

# UWB UWBMAC API

Qorvo

Release R12.7.0-405-gb33c5c4272



Contents

1	Overview	3
2	UWBMAC API	3
2.1	UWBMAC API	3
2.2	UWBMAC embedded API	63
2.3	FiRa helper API	65
2.4	PCTT helper API	184
	Index	203

---

1 Overview

We need an abstraction to use our uwb stack that is the same for all hardware/software architecture in order to develop cross platform user apps and the like. We divided this interface in two layers:

- UWBMAC API:

The first layer is UWBMAC API. It abstracts calls to the MAC. This layer respects the open/close principle and allows to develop any possible application with our stack.

- Helpers:

The second layer are helpers. Helpers are here to facilitate the development of application using known protocols (such as FiRa). This layer uses exclusively UWBMAC API to expose protocols related abstractions.

2 UWBMAC API

2.1 UWBMAC API

2.1.1 macro QDEPRECATED

**QDEPRECATED**(when, what)

Marks a function as deprecated.

Parameters

- **when** – when the function will be removed.
- **what** – what to use instead.

### 2.1.2 macro UWBMAC\_MAX\_CHANNEL\_COUNT

**UWBMAC\_MAX\_CHANNEL\_COUNT()**

Maximum number of channels in use at the same time.

### 2.1.3 enum uwbmact\_device\_state

enum **uwbmact\_device\_state**

State of the device.

#### 2.1.3.1 Definition

```
enum uwbmact_device_state {
    UWBMAC_DEVICE_STATE_STOPPED,
    UWBMAC_DEVICE_STATE_STARTED,
    UWBMAC_DEVICE_STATE_BROKEN
};
```

#### 2.1.3.2 Constants

**UWBMAC\_DEVICE\_STATE\_STOPPED**

Device is stopped.

**UWBMAC\_DEVICE\_STATE\_STARTED**

Device is started.

**UWBMAC\_DEVICE\_STATE\_BROKEN**

Device is in a unrecoverable broken state.

### 2.1.4 typedef uwbmact\_device\_state\_cb

void **uwbmact\_device\_state\_cb**(void \*user\_data, enum [uwbmact\\_device\\_state](#) state)

Receive a device state report callback.

#### Parameters

- **user\_data** (void\*) – data given when registering this callback.
- **state** (enum [uwbmact\\_device\\_state](#)) – New device state.

#### 2.1.4.1 Description

This is called when the device changes state.

#### 2.1.4.2 Return

nothing.

#### 2.1.5 typedef uwbmact\_call\_region\_cb

void **uwbmact\_call\_region\_cb**(void \*user\_data, uint32\_t call\_id, struct **uwbmact\_msg** \*call\_params)

Receive a region call callback.

##### Parameters

- **user\_data** (void\*) – data given when registering this callback.
- **call\_id** (uint32\_t) – the region call identifier.
- **call\_params** (struct **uwbmact\_msg**\*) – the payload of the callback.

#### 2.1.5.1 Return

nothing.

#### 2.1.6 struct uwbmact\_data\_ops

struct **uwbmact\_data\_ops**

Data operations.

##### 2.1.6.1 Definition

```
struct uwbmact_data_ops {
    void (*tx_done)(void *user_data, struct uwbmact_buf *buf, bool success);
    void (*tx_queue_stop)(void *user_data, int queue_index);
    void (*tx_queue_wake)(void *user_data, int queue_index);
    void (*rx)(void *user_data, struct uwbmact_buf *buf, int queue_index);
}
```

##### 2.1.6.2 Members

###### tx\_done

Called when a buffer given to **struct uwbmact\_tx**() can be disposed. If NULL, buffer is released.

The success parameter is true if the transmission was done successfully.

This callback must return quickly and it must not reenter the MAC. Typical implementation will release the memory, or add the buffer in a FIFO and wake up the processing thread.

###### tx\_queue\_stop

Called to signal a queue is stopped. Application should refrain from transmitting more data frame on this queue. If NULL, ignored.

This can be called while the application is calling a MAC function.

This callback must return quickly and it must not reenter the MAC. Typical implementation will clear a flag.

