair.

Example to get the caliberation values for *AOA_ANTENNAS_PDOA_CALIB* with the specific antennaID:

```
phCalibRespStatus_t calibResp = {0x00};
calibResp.inRxAntennaPair = 0x01;
status                      = UwbApi_GetCalibration(channel, AOA_
↪ANTENNAS_PDOA_CALIB, &calibResp);
if (status != UWBAPI_STATUS_OK) {
    NXPLOG_APP_E("Set Calib param AOA_ANTENNAS_PDOA_CALIB Failed
↪");
    goto exit;
}
```

**Parameters**

- **channel** – **[in]** Channel
- **paramId** – **[in]** Calibration param Id
- **calibResp** – **[inout]** Pointer to phCalibRespStatus_t

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_SetDebugParams**(uint32_t sessionHandle, uint8_t noOfparams, const UWB_DebugParams_List_t *DebugParams_List)

Set Uwb Debug Configuration Parameters. This API Can be used to set any number of debug parameters at once.

To easily set the DebugParams list, following macros have been defined.

*UWB_SET_DEBUG_PARAM_VALUE_u8(Parameter, Value)*: This macro sets the value of the corresponding parameter with the given Value.This shall be used to set value of 8 bit wide.

UWB_SET_DEBUG_PARAM_VALUE_u16(Parameter, Value): This macro sets the value of the corresponding parameter with the given Value.This shall be used to set value of 16 bit wide.

UWB_SET_DEBUG_PARAM_VALUE_u32(Parameter, Value): This macro sets the value of the corresponding parameter with the given Value.This shall be used to set value of 32 bit wide.

Example: To set DATA_LOGGER_NTF to zero, macro shall be invoked as given below.

```
UWB_DebugParams_List_t SetDebugParamsList[] = {UWB_SET_DEBUG_
↪PARAM_VALUE(DATA_LOGGER_NTF, 0)};
```

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **noOfparams** – **[in]** Number of App Config Parameters
- **DebugParams_List** – **[in]** Debug parameters values in tlv format

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetDebugParams**(uint32_t sessionHandle, uint8_t noOfparams, UWB_DebugParams_List_t *DebugParams_List)

Get Uwb Debug Configuration Parameters. This API Can be used to get any number of debug parameters at once.

To easily get the DebugParams list, following macro has been defined.

*UWB_SET_GETDEBUG_PARAM_u8(Parameter)*: This macro gets the value of the corresponding parameter.This shall be used to get values of 8 bit wide.

- UWB_SET_GETAPP_PARAM_u16(Parameter): This macro gets the value of the corresponding parameter.This shall be used to get values of 16 bit wide.
- UWB_SET_GETAPP_PARAM_u32(Parameter): This macro gets the value of the corresponding parameter.This shall be used to get values of 32 bit wide.

Example: To get DATA_LOGGER_NTF macro shall be invoked as given below.

```
UWB_DebugParams_List_t GetDebugParamsList[] = {UWB_SET_GETAPP_
↪PARAM(DATA_LOGGER_NTF),};
```

**Parameters**

- **sessionHandle** – **[in]** Initialized Session Handle
- **noOfparams** – **[in]** Number of App Config Parameters
- **DebugParams_List** – **[in]** Debug parameters values in tlv format

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GetBindingCount**(phSeGetBindingCount_t *getBindingCount)

Get the binding count using this API.

**Parameters**

**getBindingCount** – **[out]** getBindingCount data. valid only if API status is success

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized

- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_QueryTemperature**(uint8_t *pTemperatureValue)

API to get the current temperature.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_QueryUwbTimestamp**(uint8_t len, uint8_t pTimestampValue[])

API to get the UWB Timestamp for UWB time synchronization.

On successful execution, this buffer will contain 8 bytes timestamp value.

**Parameters**

- **len** – **[in]** Length of i/p buffer. It should be 8 to hold 8 bytes timestamp value
- **pTimestampValue** – **[out]** Timestamp data. It should be 8 bytes in size to hold 8 bytes timestamp value.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_BUFFER_OVERFLOW** – if response length is more than expected response size
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_VerifyCalibData**(uint8_t *pCmacTag, uint8_t tagOption, uint16_t tagVersion)

Verify Calibration data for all the set Calibration Parameters.

**Parameters**

- `pCmacTag` – **[in]** Cmac Tag
- `tagOption` – **[in]** Tag Option indicating Device/Model Specific tag
- `tagVersion` – **[in]** Tag Version only for Model Specific Tag verification process.

**Return values**

- `UWBAPI_STATUS_OK` – on success
- `UWBAPI_STATUS_NOT_INITIALIZED` – if UWB stack is not initialized
- `UWBAPI_STATUS_INVALID_PARAM` – if invalid parameters are passed
- `UWBAPI_STATUS_TIMEOUT` – if command is timeout
- `UWBAPI_STATUS_FAILED` – otherwise

*tUWBAPI_STATUS* `UwbApi_CalibrationIntegrityProtection`(eCalibTagOption tagOption, uint16_t calibBitMask)

Calibration Integrity Protection for all the Calibration Parameters.

**Parameters**

- `tagOption` – **[in]** Tag Option indicating Device/Model Specific tag
  - 0x00 indicates Device Specific tag option
  - 0x01 indicates Model Specific tag option
- `calibBitMask` – **[in]** bit mask for calibration parameters. Following bits to be set for corresponding calibration parameters to enable integrity protection.
  - bit0 - VCO_PLL
  - bit1 - TX_POWER
  - bit2 - 38.4MHz_XTAL_CAP
  - bit3 - RSSI_CALIB_CONSTANT1
  - bit4 - RSSI_CALIB_CONSTANT2
  - bit5 - SNR_CALIB_CONSTANT
  - bit6 - MANUAL_TX_POW_CTRL
  - bit7 - PDOA_OFFSET
  - bit8 - PA_PPA_CALIB_CTRL
  - bit9 - TX_TEMPERATURE_COMP

- bit10- AOA_FINE_CALIB_PARAM
- bit11- DELAY_CALIB
- bit12- AOA_CALIB_CTRL
- bit13- RFU
- bit14- RFU
- bit15- RFU

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_UpdateActiveRoundsAnchor**(uint32_t sessionHandle, uint8_t nActiveRounds, *UWB_MacAddressMode_t* macAddressingMode, const phActiveRoundsConfig_t roundConfigList[], phNotActivatedRounds_t *pNotActivatedRound)

Update the active rounds during the DL-TDoA Session for a initiator or responder device.

**Parameters**

- **sessionHandle** – **[in]** : Unique Session Handle
- **nActiveRounds** – **[in]** : Number of active rounds
- **macAddressingMode** – **[in]** : MAC addressing mode- 2/8 bytes
- **roundConfigList** – **[in]** : List/array of size nActiveRounds of round index + role tuple
- **pNotActivatedRound** – **[out]** : Structure containing list of not activated index which couldn't be activated, in case return code is *UWBAPI_STATUS_ERROR_ROUND_INDEX_NOT_ACTIVATED*

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized

- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_ERROR_ROUND_INDEX_NOT_ACTIVATED** – if one or more rounds couldn't be activated
- **UWBAPI_STATUS_ERROR_NUMBER_OF_ACTIVE_RANGING_ROUNDS_EXCEEDED** – one or more given rounds exceed number of rounds available
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_UpdateActiveRoundsReceiver**(uint32_t sessionHandle, uint8_t nActiveRounds, const uint8_t RangingroundIndexList[], phNotActivatedRounds_t *pNotActivatedRound)

Update the active rounds during the DL-TDoA Session for a receiver device.

**Parameters**

- **sessionHandle** – **[in]** : Unique Session Handle
- **nActiveRounds** – **[in]** : Number of active rounds
- **RangingroundIndexList** – **[in]** : List/array of size nActiveRounds of round index
- **pNotActivatedRound** – **[out]** : Structure containing list of not activated index which couldn't be activated, in case return code is *UWBAPI_STATUS_ERROR_ROUND_INDEX_NOT_ACTIVATED*

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_SESSION_NOT_EXIST** – if session is not initialized with sessionHandle
- **UWBAPI_STATUS_ERROR_ROUND_INDEX_NOT_ACTIVATED** – if one or more rounds couldn't be activated
- **UWBAPI_STATUS_ERROR_NUMBER_OF_ACTIVE_RANGING_ROUNDS_EXCEEDED** – one or more given rounds exceed number of rounds available
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

struct **phCalibPayload**

*#include <UwbApi_Types_Proprietary.h>* Calib params payload.

### Public Members

uint16_t **VCO_PLL**

VCO PLL code

uint8_t **TX_POWER_ID**[4]

Tx power Id

uint8_t **XTAL_CAP_VALUES**[3]

XTAL cap values. It Can be set only once. Channel independent and remains same for each channel. Octet [0]: 38.4 MHz XTAL CAP1 Octet [1]: 38.4 MHz XTAL CAP2 Octet [2]: 38.4 MHz XTAL GM CURRNT CONTROL Values : [0x00-0xFF] for Octet[1:0] Values : [0x00-0x3F] for Octet[2]

uint8_t **RSSI_CONSTANT1**[8]

RSSI CONSTANT1(4*2). this parameter is channel and antenna pair dependent. Channel number to be provide in calibration commands to set this parameter. 4 antenna pairs with 2 RX each 2 RX: RX1 and RX2

uint8_t **RSSI_CONSTANT2**[8]

RSSI CONSTANT2(4*2). this parameter is channel and antenna pair dependent. Channel number to be provide in calibration commands to set this parameter. 4 antenna pairs with 2 RX each 2 RX: RX1 and RX2

uint8_t **TX_POWER_PARAMS**[16]

Tx power parameters (4*4) 4 Antenna pairs 4 parameters for each antenna pairs Octet[0]: PA_GAIN Octet[1]: PA_DRIVE_GAIN Octet[2]: DIG_GAIN Octet[3]: TX_DAC_GAIN

uint16_t **PA_PPA_CALIB_CTRL**

PA output capacitor control

uint8_t **TX_TEMP_COMP**[16]

Tx tempreature comp. This parameter is dependent on the chosen channel and Tx antenna. (2*4*2): 2 bytes value description Octet [0]:TX_POWER_TEMP_UPPER_BOUND Octet

[1]:TX_POWER_GAIN_INDEX 4 bytes: 2 Octets repeated 4 times to allow up to 4 different temperature ranges Octet[1-0]: Temperature range 1 and gain index Octet[3-2]: Temperature range 2 and gain index Octet[5-4]: Temperature range 3 and gain index Octet[7-6]: Temperature range 4 and gain index 2 bytes: Number of Tx antenna selection options

uint16_t **DELAY_CALIB_VALUE**

Delay calibration. It Can be set only once. Same value will be applied for all channels

struct **UWB_AppParams_value_au8**

*#include <UwbApi_Types_Proprietary.h>* Set/Get App Configuration parameters value type supported in UWB API layer.

union **UWB_AppParams_value**

*#include <UwbApi_Types_Proprietary.h>* Set/Get App Configuration parameters value structure supported in UWB API layer.

### Public Members

uint32_t **vu32**

UWB_AppParams_value_au8_t **au8**

struct **UWB_Debug_Params_value**

*#include <UwbApi_Types_Proprietary.h>* Set/Get Debug Configuration parameters value type supported in UWB API layer.

union **UWB_DebugParams_value**

*#include <UwbApi_Types_Proprietary.h>* Set/Get Debug Configuration parameters value structure supported in UWB API layer.

### Public Members

uint8_t **vu8**

uint16_t **vu16**

uint32_t **vu32**

UWB_Debug_Params_value_t **param**

struct **UWB_AppParams_List**

*#include <UwbApi_Types_Proprietary.h>* Set/Get App Configuration parameters list supported in UWB API layer.

### Public Members

eAppConfig **param_id**

Input: search this tag

UWB_AppParams_type_t **param_type**

Filled Implicitly: Expected type.

UWB_AppParams_value_t **param_value**

Input: Parameter Value

struct **UWB_VendorAppParams_List**

*#include <UwbApi_Types_Proprietary.h>* Set/Get Vendor App Configuration parameters list supported in UWB API layer.

### Public Members

eVendorAppConfig **param_id**

Input: search this tag

UWB_AppParams_type_t **param_type**

Filled Implicitly: Expected type.

UWB_AppParams_value_t **param_value**

Input: Parameter Value

struct **UWB_WiFiCoEx_Ftr**

*#include <UwbApi_Types_Proprietary.h>* This configuration is used to configure the WiFi CoEx feature.

struct **UWB_DebugParams_List**

*#include <UwbApi_Types_Proprietary.h>* Set/Get App Configuration parameters list supported in UWB API layer.

### Public Members

UWB_SR1XX_DBG_CFG_t **param_id**

Input: search this tag

UWB_DebugParams_type_t **param_type**

Filled Implicitly: Expected type.

UWB_DebugParams_value_t **param_value**

Input: Parameter Value

struct **phUwbDevInfo**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the UWB Device Info Parameters.

### Public Members

uint16_t **uciGenericVersion**

UCI generic version

uint8_t **macMajorVersion**

Mac Major version

uint8_t **macMinorMaintenanceVersion**

Mac Minor version

uint8_t **phyMajorVersion**
Phy Major version

uint8_t **phyMinorMaintenanceVersion**
Phy Minor version

uint16_t **uciTestVersion**
UCI test version

uint8_t **devName**[UCI_EXT_PARAM_ID_DEVICE_NAME_LEN]
Device Name

uint8_t **fwMajor**
Fw Major Version

uint8_t **fwMinor**
Fw Minor Version

uint8_t **fwPatchVersion**
Fw Patch Version

uint8_t **mwMajor**
MW Major Version

uint8_t **mwMinor**
MW Minor Version

uint8_t **devMinor**
Device Minor Version

uint8_t **devMajor**
Device Major Version

uint8_t **serialNo**[UCI_EXT_PARAM_ID_SERIAL_NUMBER_LEN]
Serial No

uint8_t **dspMajor**
DSP Fw Major Version

uint8_t **dspMinor**
DSP Fw Minor Version

uint8_t **dspPatchVersion**
DSP Fw Patch Version

uint8_t **bbMajor**
Fw Major Version

uint8_t **bbMinor**
Fw Minor Version

uint8_t **bbPatchVersion**
Fw Patch Version

uint8_t **cccVersion**[UCI_EXT_PARAM_ID_CCC_VERSION_LEN]
Fw Patch Version

uint8_t **devNameLen**
Device Name length

uint8_t **fwRc**
Fw Rc Version

uint8_t **nxpUciMajor**
NXP UCI Major Version

uint8_t **nxpUciMinor**
NXP UCI Minor Version

uint8_t **nxpUciPatch**
NXP UCI Patch Version

uint8_t **nxpChipId**[16]
NXP Chip Id

uint8_t **maxPpmValue**
Max PPM Value

int16_t **txPowerValue**[2]

TX Power Value

uint8_t **mwRc**

Mw Rc Version

uint8_t **uciGenericMajor**

NXP FIRA UCI generic major version

uint8_t **uciGenericMinorMaintenanceVersion**

NXP FIRA UCI generic minor version

uint8_t **uciGenericPatch**

NXP FIRA UCI generic patch version

uint8_t **uciTestMajor**

NXP FIRA UCI test major version

uint8_t **uciTestMinor**

NXP FIRA UCI test minor version

uint8_t **uciTestPatch**

NXP FIRA UCI test patch version

uint8_t **fwBootMode**

Fw Boot Mode

struct **phCalibRespStatus**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the calibration command response/notification.