### tx\_queue\_wake

Called to signal a queue is woken up. Application can resume data frame transmission on this queue. If NULL, ignored.

This callback must return quickly and it must not reenter the MAC. Typical implementation will set a flag and wake up the processing thread.

### rx

Called when a data frame has been received and that the receiving queue is not stopped. If NULL, buffer is released.

This callback must return quickly and it must not reenter the MAC. Typical implementation will add the buffer in a FIFO and wake up the processing thread.

## 2.1.6.3 Description

The same interface is used for any data transfer when at least one of the active regions implements it.

Data is sent and received as MPDU without the FCS, this means that the MAC header must be included, but not the MAC footer. The data must be included in a struct `uwbmac_buf`.

## 2.1.6.4 Transmission

To send a data frame, use the `uwbmac_tx()` function. The MAC will handle all the timing details and send the frame when possible. Once the frame has been sent, or when the MAC determined that the frame cannot be sent, the `uwbmac_data_ops.tx_done` callback is called so that the application can have a status of the transmission and reclaim memory.

The MAC can handle several queues. Frame ordering for a recipient inside a queue is guaranteed, but not between two different recipients or between two different queues.

A queue can be stopped or woken up. When a queue is stopped, the application is expected to refrain transmission of any other frame for the same queue. Any transmission attempt will result in a error returned by `uwbmac_tx()`. Queue state change is signaled by `uwbmac_data_ops.tx_queue_stop` and `uwbmac_data_ops.tx_queue_wake` callbacks. Queues start in the woken up state.

## 2.1.6.5 Reception

When a data frame is received by the MAC, the `uwbmac_data_ops.rx` callback is called. The callback must quickly handle the received frame and return. Typical implementation will add the received data in a FIFO and wake the processing thread. Once the frame data has been processed, the application must release the associated memory.

Application can also control the flow of data reception by calling the `uwbmac_rx_queue_stop()` and `uwbmac_rx_queue_wake()` function. Queues start in the woken up state.

## 2.1.7 uwbmac\_get\_device\_count

```
enum qerr uwbmac_get_device_count(struct uwbmac_context *context, int *count)
```

Get the number of uwb chips available.

### Parameters

- **context** (struct `uwbmac_context*`) – UWB MAC context.
- **count** (int\*) – Number of uwb devices.

### 2.1.7.1 Return

QERR\_SUCCESS or error.

## 2.1.8 uwbmactet\_get\_supported\_channels

enum qerr **uwbmactet\_get\_supported\_channels**(struct uwbmactet\_context \*context, uint16\_t \*channels)

Get the supported UWB channels

### Parameters

- **context** (struct uwbmactet\_context\*) – UWB MAC context.
- **channels** (uint16\_t\*) – (out parameter) bitmask for supported channels. First bit is for channel 0, and so on.

### 2.1.8.1 Return

QERR\_SUCCESS or error.

## 2.1.9 uwbmactet\_init\_device

enum qerr **uwbmactet\_init\_device**(struct uwbmactet\_context \*context, unsigned int idx)

Fill the corresponding device information.

### Parameters

- **context** (struct uwbmactet\_context\*) – UWB MAC context.
- **idx** (unsigned int) – index of the device.

### 2.1.9.1 NOTE

use [struct uwbmactet\\_get\\_device\\_count](#) to check how many devices are present.

### 2.1.9.2 Return

QERR\_SUCCESS or error.

## 2.1.10 uwbmactet\_register\_device\_state\_callback

void **uwbmactet\_register\_device\_state\_callback**(struct uwbmactet\_context \*context, [uwbmactet\\_device\\_state\\_cb](#) cb, void \*user\_data)

Register a callback for device state change.

### Parameters

- **context** (struct uwbmactet\_context\*) – UWB MAC context.
- **cb** ([uwbmactet\\_device\\_state\\_cb](#)) – Callback to call on device state change.
- **user\_data** (void\*) – Context to give back to callback.

### 2.1.11 uwbbmac\_channel\_create

enum qerr **uwbbmac\_channel\_create**(struct uwbbmac\_context \*context, struct uwbbmac\_channel \*channel)  
Create a new channel.

#### Parameters

- **context** (struct uwbbmac\_context\*) – UWB MAC context.
- **channel** (struct uwbbmac\_channel\*) – The channel to be created.

#### 2.1.11.1 Return

QERR\_SUCCESS or error.

### 2.1.12 uwbbmac\_channel\_release

enum qerr **uwbbmac\_channel\_release**(struct uwbbmac\_channel \*channel)  
Release a channel.

#### Parameters

- **channel** (struct uwbbmac\_channel\*) – The channel to be released.

#### 2.1.12.1 Return

QERR\_SUCCESS or error.

### 2.1.13 uwbbmac\_channel\_set\_timeout

enum qerr **uwbbmac\_channel\_set\_timeout**(struct uwbbmac\_channel \*channel, int timeout)  
Set a timeout on a channel.

#### Parameters

- **channel** (struct uwbbmac\_channel\*) – The channel .
- **timeout** (int) – The timeout in seconds.

#### 2.1.13.1 Return

QERR\_SUCCESS or error.

### 2.1.14 uwbbmac\_channel\_receive

enum qerr **uwbbmac\_channel\_receive**(struct uwbbmac\_channel \*channel)  
Ask channel to process incoming messages if any.

#### Parameters

- **channel** (struct uwbbmac\_channel\*) – The channel that should process the messages.



#### 2.1.14.1 Return

QERR\_SUCCESS or error.

### 2.1.15 uwbmact\_register\_report\_callback

```
enum qerr uwbmact_register_report_callback(struct uwbmact_channel *channel, uwbmact\_call\_region\_cb
                                          msg_cb, void *user_data)
```

Register a region callback for a specific channel.

#### Parameters

- **channel** (struct uwbmact\_channel\*) – The channel associated with this callback.
- **msg\_cb** ([uwbmact\\_call\\_region\\_cb](#)) – Callback to call when a report is available on this channel.
- **user\_data** (void\*) – Context to give back to callback.

#### 2.1.15.1 Description

This function registers the callback to call in case of a mac event.

#### 2.1.15.2 NOTE

In embedded application, the callback might be called from MAC context, large treatments should be deferred.

#### 2.1.15.3 Return

QERR\_SUCCESS or error.

### 2.1.16 uwbmact\_register\_data\_ops

```
void uwbmact_register_data_ops(struct uwbmact_context *context, void *user_data, const struct
                               uwbmact\_data\_ops *ops)
```

Set callbacks used for data transfer.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **user\_data** (void\*) – Context to give back to callback.
- **ops** (const struct [uwbmact\\_data\\_ops](#)\*) – Structure with the data callbacks must be kept valid. NULL to clear callbacks.

### 2.1.16.1 Description

Please see [struct uwbmact\\_data\\_ops](#) for details.

### 2.1.17 uwbmact\_init

enum qerr **uwbmact\_init**(struct uwbmact\_context \*\*context)  
Initialize the UWB MAC and return an UWB MAC context.

#### Parameters

- **context** (struct uwbmact\_context\*\*) – UWB MAC context.

#### 2.1.17.1 NOTE

Some flavors of uwbmact have their own init method in their dedicated headers.

#### 2.1.17.2 Return

QERR\_SUCCESS or error.

### 2.1.18 uwbmact\_exit

void **uwbmact\_exit**(struct uwbmact\_context \*context)  
Free the UWB MAC.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.

### 2.1.19 uwbmact\_start

enum qerr **uwbmact\_start**(struct uwbmact\_context \*context)  
Start the device.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.

#### 2.1.19.1 Return

QERR\_SUCCESS or error.

### 2.1.20 uwbmac\_stop

enum qerr **uwbmac\_stop**(struct uwbmac\_context \*context)

Stop the device.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.

#### 2.1.20.1 Return

QERR\_SUCCESS or error.

### 2.1.21 uwbmac\_is\_started

bool **uwbmac\_is\_started**(struct uwbmac\_context \*context)

Return the state of UWB MAC.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.

#### 2.1.21.1 Return

true if UWB MAC is started, false otherwise.

### 2.1.22 uwbmac\_poll\_events

enum qerr **uwbmac\_poll\_events**(struct uwbmac\_context \*context, uint64\_t timeout\_us)

Poll next event.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **timeout\_us** (uint64\_t) – Timeout, in micro-seconds, for the poll.