### Public Members

uint8_t **status**

Status

uint8_t **rfu**

One byte RFU in case of Do_calib command. Calibration state in Get_calib command.

uint8_t **calibValueOut**[MAX_UCI_PACKET_SIZE]

Calibration value out

uint16_t **length**

Calibration value length

*eCalibState* **calibState**

Calibration State in case of get_calib command and not used in case of do_calib command

uint8_t **inRxAntennaPair**

Calibration antenna pair only for aoa antennae pdoa calib

struct **phGenerateTagRespStatus**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the Generate Tag command response/notification.

### Public Members

uint8_t **status**

Status

uint8_t **cmactag**[0x10U]

CMAC Tag

struct **phBlinkRespStatus**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the blink command response/notification.

### Public Members

uint8_t **repetition_count_status**

repetition count status

struct **phSeDoBindStatus**

*#include <UwbApi_Types_Proprietary.h>* UWBD Type for SE_DO_BIND Notification.

### Public Members

uint8_t **status**

Binding status 0x00: Not Bound, 0x01: Bound Unlocked, 0x02: Bound Locked, 0x03: Unknown ( if any error occurred during getting binding state from SE)

uint8_t **count_remaining**

Remaining Binding Count

uint8_t **binding_state**

Binding state

uint16_t **se_instruction_code**

command for which SE communication is failed (Itshall indicate last APDU while binding procedure)

uint16_t **se_error_status**

status codes (SW1 SW2)

struct **phSeGetBindingStatus**

*#include <UwbApi_Types_Proprietary.h>* UWBD Type for SE_GET_BINDING_STATUS Notification.

### Public Members

uint8_t **status**

Binding status 0x00: Not Bound, 0x01: Bound Unlocked, 0x02: Bound Locked, 0x03: Unknown ( if any error occurred during getting binding state from SE)

uint8_t **se_binding_count**

Remaining binding count in SE

uint8_t **uwbd_binding_count**

Remaining binding count in uwb device

uint16_t **se_instruction_code**

command for which SE communication is failed (Itshall indicate last APDU while binding procedure)

uint16_t **se_error_status**

status codes (SW1 SW2)

struct **SeConnectivityStatus**

*#include <UwbApi_Types_Proprietary.h>* UWBD Type for UWB_ESE_CONNECTIVITY_CMD.

### Public Members

uint8_t **status**

0x00 : Success 0x01 : SE Error 0x02 : Time-out 0x03 : I2C interface error between UWB and eSE 0x04 : No Applet in eSE 0x74 : APDU command is rejected by eSE 0x75 : Authentication to eSE failed 0x76 : I2C write fail 0x77 : I2C Read fail with IRQ low 0x78 : I2C Read fail with IRQ high 0x79 : I2C timeout 0x7A : I2C write time out with IRQ high

uint16_t **se_instruction_code**

command for which SE communication is failed (Itshall indicate last APDU while binding procedure)

uint16_t **se_error_status**

status codes (SW1 SW2)

struct **phSessionHandleList_t**

*#include <UwbApi_Types_Proprietary.h>* UWBD Type for URSK_DELETION_REQ Notification.

### Public Members

uint32_t **sessionHandle**

Session Handle

uint8_t **status**

Status

struct **phUrskDeletionRequestStatus_t**

*#include <UwbApi_Types_Proprietary.h>* UWBD Type for URSK_DELETION_REQ Notification.

### Public Members

uint8_t **status**

Status

uint8_t **noOfSessionHandles**

No of Session Handles

*phSessionHandleList_t* ***sessionHandleList**

Session Handle list

struct **hSeGetBindingCount**

*#include <UwbApi_Types_Proprietary.h>* UWBD Type for SE_GET_BINDING_COUNT_RSP.

### Public Members

uint8_t **bindingStatus**
Binding Status

uint8_t **uwbdBindingCount**
Remaining binding count in uwb device

uint8_t **seBindingCount**
Remaining binding count in SE.

> **Warning:** for SR150, this field has to be 0.

struct **phRframeData**
*#include <UwbApi_Types_Proprietary.h>* Structure lists out the additional rframe ranging notification information.

### Public Members

uint16_t **dataLength**
Data Length

uint8_t **data**[(sizeof(phUwbRframeLogNtf_t) * 2 * MAX_NUM_RESPONDERS)]
Data

struct **phDebugData**
*#include <UwbApi_Types_Proprietary.h>* Structure lists out the debug notification information.

### Public Members

uint16_t **dataLength**
Data Length

uint8_t **data**[MAX_DEBUG_NTF_SIZE]
Data

struct **phSchedStatusNtfData**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the scheduler status notification information.

### Public Members

uint16_t **dataLength**

Data Length

uint8_t **data**[MAX_UCI_PACKET_SIZE]

Data

struct **phSeTestLoopData**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the SE test loop information.

### Public Members

uint8_t **status**

Status

uint16_t **loop_cnt**

No of times loop was run

uint16_t **loop_pass_count**

No of times loop successfully completed

struct **phSeCommError**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the SE Comm Error notification.

### Public Members

uint8_t **status**

Status

uint16_t **cla_ins**

T=1 command for which SE communication is failed.

uint16_t **t_eq_1_status**

T=1 status codes(SW1SW2)

struct **phPdoaTableDef**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the pdoa table define config.

### Public Members

uint8_t **calibStepSize**

The calibration table step size in degrees which indicates the step size between two consecutive points

struct **clkConfigSrc**

*#include <UwbApi_Types_Proprietary.h>* Clock config parameters.

### Public Members

uint8_t **clk_src_opt**

Clock source option

uint8_t **xtal_opt**

Crystal option for RF clock

struct **phDdfsToneConfig**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the ddfs tone config.

### Public Members

uint8_t **channel_no**

channel no

uint8_t **tx_antenna_selection**

Tx antenna selection

uint32_t **tx_ddfs_tone_0**

content of TX_DDFS_TONE_0

uint32_t **tx_ddfs_tone_1**

content of TX_DDFS_TONE_1

uint32_t **spur_duration**

spur duration

uint8_t **gainval_set**

value of GAINVAL_SET

uint8_t **ddfsgainbypass_enbl**

content of DDFSGAINBYPASS_ENBL

uint16_t **periodicity**

periodicity

struct **phAoACalibCtrlAvgThreshPdoa**

*#include <UwbApi_Types_Proprietary.h>* Structure lists out the aoa Calibration average threshold.

### Public Members

uint16_t **avg_thresh_pdoa**[NO_OF_CALIB_PAIRS]

avg threshold pdoa for all angle sweeps

struct **phAntennaDefines**

*#include <UwbApi_Types_Proprietary.h>* Structure for antenna Identifier defines.

union **phDeviceConfig**

*#include <UwbApi_Types_Proprietary.h>* Device config data.

### Public Members

uint8_t **lowPowerMode**

low power mode

uint16_t **dpdEntryTimeout**

DPD entry timeout

uint16_t **hpdEntryTimeout**

DPD entry timeout

uint8_t **mhrInCcm**

MHR_IN_CCM

uint8_t **ddfsToneConfig**

DDFS tone config enable

phTxPulseShapeConfig_t **txPulseShapeConfig**

Tx Telec config

uint8_t **nxpExtendedNtfConfig**

nxp extended ntf config

uint8_t **dpdWakeupSrc**

DPD wakeup src

uint8_t **wtxCountConfig**

WTX count config

uint8_t **txBaseBandConfig**

DDFS tone config enable

uint8_t **rxAntennaSelectionConfig**

Anttena selection Config

UWB_WiFiCoEx_Ftr_t **wifiCoExFtr**

WiFi CoEx Feature Confg

uint8_t **wifiCoExChannelCfg**

WiFi CoEx Channel confg

uint8_t **wifiCoexUartUserCfg**

UART based WiFi-CoEx Interface User Configuration

phDdfsToneConfig_t **ddfsToneConfig**[NO_OF_CHANNELS]

DDFS Tone confg

uint16_t **hostMaxUCIPayloadLen**

Host Max UCI Payload Len

phPdoaTableDef_t **pdoaCalibTableDef**

phAoACalibCtrlAvgThreashPdoa_t **aoaCalibThresholdPdoa**

PDoA threshold values for all the calib pairs

uint16_t **initialRxOnOffsetAbs**

Negative offset when Rx should be enabled to receive the first message of a ranging round on Controlee compared to expected reception time. Default = 100 us

uint16_t **initialRxOnOffsetRel**

Negative offset when Rx should be enabled to receive the first message of a ranging round on Controlee compared to expected reception time. Default = 100 ppm

phAntennaDefines_t **antennaDefines**

To Define/Create all antenna Identifier for RX/TX/RX Pair

phClkConfigSrc_t **clockConfigCtrl**

Clock config parameters

uint16_t **clockPresentWaitingTime**

Maximum waiting time until clock is present struct **phActiveRoundsConfig**

*#include <UwbApi_Types_Proprietary.h>* Structure for storing Active round config Context.

### Public Members

uint8_t **roundIndex**

Active Round Index

uint8_t **rangingRole**

Device role within the given round index. See *UWB_DeviceRole_t*

uint8_t **noofResponders**

Number M of Responder MAC Addresses, Possible values are between 1 to 8.

uint8_t ***responderMacAddressList**

Responder MAC Address List for the specified ranging round as Initiator DT-Anchor.

uint8_t **responderSlotScheduling**

Responder slot presence Possible values are: 0x00: implicit scheduling, i.e., responder slots are not present; 0x01: responder slots are present; therefore, M octets shall follow, specifying the assigned slot for each Responder DT-Anchor. 0x02-0xFF: RFU

uint8_t ***responderSlots**

Responder slot index assigned for responder transmissions

struct **phNotActivatedRounds**

*#include <UwbApi_Types_Proprietary.h>* Structure for storing active round config rsp data in case of *UWBAPI_STATUS_ERROR_ROUND_INDEX_NOT_ACTIVATED*.

### Public Members

uint8_t **noOfIndex**

No of index

uint8_t **indexList**[MAX_NO_OF_ACTIVE_RANGING_ROUND]

Index list

struct **phFwCrashLogInfo**

*#include <UwbApi_Types_Proprietary.h>* Structure for storing Firmware Crash Info, allocate pLog buffer with max response.

length value and update the logLen filed

struct **phSessionSetLocZone**

*#include <UwbApi_Types_Proprietary.h>* Structure for storing fields of Session Set Localization Zone Command.

### Public Members

uint32_t **setLocZone_SessionHandle**

Session Handle of the Session for which localization zone is estimated.

uint16_t **setLocZone_RangingBlockIndex**

Ranging Block Index for which localization zone is estimated.

uint8_t **setLocZone_LocZone**

The estimated localization zone. eLocalizationZone_t

## 6.6 UWB Functional APIs (SR150 Specific)

*group* **uwb_apis_sr150**

Various SR150 Specific APIs.

### Functions

*tUWBAPI_STATUS* **UwbApi_ReadOtpCmd**(eOtpParam_Type_t paramType, uint8_t *pInfo, uint8_t *infoLength)

Read the Information from OTP based on param type See :cpp:type:`eOtpParam_Type_t` This api can be used with both Factory and Mainline Firmware

**Parameters**

- **paramType** – **[in]** parameter to write into otp See :cpp:type:`eOtpParam_Type_t`
- **infoLength** – **[inout]** Size of pInfo buffer, number of bytes expected
- **pInfo** – **[out]** Otp Info, the size of this buffer shall be equal to infoLength

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if the operation timed out
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_ConfigureData_iOS**(uint8_t *pShareableData, uint16_t ShareableDataLength, phUwbProfileInfo_t *pProfileInfo, uint8_t noOfVendorAppParams, const UWB_VendorAppParams_List_t *VendorAppParams_List, uint8_t noOfDebugParams, const UWB_DebugParams_List_t *DebugParams_List)

Prepare sharable Configuration Data.

**Parameters**

- **pShareableData** – **[in]** : sharable data which contain all information.
- **ShareableDataLength** – **[in]** : Size of sharable data
- **pProfileInfo** – **[inout]** : contains profile information.
- **noOfVendorAppParams** – **[in]** : number of VendorAppParams.
- **VendorAppParams_List** – **[in]** : List of VendorAppParams to be set.

- **noOfDebugParams** – **[in]** : number of DebugParams.
- **DebugParams_List** – **[in]** : List of Debug params to be set.

**Return values**

- **UWBAPI_STATUS_OK** – - on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – - if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – - if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – - otherwise

*tUWBAPI_STATUS* **UwbApi_ConfigureData_Android**(uint8_t *pUwbPhoneConfigData, uint16_t UwbPhoneConfigDataLen, phUwbProfileInfo_t *pProfileInfo, uint8_t noOfVendorAppParams, const UWB_VendorAppParams_List_t *VendorAppParams_List, uint8_t noOfDebugParams, const UWB_DebugParams_List_t *DebugParams_List)

Set phone uwb configuration data.

**Note:** There are 3 configurations supported namely CONFIG_ID_1, CONFIG_ID_2 and CONFIG_ID_3. The API sets/configures the respective configuration based on the value of config_id from Android jetpack.

**Parameters**

- **pUwbPhoneConfigData** – **[in]** : UwbPhoneConfigData_t data which contain all information.
- **UwbPhoneConfigDataLen** – **[in]** : Size of phone configuration data
- **pProfileInfo** – **[inout]** : contains profile information
- **noOfVendorAppParams** – **[in]** : number of VendorAppParams.
- **VendorAppParams_List** – **[in]** : List of VendorAppParams to be set.
- **noOfDebugParams** – **[in]** : number of DebugParams.
- **DebugParams_List** – **[in]** : List of Debug params to be set.

**Return values**

- **UWBAPI_STATUS_OK** – - on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – - if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – - if invalid parameters are passed
- **UWBAPI_STATUS_FAILED** – - otherwise

*tUWBAPI_STATUS* **UwbApi_SessionSetLocZone**(phSessionSetLocZone_t *pSetLocZone)

Command to set the localization Zone information which shall be sent in Final Data 2 message in the following Ranging Blocks.

**Note:** This command is only applicable to controlee.

**Parameters**

**psetLocZone** – **[in]** : Session set localization zone information set from Application.

**Return values**

- **UWBAPI_STATUS_OK** – - on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – - if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – - if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – - if command is timeout
- **UWBAPI_STATUS_FAILED** – - otherwise

## 6.7 UWB Factory test APIs (SR100/SR150 Specific)

*group* **uwb_factorytest**

These APIs are only applicable in factory mode.

APIs for factory mode test

## Functions

*tUWBAPI_STATUS* **UwbApi_FactoryInit**(tUwbApi_AppCallback *pCallback)

Initialize the UWB Middleware stack with Factory Firmware.

**Parameters**

**pCallback** – **[in]** Pointer to *tUwbApi_AppCallback* (Callback function to receive notifications at application layer.)

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_RecoverFactoryUWBS**()

API to recover from Factory Firmware crash, cmd timeout.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_FAILED** – otherwise
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout

*tUWBAPI_STATUS* **UwbApi_ConfigureAuthTagOptions**(uint8_t deviceTag, uint8_t modelTag, uint16_t labelValue)

API to configure the auth tag options. Only applicable in Factory Firmware.