#### 2.1.22.1 Description

This function is only available if you passed a NULL event\_loop\_ops to [uwbmac\\_init\(\)](#).

Passing 0 for timeout\_us will make the call non-bloking: existent pending event will be consume, and if there is no event the function will return instead of blocking.

Passing a value greated than 0 will make the function block until the timeout is reached when there is no pending event.

### 2.1.22.2 Return

QERR\_SUCCESS or error.

### 2.1.23 uwbmact\_set\_frame\_retries

enum qerr **uwbmact\_set\_frame\_retries**(struct uwbmact\_context \*context, int retries)

Set number of retries.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **retries** (int) – Number of retries between 0 and 7.

#### 2.1.23.1 Description

Set the number of tx frame retries when sending a frame with ACK.

#### 2.1.23.2 Return

QERR\_SUCCESS or error.

### 2.1.24 uwbmact\_tx

enum qerr **uwbmact\_tx**(struct uwbmact\_context \*context, struct uwbmact\_buf \*buf, int queue\_index)

Send a data frame.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **buf** (struct uwbmact\_buf\*) – Frame buffer.
- **queue\_index** (int) – Corresponding queue.

#### 2.1.24.1 Description

Please see [struct uwbmact\\_data\\_ops](#) for details.

#### 2.1.24.2 Return

QERR\_SUCCESS or error.

### 2.1.25 uwbmactx\_drop

void **uwbmactx\_drop**(struct uwbmactx\_buf \*buf)

Notifies a packet drop.

#### Parameters

- **buf** (struct uwbmactx\_buf\*) – Frame buffer.

### 2.1.26 uwbmactx\_queue\_stop

void **uwbmactx\_queue\_stop**(struct uwbmactx\_context \*context, int queue\_index)

Stop a reception queue.

#### Parameters

- **context** (struct uwbmactx\_context\*) – UWB MAC context.
- **queue\_index** (int) – Corresponding queue.

#### 2.1.26.1 Description

Please see [struct uwbmactx\\_data\\_ops](#) for details.

### 2.1.27 uwbmactx\_queue\_wake

void **uwbmactx\_queue\_wake**(struct uwbmactx\_context \*context, int queue\_index)

Wake up a reception queue.

#### Parameters

- **context** (struct uwbmactx\_context\*) – UWB MAC context.
- **queue\_index** (int) – Corresponding queue.

#### 2.1.27.1 Description

Please see [struct uwbmactx\\_data\\_ops](#) for details.

### 2.1.28 uwbmactx\_set\_channel

enum qerr **uwbmactx\_set\_channel**(struct uwbmactx\_context \*context, int channel)

Set UWB channel to use.

#### Parameters

- **context** (struct uwbmactx\_context\*) – UWB MAC context.
- **channel** (int) – Uwb channel, supported channels depend on driver/hardware. deprecated

### 2.1.28.1 Return

QERR\_SUCCESS or error.

### 2.1.29 uwbmac\_get\_channel

enum qerr **uwbmac\_get\_channel**(struct uwbmac\_context \*context, int \*channel)

Get used UWB channel.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **channel** (int\*) – Uwb channel, supported channels depend on driver/hardware. deprecated

### 2.1.29.1 Return

QERR\_SUCCESS or error.

### 2.1.30 uwbmac\_set\_channel\_preamble\_code

enum qerr **uwbmac\_set\_channel\_preamble\_code**(struct uwbmac\_context \*context, int channel, int preamble\_code)

Set UWB channel and preamble code to use.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **channel** (int) – UWB channel, supported channels depend on driver/hardware.
- **preamble\_code** (int) – UWB preamble code.

### 2.1.30.1 Return

QERR\_SUCCESS or error.

### 2.1.31 uwbmac\_get\_channel\_preamble\_code

enum qerr **uwbmac\_get\_channel\_preamble\_code**(struct uwbmac\_context \*context, int \*channel, int \*preamble\_code)

Get currently used UWB channel and preamble code.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **channel** (int\*) – UWB channel, supported channels depend on driver/hardware.
- **preamble\_code** (int\*) – UWB preamble code.

### 2.1.31.1 Return

QERR\_SUCCESS or error.

### 2.1.32 uwbmact\_calibration\_transaction\_start

enum qerr **uwbmact\_calibration\_transaction\_start**(struct uwbmact\_context \*context)

Start a calibration transaction allowing to update configuration value(s).

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.

#### 2.1.32.1 Description

Setting calibration value(s) as part of a transaction allows to optimize the Flash usage. When using a transaction, the new calibration values will be stored in the Flash only once, at transaction ended. While when not using a transaction will cause as many Flash write as the number of calibration values changed.

#### 2.1.32.2 A calibration transaction should be performed by the following API call sequence

1. Start the transaction by calling `uwbmact_calibration_transaction_start`.
2. Update the calibration values by calling `uwbmact_set_calibration` for each of them.
3. End the transaction by calling `uwbmact_calibration_transaction_end`.

Warning: starting a transaction without ending it will prevent storing values in the Flash, thus causing a mismatch between persistent stored calibration, and the one used by the stack. Values updated will be lost at next reboot.

### 2.1.32.3 Return

QERR\_SUCCESS or error.

### 2.1.33 uwbmact\_calibration\_transaction\_end

enum qerr **uwbmact\_calibration\_transaction\_end**(struct uwbmact\_context \*context)

End the ongoing calibration transaction.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.

#### 2.1.33.1 Description

That API should always be called after starting a transaction. See `uwbmact_calibration_transaction_start` for more details.

### 2.1.33.2 Return

QERR\_SUCCESS or error.

### 2.1.34 uwbmac\_set\_calibration

enum qerr **uwbmac\_set\_calibration**(struct uwbmac\_context \*context, const char \*key, void \*value, size\_t value\_size)

Send a calibration key and its value

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **key** (const char\*) – the calibration key name
- **value** (void\*) – the value for the specified calibration key
- **value\_size** (size\_t) – the size of the calibration key's value

#### 2.1.34.1 Note

setting calibration value(s) as part of a transaction allows to optimize the Flash usage. See `uwbmac_calibration_transaction_start` for more details.

### 2.1.34.2 Return

QERR\_SUCCESS or error.

### 2.1.35 uwbmac\_get\_calibration

enum qerr **uwbmac\_get\_calibration**(struct uwbmac\_context \*context, const char \*key, void \*value, int \*length, size\_t max\_length)

Retrieve a calibration value.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **key** (const char\*) – The calibration key name.
- **value** (void\*) – The output array for the specified calibration key.
- **length** (int\*) – The length of the the resulting array.
- **max\_length** (size\_t) – Capacity of the array given.



### 2.1.35.1 Return

QERR\_SUCCESS or error.

### 2.1.36 uwbmact\_get\_calibration\_key\_name

enum qerr **uwbmact\_get\_calibration\_key\_name**(struct uwbmact\_context \*context, uint16\_t key\_idx, char \*key)  
Get calibration key name for a specific key index.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **key\_idx** (uint16\_t) – Calibration key index to get name of.
- **key** (char\*) – Calibration key name to fill-in.

### 2.1.36.1 Return

QERR\_SUCCESS or error.

### 2.1.37 struct uwbmact\_list\_calibration\_context

struct **uwbmact\_list\_calibration\_context**  
context for listing calibration keys

### 2.1.37.1 Definition

```
struct uwbmact_list_calibration_context {
    const char *const *list;
    size_t key_count;
    void (*dealloc_cb)(struct uwbmact_list_calibration_context *list_calibration_ctx);
}
```

### 2.1.37.2 Members

#### list

list of retrieved calibration keys

#### key\_count

count of retrieved calibration keys

#### dealloc\_cb

callback for freeing memory buffer

### 2.1.38 uwbmac\_list\_calibrations

enum qerr **uwbmac\_list\_calibrations**(struct uwbmac\_context \*context, struct [uwbmac\\_list\\_calibration\\_context](#) \*list\_calibration\_ctx)

Retrieve the list calibration keys.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **list\_calibration\_ctx** (struct [uwbmac\\_list\\_calibration\\_context](#)\*) – Operation context.