**Parameters**

- **deviceTag** – **[in]** device Tag 0x0 or 0xFF signifies that none of the calibration parameters are integrity protected by “Device Specific” Tag. Any other value signifies that the calibration parameters are integrity protected by “Device specific tag.
- **modelTag** – **[in]** model Tag 0x0 or 0xFF signifies that none of the calibration parameters are integrity protected by “Model Specific” Tag. Any other value signifies that Calibration parameters are integrity protected by “Model specific tag.
- **labelValue** – **[in]** label Value This value must be different for each customer and also different for each model sold to same customer. NXP is responsible for choosing the customer specific label and providing to customer to be used in their production lines.

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_ConfigureAuthTagVersion**(uint16_t tagVersion)

API to configure the auth tag version. Only applicable in Factory Firmware.

**Parameters**

**tagVersion** – **[in]** Tag Version

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_GenerateTag**(uint8_t tagOption, phGenerateTagRespStatus_t *pCmacTagResp)

Generate Tag for the Calibration Parameters. Only applicable in factory firmware.

**Parameters**

- **tagOption** – **[in]** Tag Option indicating Device/Model Specific tag
- **pCmacTagResp** – **[out]** Pointer to phGenerateTagRespStatus_t

**Return values**

- **UWBAPI_STATUS_OK** – on success
- **UWBAPI_STATUS_NOT_INITIALIZED** – if UWB stack is not initialized
- **UWBAPI_STATUS_INVALID_PARAM** – if invalid parameters are passed
- **UWBAPI_STATUS_TIMEOUT** – if command is timeout
- **UWBAPI_STATUS_BUFFER_OVERFLOW** – if response length is more than expected response size
- **UWBAPI_STATUS_FAILED** – otherwise

*tUWBAPI_STATUS* **UwbApi_WriteOtpCmd**(eOtpParam_Type_t paramType, uint8_t *pInfo, uint8_t infoLength)

Write the Information to OTP based on param type See :cpp:type:`eOtpParam_Type_t` This api can only be used with Factory Firmware.

**Parameters**

- **`paramType`** – **[in]** parameter to write into otp See :cpp:type:`eOtpParam_Type_t`
- **`pInfo`** – **[in]** Info to write
- **`infoLength`** – **[in]** Info Length

**Return values**

- **`UWBAPI_STATUS_OK`** – on success
- **`UWBAPI_STATUS_NOT_INITIALIZED`** – if UWB stack is not initialized
- **`UWBAPI_STATUS_INVALID_PARAM`** – if invalid parameters are passed
- **`UWBAPI_STATUS_TIMEOUT`** – if the operation timed out
- **`UWBAPI_STATUS_FAILED`** – otherwise

## 6.8 APIs for Porting

*group* **`uwb_uwbs`**

This layer depends on `uwb_bus_board.h` and `uwb_bus_interface.h`

The implementation of all these APIs would be UWBD and Transport specific. e.g. If PnP mode of transport is used, RTCSync pins of SR1XX need be bothered about.

These APIs are HANDSHAKE and Protocol Aware with the UWBS.

- It menas waiting for any PIN before any operation
- It means reading “header” first, then then reading the rest of the frame later.

### Defines

**SWITCH_PROTOCOL_SWUP**

**SWITCH_PROTOCOL_UCI**

### Functions

UWBStatus_t **uwb_uwbs_tml_init**(*uwb_uwbs_tml_ctx_t* *pCtx)

Initailize it with some sane values

**Parameters**

**pCtx** – The context

UWBStatus_t **uwb_uwbs_tml_setmode**(*uwb_uwbs_tml_ctx_t* *pCtx, *uwb_uwbs_tml_mode_t* mode)

Initailize it with some sane values

**Parameters**

- **pCtx** – The context
- **mode** – See *uwb_uwbs_tml_mode_t*

UWBStatus_t **uwb_uwbs_tml_deinit**(*uwb_uwbs_tml_ctx_t* *pCtx)

De-Iniailize the context and free up references

**Parameters**

**pCtx** – The context

*group* **uwb_uwbs_tml_ctx**

### Enums

enum **uwb_uwbs_tml_mode_t**

Mode of operation of the TML Layer

This functional mode selection simplifies handling of low level transport protocol.

*Values:*

enumerator **kUWB_UWBS_TML_MODE_UCI**

Defaut Mode

enumerator **kUWB_UWBS_TML_MODE_SWUP**

FW Download mode for SR040

enumerator **kUWB_UWBS_TML_MODE_HBCI**

FW Donwnload mode for SR1XXT

enumerator **kUWB_UWBS_TML_MODE_HDLL**

FW Download mode/protocol for SR2XXT

struct **uwb_uwbs_tml_ctx_t**

*#include <uwb_uwbs_tml_interface.h>* Context for the Transport Layer.

This allows management of data / layer information.

### Public Members

uwb_bus_board_ctx_t **busCtx**

Lower lever bus specific context SPI/PNP/SOCKET

This structure is defined by Implementation layer, and not by the common UWB Library

*uwb_uwbs_tml_mode_t* **mode**

tml interface read write mode

void ***mSyncMutex**

This mutex is used to make tml interface read and write operations mutually exclusive

*group* **uwb_uwbs_tml_data**

### Functions

UWBStatus_t **uwb_uwbs_tml_data_tx**(*uwb_uwbs_tml_ctx_t* *pCtx, uint8_t *pBuf, size_t bufLen)

Transmit a data frame

**Parameters**

- **pCtx** – The context

- **pBuf** – **[in]** The data that we want to transmit
- **bufLen** – **[in]** The data length

**Returns**
Status of Transmit

UWBStatus_t **uwb_uwbs_tml_data_rx**(*uwb_uwbs_tml_ctx_t* *pCtx, uint8_t *pBuf, size_t *pBufLen)

Receive a data frame

**Parameters**

- **pCtx** – The context
- **pBuf** – **[out]** The pointer where we copy received data
- **pBufLen** – **[inout]** Input: The max length that we can copy. Output: actual length read.

**Returns**
Status of Receive

UWBStatus_t **uwb_uwbs_tml_data_trx**(*uwb_uwbs_tml_ctx_t* *pCtx, uint8_t *pTxBuf, size_t txBufLen, uint8_t *pRxBuf, size_t *pRxBufLen)

Trans-Receive a data frame.

Transmit and receive happens at the same time for this frame.

**Parameters**

- **pCtx** – The context
- **pTxBuf** – **[in]** Buffer to be transmitted
- **txBufLen** – **[in]** transmit buffer length
- **pRxBuf** – **[out]** The pointer where we copy received data
- **pRxBufLen** – **[inout]** Input: The max length that we can copy. Output: actual length read.

**Returns**
Status of Tx/Rx

UWBStatus_t **uwb_uwbs_tml_data_trx_with_Len**(*uwb_uwbs_tml_ctx_t* *pCtx, uint8_t *pTxBuf, size_t txBufLen, uint8_t *pRxBuf, size_t rxBufLen)

Trans-Receive a data frame with known receive length.

Transmit and receive happens at the same time for this frame.

**Parameters**

- **pCtx** – The context
- **pTxBuf** – **[in]** Buffer to be transmitted
- **txBufLen** – **[in]** transmit buffer length
- **pRxBuf** – **[out]** The pointer where we copy received data
- **rxBufLen** – **[in]** No of bytes to read

**Returns**
Status of Tx/Rx

UWBStatus_t **uwb_uwbs_tml_helios_reset**(*uwb_uwbs_tml_ctx_t* *pCtx)

Transmit a data frame which resets the device.

**Parameters**
**pCtx** – The context

**Returns**
Status of Tx

UWBStatus_t **uwb_uwbs_tml_helios_hardreset**(*uwb_uwbs_tml_ctx_t* *pCtx)

Transmit a data frame which resets the device.

**Parameters**
**pCtx** – The context

**Returns**
Status of Tx

UWBStatus_t **uwb_uwbs_tml_helios_get_hdll_edl_ntf**(*uwb_uwbs_tml_ctx_t* *pCtx, uint8_t *pRxBuf, size_t *pRxBufLen)

Transmit a data frame which read hdll and edl notification.

**Parameters**

- **pCtx** – The context
- **pRxBuf** – **[out]** The pointer where we copy received data
- **pRxBufLen** – **[in]** No of bytes to read

**Returns**
Status of Tx/Rx

UWBStatus_t **uwb_uwbs_tml_reset**(*uwb_uwbs_tml_ctx_t* *pCtx)

Reset uwbs tml interface

**Parameters**
**pCtx** – The context

**Returns**
Status of reset

void **uwb_uwbs_tml_flush_read_buffer**(*uwb_uwbs_tml_ctx_t* *pCtx)

Flush tml bus read buffer

**Parameters**
**pCtx** – The context

# CHAPTER SEVEN

# APIS : SE

## 7.1 SE APDU APIs

*group* **se_apdu_apis**

SE051 UWB APDU Apis.

### Defines

**SUS_MAX_BUF_SIZE_CMD**

**SUS_MAX_BUF_SIZE_RSP**

**SUS_MAX_WRAPPED_RDS_RSP_SIZE**

### Functions

smStatus_t **SUS_API_GetData**(pSe05xSession_t session_ctx, uint8_t getDataTag, uint8_t *pOutData, size_t *pOutDataLen)

SUS_API_GetData Reads SUS applet specific data. Note: This command does not require secure messaging.

*Command to Applet*

| Field | Value | Description |
|---|---|---|
| CLA | 0x80 or 0x84 | |
| INS | 0xCA | Get Data |
| P1 | 00 | Data object tag (MSB) |
| P2 | xx | Data object tag (LSB) |
| Lc | absent | |
| Payload | absent | . |
| Le | 00 | Expecting return data. |

*R-APDU Body*

| Tag(P | Name | Leng | Format |
|---|---|---|---|
| '0040' | Applet version identifier | 3 | <Major version> \|\| <Minor version> \|\| <Sequence> |
| '0042' | Applet life cycle | 1 | '02': Personalized '03': Ready '04': Locked |
| '0043' | Remaining Factory Reset | 1 | Remaining Factory Reset counter. When equal to 0, Factory Reset is unavailable. |
| '0045' | Binding history | variable | Total binding count (2 bytes) UWB subsystem count (1 byte) UWB subsystem entries (up to 3 entries) |
| '0046' | Binding State | 3 | Format description in below table |

Binding State tag 0046 Format

| Value | Description |
|---|---|
| 1st byte | '00' - Not Bound '01' - Bound & Unlocked '02' - Bound & Locked |
| 2nd byte | remaining binding attempts. (remaining factory reset counter, when equal to 0 Factory Reset is available). |
| 3rd byte | remaining binding attempts (bound & unlocked). This value shall be set to '00' if byte 1 is equal to '02' |

*R-APDU Trailer*

| SW | Description |
|---|---|
| SW_NO_ERROR | The command is handled successfully. |

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **getDataTag** – **[in]** P2 tag for get data
- **pOutData** – **[out]** Binding state data
- **pOutDataLen** – **[out]** Binding state data length

**Returns**

The APDU Transive Status.

smStatus_t **SUS_API_InitiateBinding**(pSe05xSession_t session_ctx, uint8_t heliosLC, uint8_t brkIdentifier, uint8_t *pBindinData, size_t bindinDataLen, uint8_t *pOutData, size_t *pOutDataLen)

SUS_API_InitiateBinding.

Signals the beginning of the binding procedure. As a result of processing this command, SUS will derive SCP03 static keys to bind to the UWB subsystem.*

*Command to Applet*

| Code | Value | Description |
|---|---|---|
| CLA | 0x80 | Class byte |
| INS | 0x20 | INITIATE BINDING SUS_INS_t |
| P1 | 00 or 3F | HeliosLC byte |
| P2 | 00-FF | b1-b7: BRK identifier b8: Alternative DBRK derivation constant flag |
| Lc | #(Payload) | Payload length |
| Payload | (variale) Not TLV | RAND_HE (8 bytes) Helios ID (16 bytes) |
| Le | 0x00 | |

*R-APDU Body*

| LENGTH | content |
|---|---|
| 08 | RAND_SE |
| 16 | Secure Element unique ID |

*R-APDU Trailer*

| SW | Description |
|---|---|
| SW_NO_ERROR | The command is handled successfully. |

**Parameters**

- **session_ctx** – **[in]** The SMCOM connect context
- **heliosLC** – **[in]**
- **brkIdentifier** – **[in]**
- **pBindinData** – **[in]**
- **bindinDataLen** – **[in]**
- **pOutData** – **[out]**
- **pOutDataLen** – **[out]**

smStatus_t **SUS_API_WrapData**(pSe05xSession_t session_ctx, *se_plainRDS* *pRdsData, uint8_t *pOutData, size_t *pOutDataLen)

SUS_API_WrapData.

Transfers input data for Ranging Data Set wrapping to the Secure UWB Service using the FiRa interface and returns the response from the Secure UWB Service back to the user.

*Command to Applet*

| Code | Value | Description |
|---|---|---|
| CLA | 0x80 | Class byte |
| INS | 0xA0 | WRAP DATA |
| P1 | xx | Any value |
| P2 | xx | Any value |
| Lc | #(Payload) | Payload length |
| Payload | plain RDS TLV | data to be wrapped in the Secure UWB Service |
| Le | 0x00 | |

*R-APDU Body*

| Name | LENGTH |
|---|---|
| Wrapped Ranging Data Set | depends on C-APDU Data Field |

*R-APDU Trailer*

| SW | Description |
|---|---|
| SW_NO_ERROR | The command is handled successfully. |

**Parameters**

- **`session_ctx`** – **[in]** The SMCOM connect context
- **`pRdsData`** – **[in]** Plain RDS
- **`pOutData`** – **[out]**
- **`pOutDataLen`** – **[out]**

smStatus_t **`SUS_API_Init`**(pSe05xSession_t session_ctx, uint8_t isSusClient, uint8_t logical_channel)

SUS_API_Init Opens a logical channel and selects the SUS or SUS client applet.

**Parameters**

- **`session_ctx`** – **[in]** The session context
- **`isSusClient`** – **[in]** True or False
- **`logical_channel`** – **[in]** The logical channel to be opened.

struct **se_plainRDS**

*#include <sus_APDU.h>* SUS Client Plain RDS.

### Public Members

uint8_t ***pRangingSessionKey**

Ranging Session Key 16 or 32 bytes

uint8_t ***pRspndrRangingKey**

Responder-specific ranging key

uint16_t **proxDistance**

Proximity Distance 2 bytes

int16_t **AoA**

Angle of Arrival 2 bytes

uint8_t ***pClientData**

Client specific Data 1 - 128 bytes

uint8_t ***pTransactionId**

Transaction identifier

uint8_t ***pKeyId**

Key identifier 20 bytes

uint8_t ***pArbtData**

Aribitrary Data

uint8_t ***pAppletAid**

RFU Finalization Applet AID

uint8_t ***pSessionId**

Session ID

## 7.2 SE Functional APIs

*group* **se_apis**

SE051 UWB Wrapper Apis.

### Defines

**SE_API_ALLOW_GET_BDI**

### Typedefs

typedef *se_bindingHistory_t* ***pBindHistry_t**

typedef *se_SusSession_t* ***pSusSession_t**

### Enums

enum **firalite_status**

FiRaLite Status.

*Values:*

enumerator **FIRALITE_STATUS_TRANSACTION_COMPLETE_WITH_NO_ERRORS**

enumerator **FIRALITE_STATUS_COMMAND_PROCESSED_RETURN_TO_COUNTERPART_DEVICE**

enumerator **FIRALITE_STATUS_COMMAND_PROCESSED_RETURN_TO_HOST_APP**

enumerator **FIRALITE_STATUS_TRANSACTION_COMPLETE_WITH_ERRORS**

enum **se_status_t**

SE Bind State.