#### 2.1.38.1 Description

The list must be freed by client by calling list\_calibration\_ctx->dealloc\_cb.

#### 2.1.38.2 Return

QERR\_SUCCESS or error.

### 2.1.39 uwbmac\_reset\_calibration

enum qerr **uwbmac\_reset\_calibration**(struct uwbmac\_context \*context)

Reset values for all calibration keys.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.

#### 2.1.39.1 Return

QERR\_SUCCESS or error.

### 2.1.40 uwbmac\_set\_pan\_id

enum qerr **uwbmac\_set\_pan\_id**(struct uwbmac\_context \*context, uint16\_t pan\_id)

Set pan id to use.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **pan\_id** (uint16\_t) – Pan id.

#### 2.1.40.1 NOTE

HW Filtering is disabled if promiscuous mode is enabled.

#### 2.1.40.2 Return

QERR\_SUCCESS or error.

#### 2.1.41 uwbmact\_set\_short\_addr

enum qerr **uwbmact\_set\_short\_addr**(struct uwbmact\_context \*context, uint16\_t short\_addr)

Set short address to use.

##### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **short\_addr** (uint16\_t) – Short address.

#### 2.1.41.1 NOTE

HW Filtering is disabled if promiscuous mode is enabled.

#### 2.1.41.2 Return

QERR\_SUCCESS or error.

#### 2.1.42 uwbmact\_set\_extended\_addr

enum qerr **uwbmact\_set\_extended\_addr**(struct uwbmact\_context \*context, uint64\_t extended\_addr)

Set extended address to use.

##### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **extended\_addr** (uint64\_t) – extended address.

#### 2.1.42.1 NOTE

HW Filtering is disabled if promiscuous mode is enabled.

### 2.1.42.2 Return

QERR\_SUCCESS or error.

## 2.1.43 uwbmact\_set\_promiscuous\_mode

enum qerr **uwbmact\_set\_promiscuous\_mode**(struct uwbmact\_context \*context, bool on)

Set promiscuous mode.

### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **on** (bool) – True to enable promiscuous mode.

### 2.1.43.1 Description

Control hardware filtering, if promiscuous mode is enabled, the hardware filtering is disabled.

### 2.1.43.2 Return

QERR\_SUCCESS or error.

## 2.1.44 uwbmact\_set\_scheduler

enum qerr **uwbmact\_set\_scheduler**(struct uwbmact\_context \*context, const char \*name, const struct [uwbmact\\_msg](#) \*params)

Set the scheduler responsible for managing the schedule, and configure its parameters.

### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **name** (const char\*) – Scheduler name.
- **params** (const struct [uwbmact\\_msg](#)\*) – Scheduler parameters.

### 2.1.44.1 Description

Device should not be started for the moment.

### 2.1.44.2 Return

QERR\_SUCCESS or error.

### 2.1.45 uwbbmac\_get\_scheduler

enum qerr **uwbbmac\_get\_scheduler**(struct uwbbmac\_context \*context, char \*name, int max\_length)

Get the scheduler name in use.

#### Parameters

- **context** (struct uwbbmac\_context\*) – UWB MAC context.
- **name** (char\*) – The buffer to fill with the scheduler name.
- **max\_length** (int) – Length of provided buffer.

#### 2.1.45.1 Return

QERR\_SUCCESS or error.

### 2.1.46 uwbbmac\_close\_scheduler

enum qerr **uwbbmac\_close\_scheduler**(struct uwbbmac\_context \*context)

Close the current scheduler and all regions.

#### Parameters

- **context** (struct uwbbmac\_context\*) – UWB MAC context.

#### 2.1.46.1 Return

QERR\_SUCCESS or error.

### 2.1.47 uwbbmac\_set\_scheduler\_parameters

enum qerr **uwbbmac\_set\_scheduler\_parameters**(struct uwbbmac\_context \*context, const char \*name, const struct [uwbbmac\\_msg](#) \*params)

Set the scheduler parameters.

#### Parameters

- **context** (struct uwbbmac\_context\*) – UWB MAC context.
- **name** (const char\*) – Scheduler name.
- **params** (const struct [uwbbmac\\_msg](#)\*) – Scheduler parameters.