*Values:*

enumerator **SE_STATUS_OK**

enumerator **SE_STATUS_NOT_OK**

enum **se_boundStatus_t**

SE Bound status.

*Values:*

enumerator **SE_Not_Bound**

enumerator **SE_Bound_And_Unlocked**

enumerator **SE_Bound_And_Locked**

enumerator **SE_Bound_NA**

enum **se_lifeCycleStatus_t**

Applet Life Cycle.

*Values:*

enumerator **SE_None**

enumerator **SE_Personalized**

enumerator **SE_Ready**

enumerator **SE_Locked**

enum **se_applet_t**

Applet types.

*Values:*

enumerator **SE_APPLET_SUS**

enumerator **SE_APPLET_SUS_CLIENT**

enumerator **SE_APPLET_FIRALITE**

### Functions

*se_status_t* **Se_API_SetHandle**(void *se_ctx)

APIs exposed to application to access SE051 UWB Functionality.

Initializes Communication. This function sets the SE handle for communication.

**Parameters**
`se_ctx` – **[in]** connection context

**Return values**

- `SE_STATUS_OK` – The operation has completed successfully.
- `SE_STATUS_NOT_OK` – The operation has failed.

**Returns**
Status of the operation

void **Se_API_ResetHandle**(void)

DeInitialise Communication. This function clears the SE handle for communication..

*se_status_t* **Se_API_Init**(*se_applet_t* applet, uint8_t logical_channel)

Select the desired applet. The function selects the Fira applet for further communication.

**Parameters**

- `applet` – **[in]** applet to select See :cpp:type:`se_applet_t`
- `logical_channel` – **[in]** Logical channel to be used for SE communication

**Return values**

- `SE_STATUS_OK` – The operation has completed successfully.
- `SE_STATUS_NOT_OK` – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_GetBindingState**(*se_bindState_t* *pBindState)

Get SE bind state. The function gets the SE bind state.

**Parameters**
`pBindState` – **[out]** returns binding State. see :cpp:type:'*se_bindState_t*'

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_GetVersion**(uint8_t *verString, size_t *verStringLen)

Get SUS Applet Version. The function gets the version of SUS applet.

**Parameters**

- **verString** – **[out]** returns applet version string.
- **verStringLen** – **[out]** returns len of applet version string.

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_GetFiraLiteVersion**(uint8_t *verString, size_t *verStringLen, char *verDiscription, size_t *verDiscriptionLen)

Get FiraLite Applet Version. The function gets the version of FiraLite applet.

**Parameters**

- **verString** – **[out]** returns applet version string.
- **verStringLen** – **[out]** returns len of applet version string.
- **verDiscription** – **[out]** returns version description in ASCII format.
- **verDiscriptionLen** – **[out]** returns len of version description.

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_GetLifeCycle**(*se_lifeCycleStatus_t* *pLcState)

Get SUS Applet Life Cycle State. The function gets the Life cycle state.

**Parameters**
**pLcState** – **[out]** returns applet life cycle state See :cpp:type:`se_lifeCycleStatus_t`.

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_GetBindingHistory**(*pBindHistry_t* pBindHistry)

Get Binding History. The function gets the binding history of applet with UWB subsystem.

**Parameters**
**pBindHistry** – **[out]** returns binding history See :cpp:type:*se_bindingHistory_t*.

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_SendReceive**(uint8_t *pInData, size_t InDataLen, uint8_t *pOutData, size_t *pOutDataLen)

Send Receive to SE. The function is used to send and received data to SE.

**Parameters**

- **pInData** – **[in]** Input buffer
- **InDataLen** – **[in]** Length of the input in bytes
- **pOutData** – **[out]** Output buffer
- **pOutDataLen** – **[out]** Length of the output in bytes

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_InitiateBinding**(uint8_t Lc, uint8_t brk, uint8_t *pbindInData, size_t bindInDataLen, uint8_t *pbindOutData, size_t *pbindOutDataLen)

Send Initiate Binding APDU to SE . The function is used to start the binding process to SE.

**Parameters**

- **Lc** – **[in]** Lifecycle state as input
- **brk** – **[in]** brk Id as input
- **pbindInData** – **[in]** Input buffer
- **bindInDataLen** – **[in]** Length of the input in bytes
- **pbindOutData** – **[out]** Output buffer
- **pbindOutDataLen** – **[out]** Length of the output in bytes

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

*se_status_t* **Se_API_GetWrappedRDS**(*se_applet_t* applet, *se_rds_t* *pRds, uint8_t *pWrappedRds, size_t *pWrappedRdsLen)

Utility function to get Wrapped RDS from SUSClient/FiraLite In case of FiraLite Get the wrapped RDS from dispatch buffer.

**Parameters**

- **applet** – **[in]** Applet FiraLite/SUS Client
- **pRds** – **[in]** Union containing rds data for FiraLite/SUS Client
- **pWrappedRds** – **[out]** wrapped RDS as Output buffer
- **pWrappedRdsLen** – **[out]** Length of the output in bytes

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

*se_status_t* **Se_API_WrapData**(*se_plainRDS* *pRdsData, uint8_t *pWrappedRds, size_t *pWrappedRdsLen)

Send plain RDS Data to SE. The function is used to get wrapped RDS from SUS Client. *se_plainRDS* is converted into final RDS TLV payload. The maximum TLV Length of the plain RDS is allowed to 223 bytes. Please refer applet spec.

**Parameters**

- **pRdsData** – **[in]** plain RDS Data as input

- **pWrappedRds** – **[out]** wrapped RDS as Output buffer
- **pWrappedRdsLen** – **[out]** Length of the output in bytes

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

*se_status_t* **Se_API_DeInit**(void)

Close the channel and deselect the applet.

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

*se_status_t* **Se_API_SelectADF**(const *se_firalite_optsa_t* optsA, *se_firelite_oid_entry* *oid_entries, const size_t oid_entries_count, *se_firelite_selectadf_reponse_t* *pResponse)

Send SelectADF command to SE.

**Parameters**

- **optsA** – **[in]** type of crypto
- **oid_entries** – **[in]** reference to OID entries
- **oid_entries_count** – **[in]** number of OID entries
- **pResponse** – **[out]** buffer containing the response

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

*se_status_t* **Se_API_InitiateTransaction**(*se_firelite_oid_entry* *oid_entries, const size_t oid_entries_count, uint8_t *pSessionId, const size_t sessionIdLen, uint8_t *pStatus, uint8_t *pDataBuffer, size_t *pDataLen)

Send InitiateTransaction command to SE.

**Parameters**

- **oid_entries** – **[in]** reference to OID entries
- **oid_entries_count** – **[in]** number of OID entries
- **pSessionId** – **[in]** Optional reference to the UWB session id used in case of Multicast Ranging
- **sessionIdLen** – **[in]** length of the UWB session id if pSessionId is not null
- **pStatus** – **[out]** pointer to the status
- **pDataBuffer** – **[out]** buffer containing CAPDU (must be large enough to hold SE response)
- **pDataLen** – **[out]** length of the CAPDU

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

*se_status_t* **Se_API_Dispatch**(uint8_t *pDispatchData, size_t dispatchDataLen, uint8_t *pStatus, uint8_t *pDataBuffer, size_t *pDataLen, uint8_t *pEventId, uint8_t *pEventDataBuffer, size_t *pEventDataLen)

Send Dispatch command to SE.

**Parameters**

- **pDispatchData** – **[in]** pointer to the data to be dispatched
- **dispatchDataLen** – **[in]** length of the data to be dispatched
- **pStatus** – **[out]** pointer to the status
- **pDataBuffer** – **[out]** buffer containing the response data
- **pDataLen** – **[out]** length of the response data (0 if not part of response)
- **pEventId** – **[out]** pointer to the event Id
- **pEventDataBuffer** – **[out]** buffer containing the event data
- **pEventDataLen** – **[out]** length of the event data

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_RemoteGetData**(const uint8_t *pInBuf, const size_t inBufLen, uint8_t *pStatus, uint8_t *pDataBuffer, size_t *pDataLen)

Send GetData command over tunnel to SE.

**Parameters**

- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length
- **pStatus** – **[out]** pointer to the status
- **pDataBuffer** – **[out]** buffer containing the response data
- **pDataLen** – **[out]** length of the response data

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_RemotePutData**(const uint8_t *pInBuf, const size_t inBufLen, uint8_t *pStatus, uint8_t *pDataBuffer, size_t *pDataLen)

Send PutData command over tunnel to SE.

**Parameters**

- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length
- **pStatus** – **[out]** pointer to the status
- **pDataBuffer** – **[out]** buffer containing the response data
- **pDataLen** – **[out]** length of the response data

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_LocalGetData**(const uint8_t tagp1, const uint8_t tagp2, uint8_t *pDataBuffer, size_t *pDataLen)

Send GetData command to local SE.

**Parameters**

- **tagp1** – **[in]** Tag P1
- **tagp2** – **[in]** Tag P2
- **pDataBuffer** – **[out]** buffer containing the response data
- **pDataLen** – **[out]** length of the response data

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_LocalPutData**(const uint8_t *pInBuf, const size_t inBufLen)

Send PutData command to local SE.

**Parameters**

- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**
Status of the operation

*se_status_t* **Se_API_RemoteGetData_WithoutTunnel**(const uint8_t *pInBuf, const size_t inBufLen, uint8_t *pRspBuffer, size_t *pRspBufLen)

Send Get Data command to Remote SE. To be sent in plain without tunnel.

**Parameters**

- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length

- **pRspBuffer** – **[out]** buffer containing CAPDU will be sent OOB (must be large enough to hold SE response)
- **pRspBufLen** – **[out]** length of the CAPDU

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

*se_status_t* **Se_API_RemotePutData_WithoutTunnel**(const uint8_t *pInBuf, const size_t inBufLen, uint8_t *pRspBuffer, size_t *pRspBufLen)

Send Put Data command to Remote SE. To be sent in plain without tunnel.

**Parameters**

- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length
- **pRspBuffer** – **[out]** buffer containing CAPDU will be sent OOB (must be large enough to hold SE response)
- **pRspBufLen** – **[out]** length of the CAPDU

**Return values**

- **SE_STATUS_OK** – The operation has completed successfully.
- **SE_STATUS_NOT_OK** – The operation has failed.

**Returns**

Status of the operation

struct **se_bindState_t**

*#include <SE_Wrapper.h>* SE Binding State.

struct **firaLiteWrappedRds_t**

*#include <SE_Wrapper.h>* FiraLite Wrapped RDS, command buffer from dispatch.

union **se_rds_t**

*#include <SE_Wrapper.h>* RDS structure to use depending on applet SUSClient/FiraLite.

### Public Members

*firaLiteWrappedRds_t* ***pFlWrappedRds**

*se_plainRDS* ***pSusPlaindRds**

struct **uwbSystmHistory_t**

*#include <SE_Wrapper.h>* UWB subystem entry.

struct **se_bindingHistory_t**

*#include <SE_Wrapper.h>* UWB-SE Binding History Context.

struct **se_SusSession_t**

*#include <SE_Wrapper.h>* SE Session Context.

## 7.3 SE FiRaLite APDU APIs

*group* **se_firalite_apdu_apis**

SE051 UWB FiRaLite APDU Apis.

### Enums

enum **se_firalite_route_target_t**

FiraLite route target

*Values:*

enumerator **se_firalite_route_NA**

Invalid

enumerator **se_firalite_route_firaLiteApplet**

Route to App in FiraLite Applet

enumerator **se_firalite_route_host**

Route to App in host

enum **se_firalite_optsa_t**

FiraLite optsA crypto parameter

*Values:*

enumerator **se_firalite_optsa_symm_crypto**

Support symmetric crypto

## Functions

smStatus_t **se_FiRaLite_API_Select**(pSe05xSession_t session_ctx, uint8_t logical_channel)

se_FiRaLite_API_Select This selects FiRalite applet on channel 1.

*Selects FiRalite Applet*

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **logical_channel** – **[in]** The logical channel to be opened.

**Returns**

The select Status.

smStatus_t **se_FiRaLite_API_SelectADF**(pSe05xSession_t session_ctx, const *se_firalite_optsa_t* optsA, *se_firelite_oid_entry* *oid_entries, const size_t oid_entries_count, uint8_t *pRspData, size_t *pRspDataLen)

se_FiRaLite_API_SelectADF The SELECT ADF command is used to select and route subsequent commands to a specific ADF

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **optsA** – **[in]** Crypto Type
- **oid_entries** – **[in]** reference to OID entries
- **oid_entries_count** – **[in]** number of OID entries
- **pRspData** – **[out]** Response data
- **pRspDataLen** – **[out]** Response data Length

**Returns**

The APDU Transceive Status.

smStatus_t **se_FiRaLite_API_InitiateTransaction**(pSe05xSession_t session_ctx, *se_firelite_oid_entry* *oid_entries, const size_t oid_entries_count, uint8_t *pSessionId, const size_t sessionIdLen, uint8_t *pRspData, size_t *pRspDataLen)

se_FiRaLite_API_InitiateTransaction The INITIATE TRANSACTION command is issued to begin a transaction with the counterpart device

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **oid_entries** – **[in]** reference to OID entries
- **oid_entries_count** – **[in]** number of OID entries
- **pSessionId** – **[in]** reference to the UWB session id
- **sessionIdLen** – **[in]** length of the UWB session id
- **pRspData** – **[out]** Response data
- **pRspDataLen** – **[out]** Response data Length

**Returns**

The APDU Transceive Status.

smStatus_t **se_FiRaLite_API_Dispatch**(pSe05xSession_t session_ctx, uint8_t *pCmdData, const size_t cmdDataLen, uint8_t *pRspData, size_t *pRspDataLen)

se_FiRaLite_API_Dispatch The DISPATCH command is used to transfer command/responses to the FiRaLite Applet. Any command received over an out-of-band channel, shall be passed on to the FiRaLite applet using the DISPATCH command

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **pCmdData** – **[in]** command Buffer
- **cmdDataLen** – **[in]** command Buffer length
- **pRspData** – **[out]** Response data
- **pRspDataLen** – **[out]** Response data Length

**Returns**

The APDU Transceive Status.

smStatus_t **se_FiRaLite_API_Tunnel**(pSe05xSession_t session_ctx, uint8_t *pCmdData, const size_t cmdDataLen, uint8_t *pRspData, size_t *pRspDataLen)

se_FiRaLite_API_Tunnel The TUNNEL command is used to wrap application specific data that is exchanged over an OoB channel.

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **pCmdData** – **[in]** command Buffer
- **cmdDataLen** – **[in]** command Buffer length
- **pRspData** – **[out]** Response data
- **pRspDataLen** – **[out]** Response data Length

**Returns**

The APDU Transceive Status.

smStatus_t **se_FiRaLite_API_RemoteGetData**(const uint8_t *pInBuf, const size_t inBufLen, uint8_t *pRspData, size_t *pRspDataLen)

se_FiRaLite_API_RemoteGetData GET DATA is used to read one or more data elements from the currently selected ADF data structure from remote device.

**Parameters**

- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length
- **pRspData** – **[out]** Response data
- **pRspDataLen** – **[out]** Response data Length

**Returns**

The operation status.

smStatus_t **se_FiRaLite_API_RemotePutData**(const uint8_t *pInBuf, const size_t inBufLen, uint8_t *pRspData, size_t *pRspDataLen)

se_FiRaLite_API_RemotePutData The PUT DATA command is used to write one data element into the currently selected ADF data for remote device.

**Parameters**

- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length
- **pRspData** – **[out]** Response data

- **pRspDataLen** – **[out]** Response data Length

**Returns**

The operation status.

smStatus_t **se_FiRaLite_API_LocalGetData**(pSe05xSession_t session_ctx, uint8_t tagMSB, uint8_t tagLSB, uint8_t *pRspData, size_t *pRspDataLen)

se_FiRaLite_API_LocalGetData The GET DATA command is used to get data from FiraLite Applet for local device. It cannot be used to read ADF data.

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **tagMSB** – **[in]** P1 Value
- **tagLSB** – **[in]** P2 Value
- **pRspData** – **[out]** Response data
- **pRspDataLen** – **[out]** Response data Length

**Returns**

The operation status.

smStatus_t **se_FiRaLite_API_LocalPutData**(pSe05xSession_t session_ctx, const uint8_t *pInBuf, const size_t inBufLen)

se_FiRaLite_API_LocalPutData The PUT DATA command is used to put data structure for currently selected ADF for local SE.

**Parameters**

- **session_ctx** – **[in]** The smcom session context
- **pInBuf** – **[in]** command Buffer
- **inBufLen** – **[in]** command Buffer length

**Returns**

The operation status.

struct **se_firelite_oid_entry**

*#include <se_FiRaLite_API.h>* FiraLite OID descriptor

### Public Members

uint8_t ***pOIDData**

Reference to OID data

size_t **OIDDataLen**

length of OID data

struct **se_firelite_selectadf_reponse_t**

*#include <se_FiRaLite_API.h>* FiraLite SelectADF response

### Public Members

uint8_t **pAid**[16]

AID data

size_t **aidLen**

length of AID

bool **is_privacy_enabled**

indicates whether proprietary info is used

uint8_t **pPropInfo**[32]

proprietary info

struct **se_firalite_secure_ranging_info_t**

*#include <se_FiRaLite_API.h>* Firalite secure ranging info

### Public Members

uint8_t ***pUWBSessionKey**

Firalite session key info

uint8_t **UWBSessionKeyLen**

Firalite session key length

uint8_t ***pUWBSubSessionKey**

Firalite sub session key info

uint8_t **UWBSubSessionKeyLen**

Firalite sub session key length

uint32_t **subSessionId**

Firalite session key Id

uint8_t **makeRDSAvailable**

Firalite rds flag

struct **se_firalite_cmd_routing_info_t**

*#include <se_FiRaLite_API.h>* Firalite command routing info

### Public Members

*se_firalite_route_target_t* **target**

Firalite command route target

uint8_t ***pCmd**

Firalite command buffer

uint8_t **CmdLen**

Firalite command buffer length

CHAPTER

# EIGHT

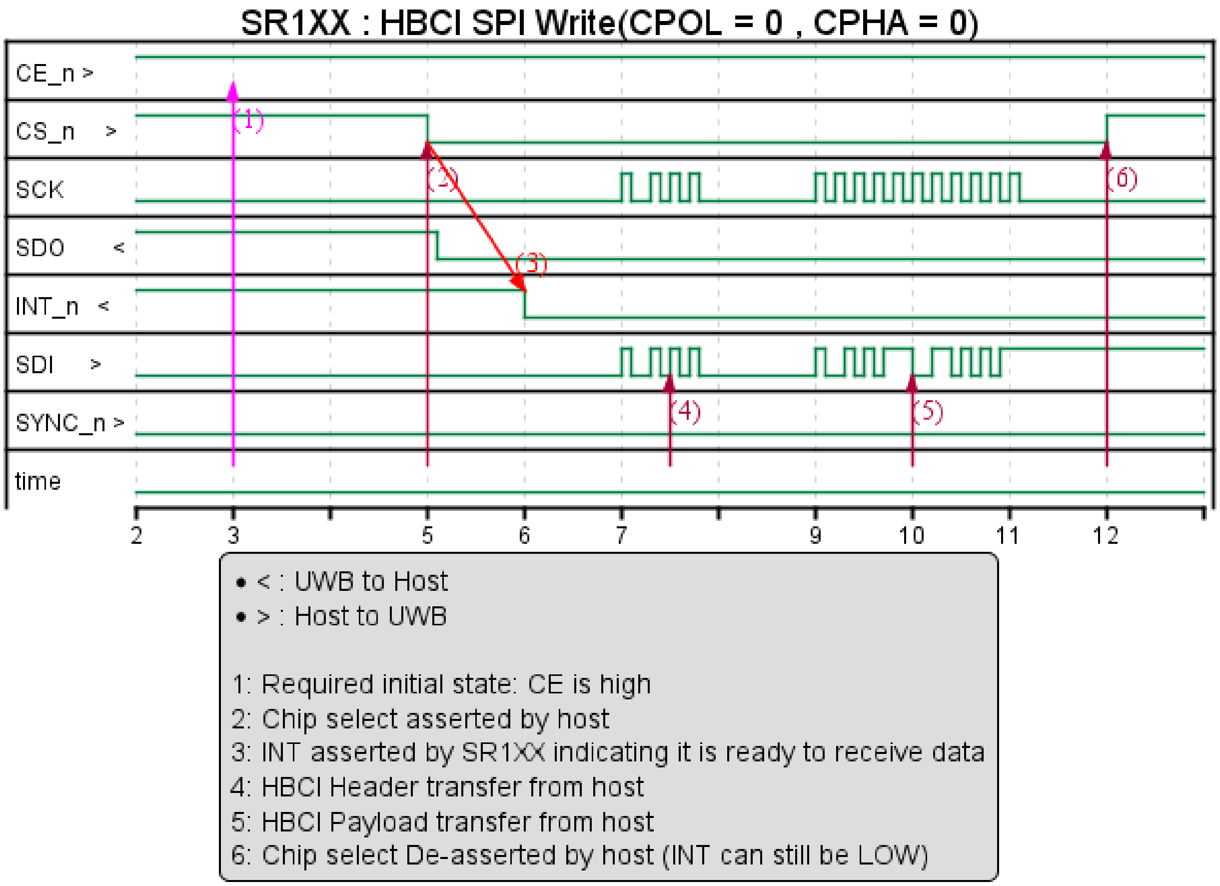
# SR1XX

## 8.1 SR1XX SPI communication

This section shows SPI level handshake and communication with SR1XX under various scenarios. It can be used to understand how the underlying system communication is handled.

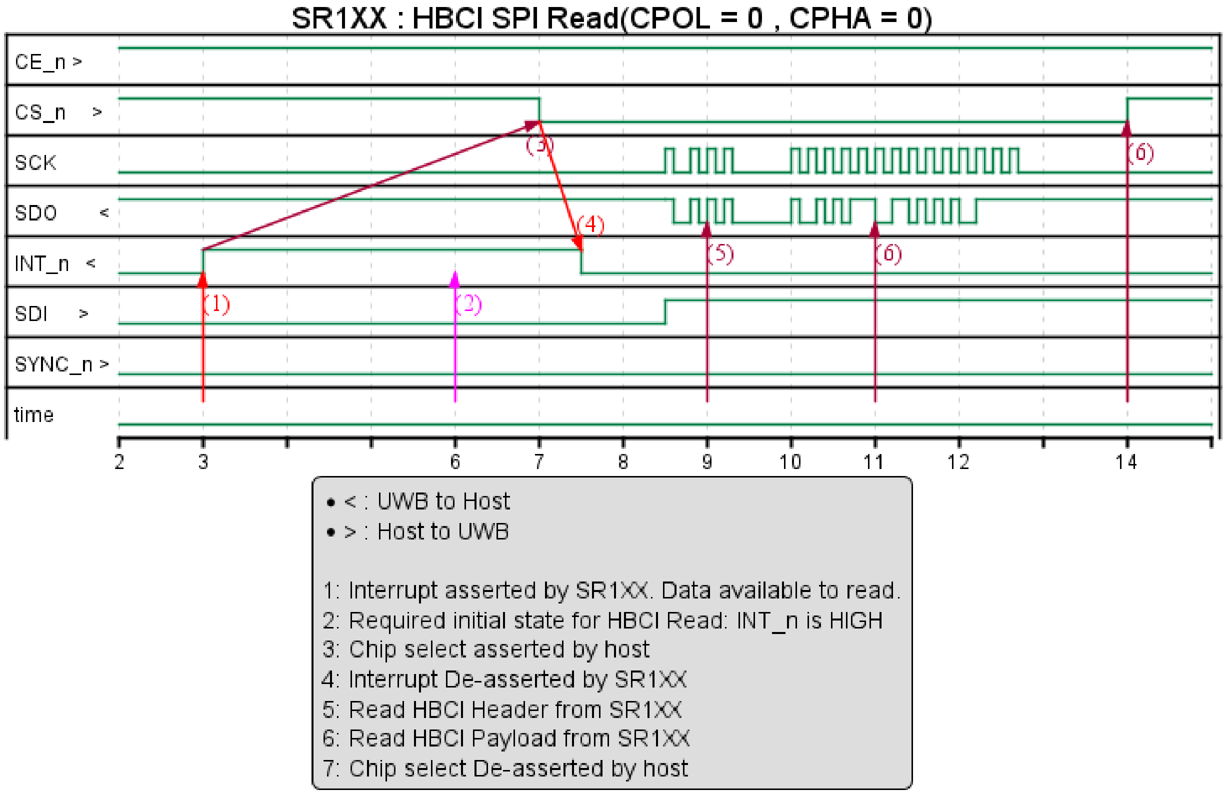
### 8.1.1 HBCI SPI Write

This timing diagram shows HBCI Write handshake and sequence to SR1XX.



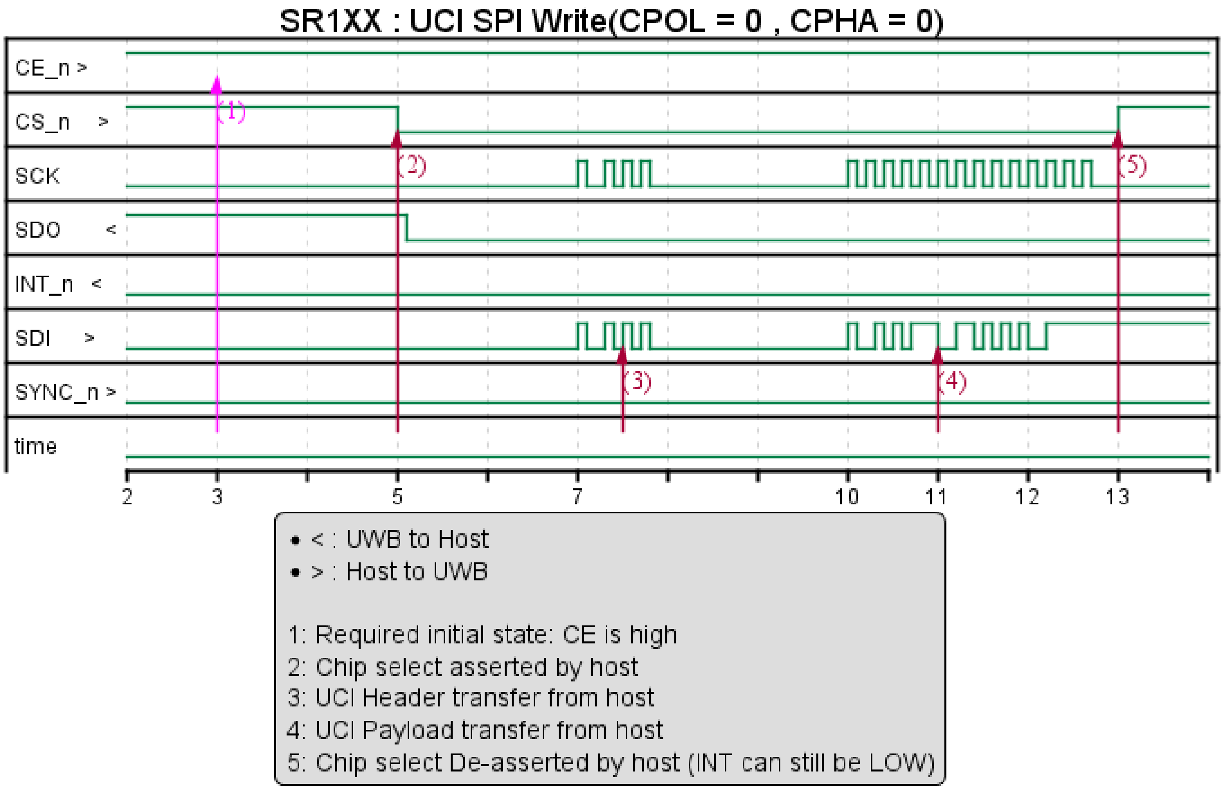

## 8.1.2 HBCI SPI Read

This timing diagram shows HBCI Read handshake and sequence to SR1XX.



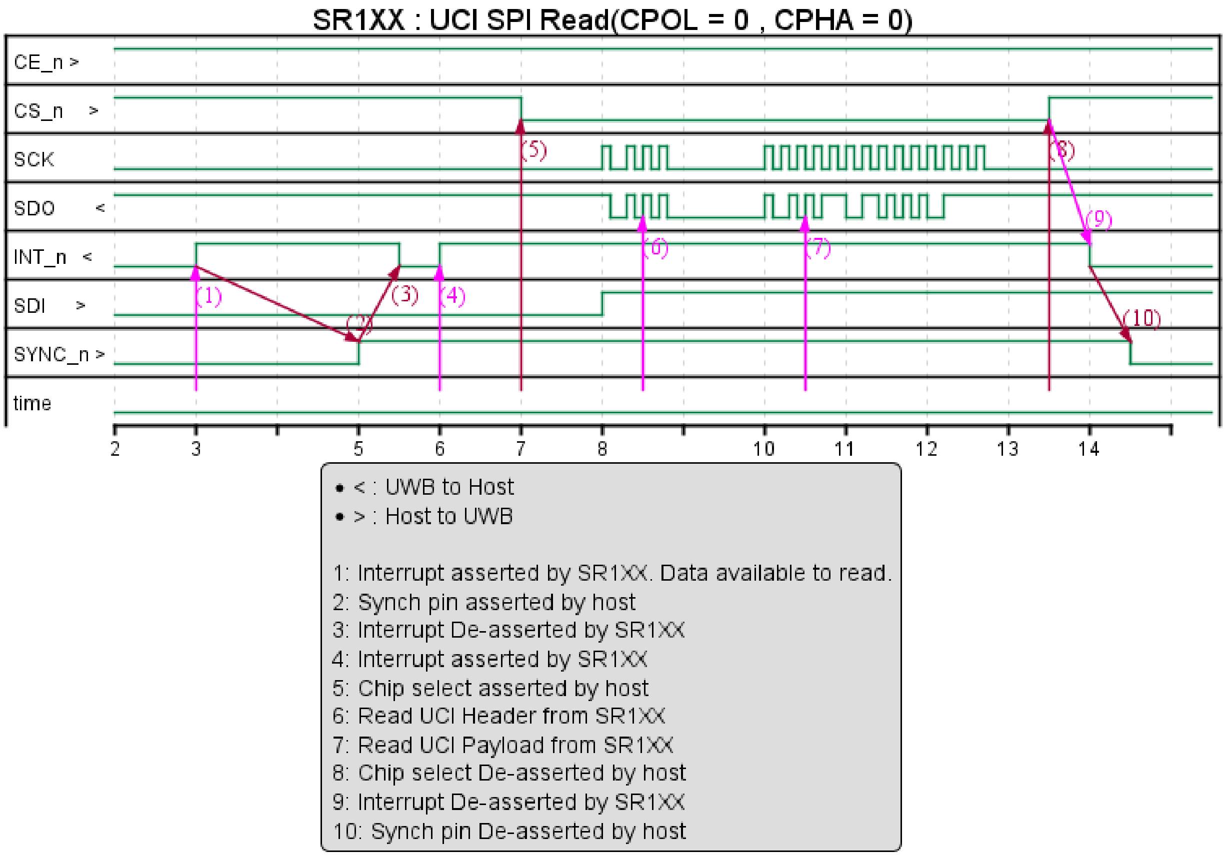

## 8.1.3 SPI Write

This timing diagram shows UCI Write handshake and sequence to SR1XX.