#### 2.1.47.1 Return

QERR\_SUCCESS or error.

## 2.1.48 uwbmact\_get\_scheduler\_parameters

enum qerr **uwbmact\_get\_scheduler\_parameters**(struct uwbmact\_context \*context, const char \*name, struct [uwbmact\\_msg](#) \*reply)

Get the scheduler parameters.

### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **name** (const char\*) – Scheduler name.
- **reply** (struct [uwbmact\\_msg](#)\*) – Message filled with the parameters.

### 2.1.48.1 Return

QERR\_SUCCESS or error.

## 2.1.49 uwbmact\_set\_regions

enum qerr **uwbmact\_set\_regions**(struct uwbmact\_context \*context, const char \*scheduler\_name, uint32\_t region\_id, const char \*region\_name, const struct [uwbmact\\_msg](#) \*params)

Set regions that populate the schedule.

### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **scheduler\_name** (const char\*) – Scheduler name.
- **region\_id** (uint32\_t) – Identifier of the region, scheduler specific.
- **region\_name** (const char\*) – Name of region to attach to the scheduler.
- **params** (const struct [uwbmact\\_msg](#)\*) – Region parameters.

### 2.1.49.1 Return

QERR\_SUCCESS or error.

## 2.1.50 uwbmact\_set\_region\_parameters

enum qerr **uwbmact\_set\_region\_parameters**(struct uwbmact\_context \*context, const char \*scheduler\_name, uint32\_t region\_id, const char \*region\_name, const struct [uwbmact\\_msg](#) \*params)

Set region parameters.

### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **scheduler\_name** (const char\*) – Scheduler name.
- **region\_id** (uint32\_t) – Identifier of the region, scheduler specific.
- **region\_name** (const char\*) – Name of region to attach to the scheduler.
- **params** (const struct [uwbmact\\_msg](#)\*) – Region parameters.

### 2.1.50.1 Return

QERR\_SUCCESS or error.

### 2.1.51 uwbmact\_get\_region\_parameters

```
enum qerr uwbmact_get_region_parameters(struct uwbmact_context *context, const char *scheduler_name,
                                       uint32_t region_id, const char *region_name, struct uwbmact_msg
                                       *reply)
```

Get region parameters.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **scheduler\_name** (const char\*) – Scheduler name.
- **region\_id** (uint32\_t) – Identifier of the region, scheduler specific.
- **region\_name** (const char\*) – Name of the region to call.
- **reply** (struct uwbmact\_msg\*) – Empty message to store parameters.

#### 2.1.51.1 NOTE

uwbmact\_call\_region\_free must be called on the reply when done.

### 2.1.51.2 Return

QERR\_SUCCESS or error.

### 2.1.52 uwbmact\_call\_scheduler

```
enum qerr uwbmact_call_scheduler(struct uwbmact_context *context, const char *name, uint32_t call_id, const
                                struct uwbmact_msg *params, const struct uwbmact_channel *channel)
```

Call scheduler specific procedure.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **name** (const char\*) – Scheduler name.
- **call\_id** (uint32\_t) – Identifier of the procedure, scheduler specific.
- **params** (const struct uwbmact\_msg\*) – Scheduler call parameters.
- **channel** (const struct uwbmact\_channel\*) – Channel to get response.

### 2.1.52.1 Return

QERR\_SUCCESS or error.

### 2.1.53 uwbmactall\_region

```
int uwbmactall_region(struct uwbmactcontext *context, const char *scheduler_name, uint32_t region_id, const
char *region_name, uint32_t call_id, const struct uwbmactmsg *params, const struct
uwbmactchannel *channel, struct uwbmactmsg *reply)
```

Call region specific procedure.

#### Parameters

- **context** (struct uwbmactcontext\*) – UWB MAC context.
- **scheduler\_name** (const char\*) – Scheduler name.
- **region\_id** (uint32\_t) – Identifier of the region, scheduler specific.
- **region\_name** (const char\*) – Name of the region to call.
- **call\_id** (uint32\_t) – Identifier of the procedure, region specific.
- **params** (const struct uwbmactmsg\*) – Region call parameters.
- **channel** (const struct uwbmactchannel\*) – Channel to get response if reply is not NULL.
- **reply** (struct uwbmactmsg\*) – If not NULL, wait for a reply and store its payload here.