### 8.1.4 SPI Read

This timing diagram shows UCI Read handshake and sequence from SR1XX.

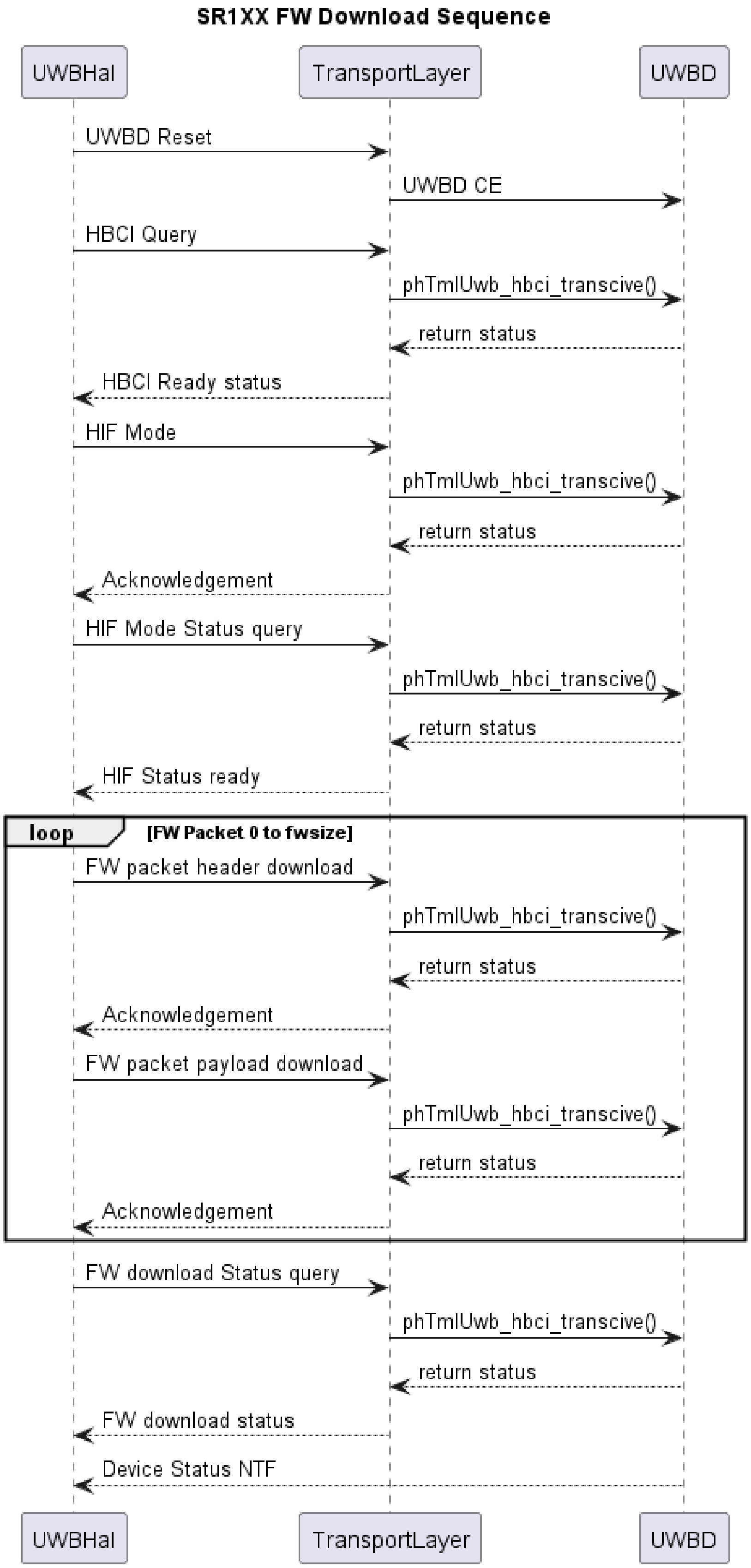



## 8.2 HBCI

SR1XX has a HBCI (Host BootROM Command Interface) mode, based on HBCI protocol. By default the IC bootup in HBCI mode. As SR1XX is RAM based one need to Download FW every boot. so to download FW HBCI host interface protocol is used.

### 8.2.1 SR1XX FW download Flow

Given below is the SR1XX FW download sequence diagram.


SR1XX FW Download Sequence
UWBHal
TransportLayer
UWBD
UWBD Reset
UWBD CE
HBCI Query
phTmlUwb_hbci_transcive()
return status
HBCI Ready status
HIF Mode
phTmlUwb_hbci_transcive()
return status
Acknowledgement
HIF Mode Status query
phTmlUwb_hbci_transcive()
return status
HIF Status ready
loop
[FW Packet 0 to fwsize]
FW packet header download
phTmlUwb_hbci_transcive()
return status
Acknowledgement
FW packet payload download
phTmlUwb_hbci_transcive()
return status
Acknowledgement
FW download Status query
phTmlUwb_hbci_transcive()
return status
FW download status
Device Status NTF
UWBHal
TransportLayer
UWBD

CHAPTER

NINE

# APPENDIX

## 9.1 Peer-to-Peer ranging

Peer-To-Peer Ranging Use Case works in the following flow:

1) Initialize the UWB Stack, call *UwbApi_Init()*.

2) Create a session for Ranging with a 4-byte unique session ID and session type `Ranging` (0x00) using *UwbApi_SessionInit()*.

   **Note:** Same session ID should be used by both Initiator and responder.

   All default session configs are applied at the time of session creation.

3) Set mandatory ranging params with `UwbApi_SetRangingParam()`

   - SR100T is configured as Controlee Responder
   - SR040 is configured as Controller Initiator
   - Number of anchors are set to 1 for unicast session
   - Source MAC address (2 byte SR100T dev address)
   - Dest MAC address (2 byte R4 address)

   **Note:** While setting mandatory params Source address of SR100T must be set as Destination address for R4 and vice versa.

4) Set static STS using *UwbApi_SetStaticSts()* (2-byte VendorID and 6-byte STS IV). Same values must be set for both SR100T and SR040.

5) Apply the application specific ranging configurations by calling *UwbApi_SetAppConfig()*. For reference, the following configurations can be set

   - `RANGING_SLOT_LEN = 2000`

- `RANGING_DURATION = 0x60`
- `STS_INDEX = 0`
- `PREAMBLE_ID = 0x0A`
- `SFD_ID = 0x00`
- `STS_CONFIG = 0x00`
- `MAC_FCS_TYPE = 0x00`
- `PREAMBLE_DURATION = 1`
- `MAC_CFG = 3`
- `PPDU_CONFIG = 0x03`
- `RNG_DATA_NTF = 1`
- `RANGING_ROUND_HOPPING = 0`
- `RX_MODE = 0`
- `RX_ANTENNA_SELECTION = 1`
- `NUMBER_OF_STS_SEGMENTS = 1`
- `CHANNEL_ID = 0x09`
- `PSDU_DATA_RATE = 0x00`
- `RANGING_METHOD = 0x02`
- `RANGING_ROUND_CONTROL = 0x02`

For more details, refer to FIRA UCI Generic Spec.

Set same configurations for SR100T and SR040.

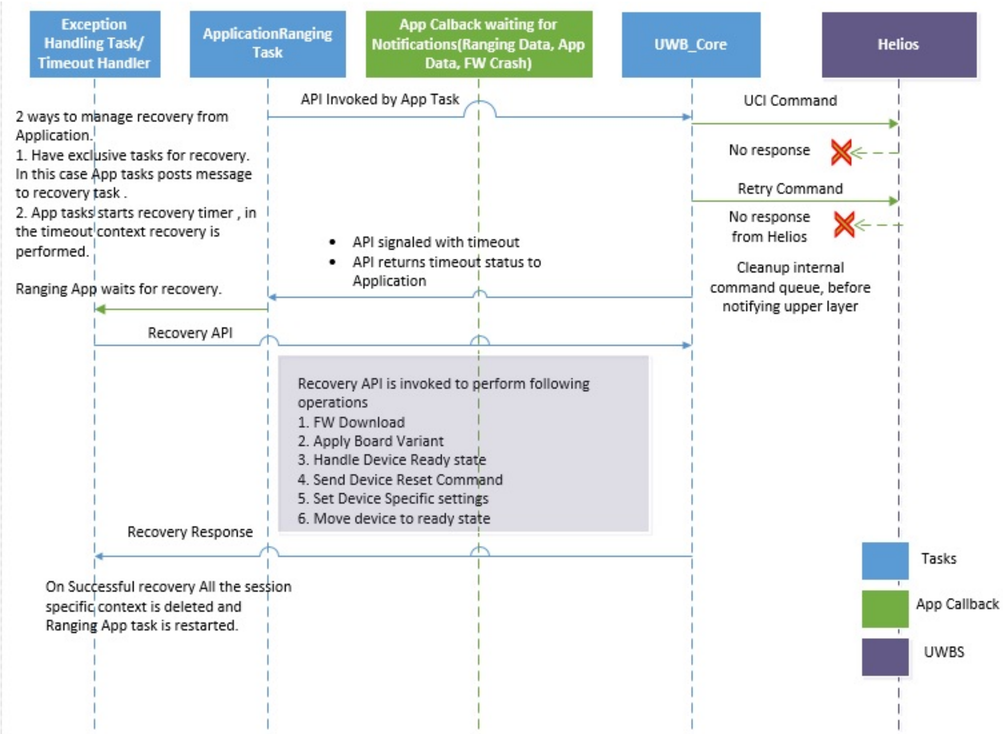
6) Start ranging by calling `UwbApi_StartRangingSession()`.
7) Stop the ranging session with `UwbApi_StopRangingSession()` and de-initialize the session by calling `UwbApi_SessionDeinit()`.

## 9.2 UCI Command timeout handling

The following process is done to handle command timeouts:

1) API is invoked from Application Task, UCI command is sent to UWBS as a part of API.
2) If response is not received command retry is attempted. If response is not received for the second time as well, UCI timeout status is notified to upper layers.
3) API gets unblocked wit status UCI timeout. Same status is returned to Application.

4) Applications post message to exception handling task or strt recovery timer.
5) Application tasks waits for recovery
6) Exception handling task/ or in the context of timeout, Recovery API is invoked. Recovery API performs certain operations are shown in the flow diagram.
7) After recovery All session state context is deleted, and application task is restarted.




## 9.3 Logging Configurations

1) Logging can be configured for each module and for all modules. All logging specific settings can be configured in the file `/uwbiot-top/libs/halimpl/inc/phNxpLogDefault.h`.
2) Configuring Logging for all modules is via **global logging level**.

   This defines the common logging level for all the modules. Global logging level overrides the local logging level.

   Example: If Global Logging level is set to `UWB_LOG_INFO_LEVEL` and local logging level is set to `UWB_LOG_DEBUG_LEVEL` then only Global Logging level `UWB_LOG_INFO_LEVEL` messages will be seen.
3) Logging levels.

There are 7 different log levels defined in the file `/uwbiot-top/libs/halimpl/inc/uwb_logging.h`:

```
#define UWB_LOG_SILENT_LEVEL    0x00    // "Silent"     level
#define UWB_LOG_ERROR_LEVEL     0x01    // "Error"      level
#define UWB_LOG_WARN_LEVEL      0x02    // "Warning"    level
#define UWB_LOG_INFO_LEVEL      0x03    // "Info"       level␣
↪[Default]
#define UWB_LOG_DEBUG_LEVEL     0x04    // "Debug"      level
#define UWB_LOG_TX_LEVEL        0x05    // "Tx"         level
#define UWB_LOG_RX_LEVEL        0x06    // "Rx"         level
```

4) Configure global level logging

   `#define UWB_GLOBAL_LOG_LEVEL UWB_LOG_INFO_LEVEL`

   To set the global logging level, use above macro defined in `phNxpLogDefault.h` file. Depending on this macro, sub modules are enabled/disabled. If global level logging is higher than module level logging, global level logging is considered. For e.g. If global logging is default level and APP module logging is debug level then debug level for APP module is not considered.

   1) To make logging level silent:

   ```
   #define UWB_GLOBAL_LOG_LEVEL UWB_LOG_SILENT_LEVEL
   ```

   2) By default logging level:

   ```
   #define UWB_GLOBAL_LOG_LEVEL UWB_LOG_INFO_LEVEL
   ```

5) Configuring Logging for a specific module is via **local logging level**.

   This defines the logging level only for a particular module. Example: To set logging level of UWB API module, local logging level of this module needs to be set by defining `UWBAPI_LOG_LEVEL` macro value to a particular level. Also Global logging level to be set higher or equal to the local logging level. To see all the "debug" messages in the UWB API module then following logging macros to be set:

```
#define UWBAPI_LOG_LEVEL     UWB_LOG_DEBUG_LEVEL
#define UWB_GLOBAL_LOG_LEVEL UWB_LOG_DEBUG_LEVEL
```

6) Enable/Disable logging of UCI commands.

   **Tx** means the UCI command transmitted to the UWB device. **Rx** means the UCI response or notification received from the UWB device.

   UCI prints can be enabled/disabled using following macro from `/uwbiot-top/libs/halimpl/inc/phNxpLogApis_TmlUwb.h`

1) To Enable the UCI LOGGING:

```
#define ENABLE_UCI_CMD_LOGGING ENABLED
```

2) To Disable the UCI LOGGING:

```
#define ENABLE_UCI_CMD_LOGGING DISABLED
```

7) Enable/Disable Logging of TML module

Following macros to be set to enable or disable the TML module logging.

1) To disable all logging in the TML module level:

```
// in file /uwbiot-top/libs/halimpl/inc/phNxpLogApis_TmlUwb.h
#define ENABLE_UCI_CMD_LOGGING DISABLED

// in file /uwbiot-top/libs/halimpl/inc/phNxpLogDefault.h
#define TMLUWB_LOG_LEVEL       UWB_LOG_SILENT_LEVEL

// in file /uwbiot-top/libs/halimpl/inc/phNxpLogDefault.h
#define UWB_GLOBAL_LOG_LEVEL   UWB_LOG_SILENT_LEVEL
```

2) To enable all logging in the TML module level:

```
// in file /uwbiot-top/libs/halimpl/inc/phNxpLogApis_TmlUwb.h
#define ENABLE_UCI_CMD_LOGGING ENABLED

// in file /uwbiot-top/libs/halimpl/inc/phNxpLogDefault.h
#define TMLUWB_LOG_LEVEL       UWB_LOG_DEBUG_LEVEL

// in file /uwbiot-top/libs/halimpl/inc/phNxpLogDefault.h
#define UWB_GLOBAL_LOG_LEVEL   UWB_LOG_DEBUG_LEVEL
```

3) To enable the TML module logging to **Info** level [default]:

```
// in file /uwbiot-top/libs/halimpl/inc/phNxpLogApis_TmlUwb.h
#define ENABLE_UCI_CMD_LOGGING ENABLED

// in file /uwbiot-top/libs/halimpl/inc/phNxpLogDefault.h
#define TMLUWB_LOG_LEVEL       UWB_LOG_INFO_LEVEL

// in file /uwbiot-top/libs/halimpl/inc/phNxpLogDefault.h]
#define UWB_GLOBAL_LOG_LEVEL   UWB_LOG_INFO_LEVEL
```

8) UART Logging with Verbose log level for *RhodesV4* board

UART verbose level of logging with default baud rate will slow down the complete system. Therefore we shall use 3Mbps baud rate.

```
/* The UART to use for debug messages. */
/* when UWBIOT_LOG=Verbose, set the baudrate to 3Mbps for viewing␣
↪logs in terminal emulator(ex. Tera term)*/
#define BOARD_DEBUG_UART_TYPE kSerialPort_Uart
#if (UWBIOT_LOG_VERBOSE == 1)
#define BOARD_DEBUG_UART_BAUDRATE 3000000U //3Mbps
#pragma message("Logging Baud rate is set to 3Mbits. Please change␣
↪settings on Serial port GUI")
#else
#define BOARD_DEBUG_UART_BAUDRATE 3000000U
#endif
```

## 9.4 QN9090 Reference Manual

The reference for QN9090 is availabe at https://www.nxp.com/webapp/Download?colCode=UM11141&location=null.

QN9090 is used as a host MCU on FinderV3 for SR040, RhodesV4 for SR1XX and few other boards.

## 9.5 Firmware Download From External Flash

This Feature can be used to avoid bottelnecking of the MCU's Flash.

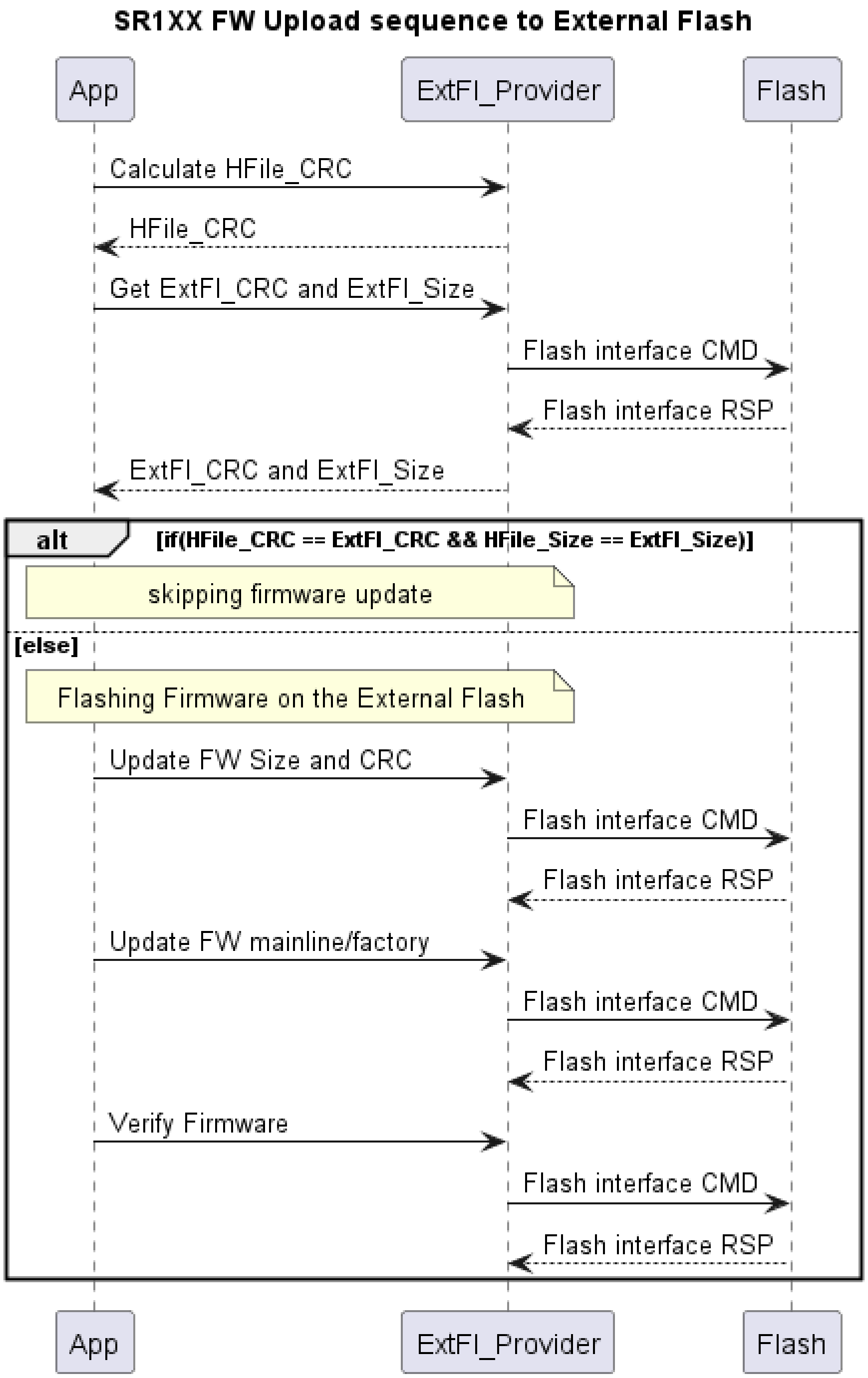
- The Firmware can be downloaded from the external flash during the Helios bootup.

### 9.5.1 Setup and Steps to Download Firmware From External Flash

1. Set the appropriate Firmware for the Helios to bootup.
2. build the `demo_mainline_extflash_fwdnld` / `demo_factory_extflash_fwdnld`
3. To run: Flash the `demo_mainline_extflash_fwdnld.bin` / `demo_factory_extflash_fwdnld.bin`

### Firmware update on External Flash Flow

Given below is the sequence diagram of Firmware update on External Flash




1) Generate the CRC of the selected Firmware.

2)**Get the CRC and the Firmware Size from the External Flash.**

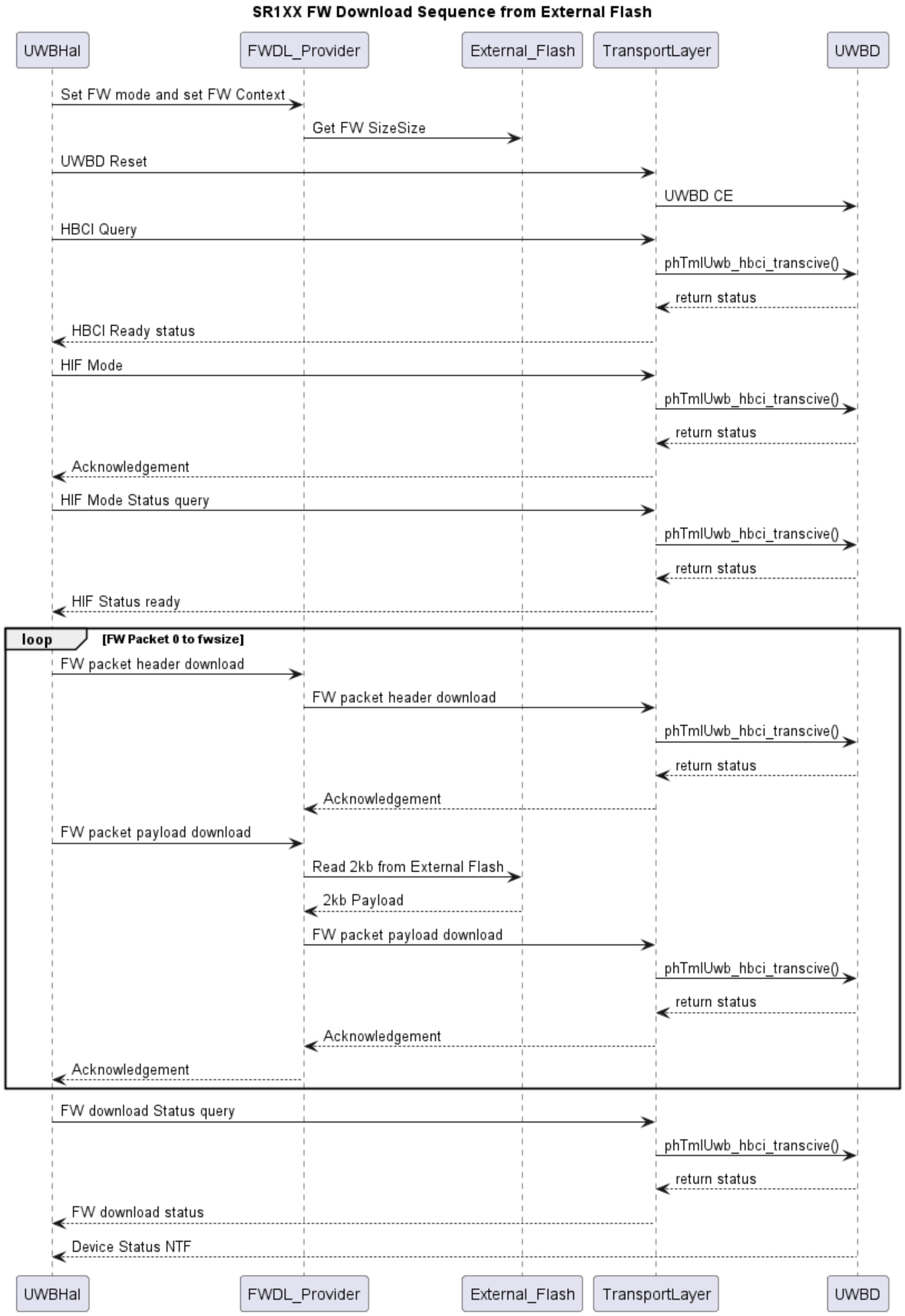
- If the external flash has the same version of Firmware, the Firmware update will be skipped.

- **Otherwise the Firmware will get flashed onto the External Flash.**
    - Update the CRC and Firmware Size on the External Flash.
    - Update the Mainline/Factory Firmware on the External Flash.
    - Verify the Firmware flashed onto the External Flash.

### 9.5.2 Firmware download from External Flash Flow

**After downloading the Firmware on the External Flash, Build and run required Application.**

Given below is the sequence diagram of Firmware download from External Flash




1) Set the appropriate firmware mode.

2) Read the Flash and get the Firmware Size.

3) Enable The UWBD.

4) Send HBCI Query.

5) Set The HIF mode.

6)**Download the Firmware to the UWBD.**

- Write the FW packet Header to the UWBD.
- Read 2kb of data from the External Flash.
- Write the 2kb Payload to the UWBD.

7) Query the download Status.

8) Wait for the status Notification from the UWBD.

## 9.6 FiRaLite ADF Provisioning

- Follow steps below to provision ADF in FiRaLite Applet.

---

**Note:**

- For that ADF provisioning needs to be used along with JCShell tool and fira-sgt tool.
- Use the ADF provisioning tool version *fira_sgt_v0.5*. The tool is packaged along with the MW available at root package under *tools/FiRaLite/*.
- Use JCShell tool version ` JCShell IOT Tool v6.9.0.11 ` available at secure_files. Please contact your local CAS/FAE for support. Doc No *sw7351*.
- JCShell tool on PC windows is used with JRCPProxy runs on windows and the se_vcom binary is flashed on MCU device.
- JRCPProxy executable is available inside the *binaries/PCWindows/*. The source code is located at PNT_Release Navigate to TOOLS & SOFTWARE pick the MW release package source code is available at ..*\simw-top\ext\JRCPProxyConsole*
- Source se_vcom is available in package demos/se_vcom prebuilt binary can be found at :file:` binaries/<MCU>/se_vcom-SE051W-vxx.xx.xx.bin` , where `MCU` is your host platform `Rhodes4_SE` .

---

### 9.6.1 Setup and Steps for JCShell usage

1. Connect device to PC Windows.
2. On connected device flash se_VCOM `se_vcom-SE051W-vxx.xx.xx.bin` from binaries folder.
3. Launch the `JRCPProxy` from PC Windows Ref section below.
4. And then open the JCShell tool `Using the JCShell` Ref section below.

5. Then perform FiRa secure ranging using demos for FiRalite secure ranging demos.

### 9.6.2 Usage of JRCPProxy

**Launch the executable**

- JRCPProxyConsole.exe -c COM5 —> check COM number

### 9.6.3 Using the JCShell

Launch the JCShell tool using the .bat file provided inside the tool. Run below command one by one on device acting as initiator and responder respectively.

For initiator

```
*****************************************************************************
     /term Remote
     /card
     path="C:\ADFprovisionTool\fira_sgt_v0.5\fira_sgt_v0.5"
     fira-provision
     gen_all
     This will generate unique SE specific data at output\gen_all\adf\
↪default
     (Check the timestamp of newly generated scripts)
     "output\gen_all\adf\default\controller.jcsh-CREATE.jcsh"
     /close
```

For Responder

```
*****************************************************************************
     /term Remote
     /card
     path="C:\ADFprovisionTool\fira_sgt_v0.5\fira_sgt_v0.5"
     fira-provision
     gen_all
     This will generate unique SE specific data at output\gen_all\adf\
↪default
     (Check the timestamp of newly generated scripts)
     "output\gen_all\adf\default\controlee.jcsh-CREATE.jcsh"
     /close
```

**Warning:** This is prerequisite for running Demos with FiRaLite Applet.

## 9.7 SR150 Boards

This section lists various development boards used/applicable to this software stack.

### 9.7.1 RHODES IV

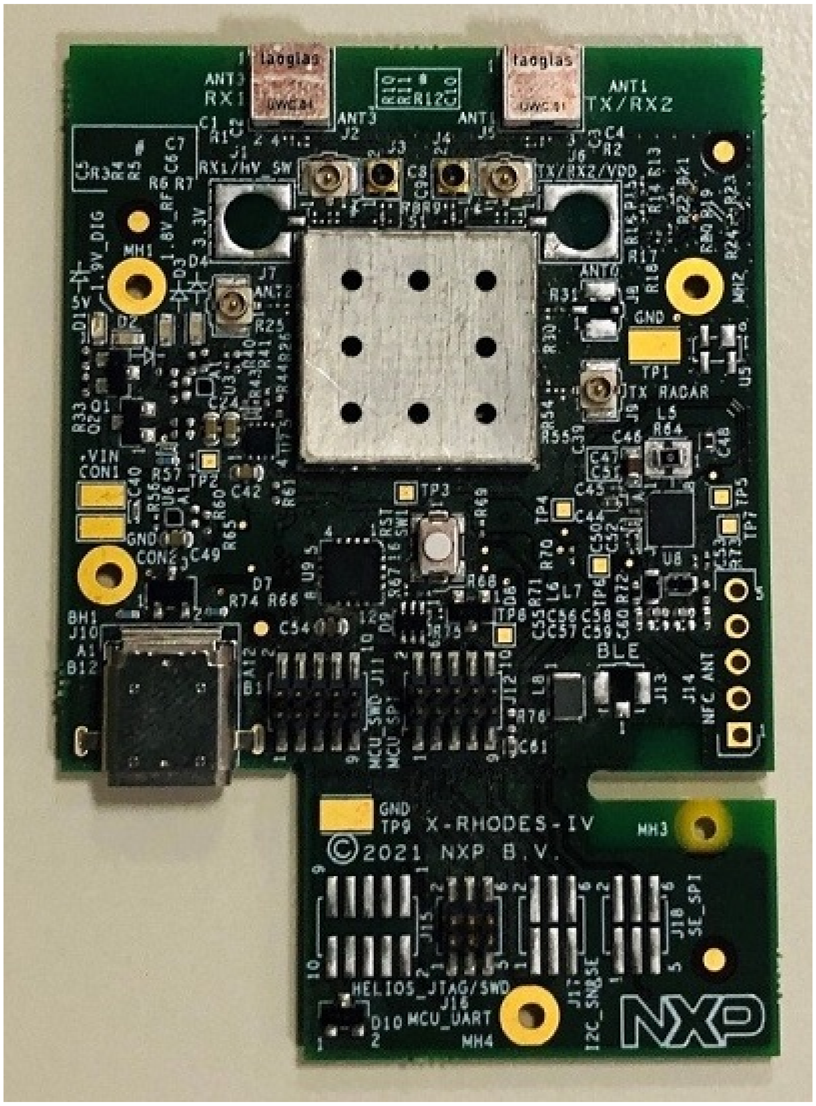
Rhodes V4 is one of the reference boards for UWB, QN9090 is the host mircocontroller. please contact local FAE/CAS Support team for UM11667_Rhodesv4_UserManual.pdf

**The board contains the following components:**

- SR150 - UWBS
- FT230K (FTDI chip - UART to USB converter) is also embedded in Rhodes v4, which provides facility to access the rv4 with USB
- Secure element : SE051
- External flash (Not supported yet)

You will need a USB Type C Cable to connect to and use this board.

Below is the image of the board.




There are few variants of this board.

1) V3 Demonstrator that is used for Ranging with 3D AoA.

   For using this board, please ensure that `USE_NAKED_BOARD` is set to `0` in `boards/Host/Rhodes4/UWB_DeviceConfig.h`

2) Bare / Naked / Plain Rhodes V4, without 3D Antennas (Only 2 Antennas are used).

   For using this board, please ensure that `USE_NAKED_BOARD` is set to 1 in `boards/Host/Rhodes4/UWB_DeviceConfig.h`

### 9.7.2 Nordic(NRF52840) + MK Shield Board (SR150)

This setup is requried for the Nordic and Mk Shield , For this we need

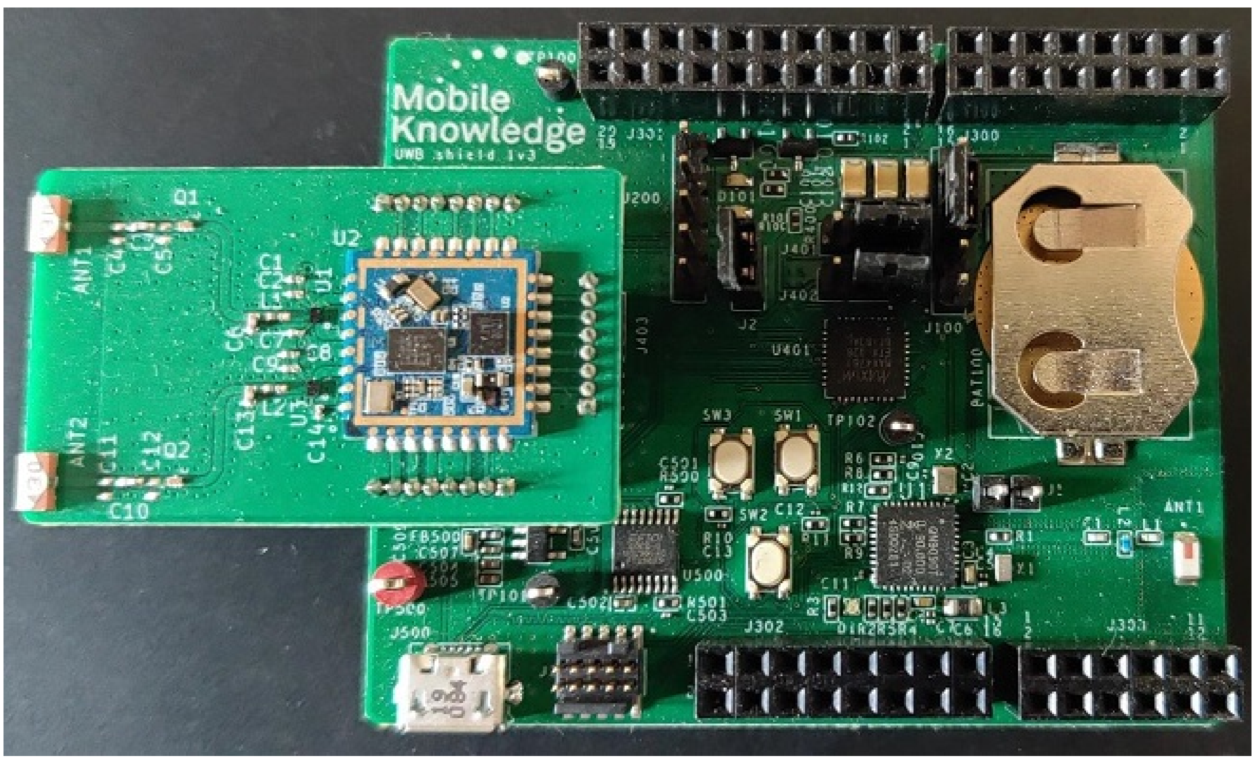
Hardware Setup :

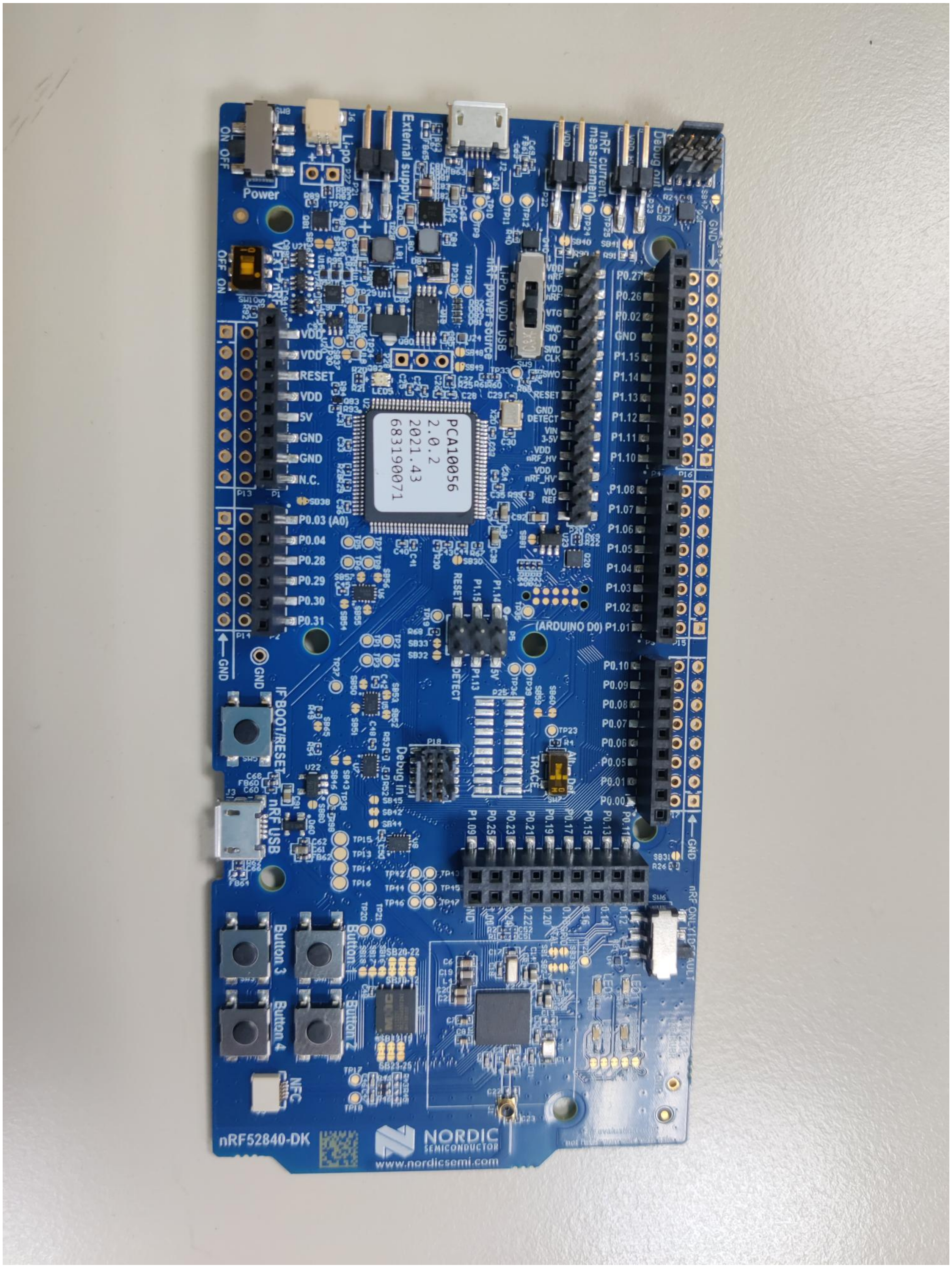
**The board contains the following components:**

- SR150 as UWB device
- Nordic Host micro controller.
- Secure element : SE051W

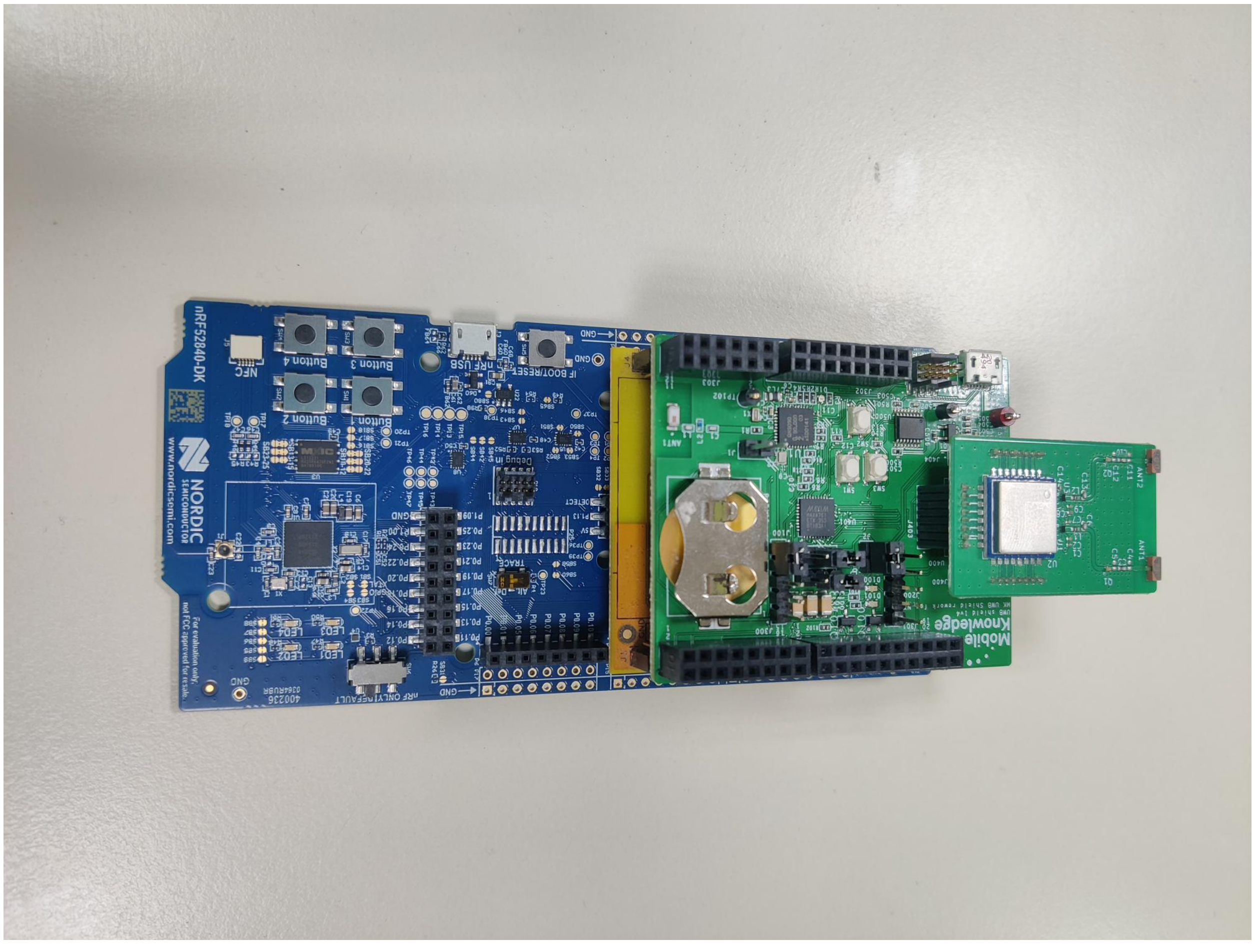
You will need a USB Type B Cable to connect to and use this board.

Below is the image of the Nordic and MK shield:






Below is the image of the stack of Nordic and MK shield:



Software Setup :

Download the segger embedded studio below mentioned link.

- https://www.segger.com/downloads/embedded-studio/
- Note: Please install version v630.
- Externally SDK download is not required it is included along with Project.

Following demos are supported for Nordic :

- UWBIOT_APP_BUILD__DEMO_BINDING
- UWBIOT_APP_BUILD__DEMO_RANGING_CONTROLLER
- UWBIOT_APP_BUILD__DEMO_RANGING_CONTROLEE
- UWBIOT_APP_BUILD__DEMO_INBAND_DATA_TRANSFER_RX
- UWBIOT_APP_BUILD__DEMO_INBAND_DATA_TRANSFER_TX
- UWBIOT_APP_BUILD__DEMO_OTP_STORAGE_FACTORY
- UWBIOT_APP_BUILD__DEMO_OTP_STORAGE_MAINLINE

- UWBIOT_APP_BUILD__DEMO_FL_INITIATOR
- UWBIOT_APP_BUILD__DEMO_FL_RESPONDER
- UWBIOT_APP_BUILD__DEMO_FL_RESPONDER_IOT_CONCURRENCY
- UWBIOT_APP_BUILD__DEMO_PNP
- UWBIOT_APP_BUILD__DEMO_MCTT_PCTT
- UWBIOT_APP_BUILD__DEMO_DLTDOA_INITIATOR
- UWBIOT_APP_BUILD__DEMO_DLTDOA_RESPONDER
- UWBIOT_APP_BUILD__DEMO_DLTDOA_TAG
- UWBIOT_APP_BUILD__SE_VCOM
- UWBIOT_APP_BUILD__DEMO_ULTDOA_ANCHOR
- UWBIOT_APP_BUILD__DEMO_ULTDOA_TAG
- UWBIOT_APP_BUILD__DEMO_ULTDOA_SYNC_ANCHOR
- UWBIOT_APP_BUILD__DEMO_TEST_TX
- UWBIOT_APP_BUILD__DEMO_TEST_RX
- UWBIOT_APP_BUILD__DEMO_HYBRID_RANGING_CONTROLLER
- UWBIOT_APP_BUILD__DEMO_HYBRID_RANGING_CONTROLEE
- UWBIOT_APP_BUILD__DEMO_CSA_CONTROLEE

# INDEX

## A

## C

## D

## E

# P

## R



## S

# T



# U

# V

## W



## X