#### 2.1.53.1 NOTE

most calls to this function do not trigger a response, so reply must only be given when a reply is expected, in which case uwbmactcall\_region\_free must be called on the reply when done.

### 2.1.53.2 Return

QERR\_SUCCESS, error or a positive return code.

### 2.1.54 uwbmactcall\_region\_free

```
void uwbmactcall_region_free(struct uwbmactmsg *reply)
```

Free internal resources after uwbmactcall\_region.

#### Parameters

- **reply** (struct uwbmactmsg\*) – The reply filled in by a call to uwbmactcall\_region.



### 2.1.55 uwbmactime\_ns

enum qerr **uwbmactime\_ns**(struct uwbmactime\_context \*context, uint64\_t \*time)

Get the current MAC time.

#### Parameters

- **context** (struct uwbmactime\_context\*) – UWB MAC context.
- **time** (uint64\_t\*) – Pointer to store current MAC time.

#### 2.1.55.1 Return

QERR\_SUCCESS or error.

### 2.1.56 uwbmactime\_version

const char \***uwbmactime\_version**(void)

Get the uwbmactime release version.

#### Parameters

- **void** – no arguments

#### 2.1.56.1 Return

The release version string.

### 2.1.57 struct uwbmactime\_pids\_info

struct **uwbmactime\_pids\_info**

UWB SPI pids.

#### 2.1.57.1 Definition

```
struct uwbmactime_pids_info {
    int spi;
    int dw3000_spi;
}
```

#### 2.1.57.2 Members

**spi**

pid of dw3000 spi controller

**dw3000\_spi**

pid of dw3000

### 2.1.58 uwbmact\_get\_spi\_pids

enum qerr **uwbmact\_get\_spi\_pids**(struct uwbmact\_context \*context, struct [uwbmact\\_pids\\_info](#) \*pids)

Return spi PIDs.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **pids** (struct [uwbmact\\_pids\\_info](#)\*) – spi PIDs returned.

#### 2.1.58.1 Return

QERR\_SUCCESS or error.

### 2.1.59 uwbmact\_set\_scanning\_mode

enum qerr **uwbmact\_set\_scanning\_mode**(struct uwbmact\_context \*context, bool enabled)

Enable or disable scanning.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **enabled** (bool) – True to enable ieee 802.15.4 scanning.

#### 2.1.59.1 Description

This mode is only used for IEEE 802.15.4 scanning, actual control must be handled by the MLME running on the client side.

#### 2.1.59.2 Return

QERR\_SUCCESS or error.

### 2.1.60 typedef uwbmact\_testmode\_cb\_t

void **uwbmact\_testmode\_cb\_t**(void \*user\_data, void \*data, int length)

Receive a testmode call response.

#### Parameters

- **user\_data** (void\*) – data given when registering this callback.
- **data** (void\*) – response given.
- **length** (int) – length of data.

### 2.1.60.1 Return

nothing.

### 2.1.61 uwbmac\_register\_testmode\_callback

```
enum qerr uwbmac_register_testmode_callback(struct uwbmac_context *context, uwbmac\_testmode\_cb\_t
                                          msg_cb, void *user_data)
```

Register a testmode callback.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **msg\_cb** ([uwbmac\\_testmode\\_cb\\_t](#)) – Callback to call when the result of the test is available.
- **user\_data** (void\*) – Context to give back to callback.

#### 2.1.61.1 Description

This function registers the callback to call in case of a mac event. The callback is called from MAC context, big treatments should be deferred.

#### 2.1.61.2 NOTE

The msg sent to the callback should be freed by the APP using uwbmac\_buf\_free.

#### 2.1.61.3 Return

QERR\_SUCCESS or error.

### 2.1.62 uwbmac\_call\_testmode

```
enum qerr uwbmac_call_testmode(struct uwbmac_context *context, void *data, int length)
```

Call a test mode function.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **data** (void\*) – Test data.
- **length** (int) – Size of test data.

### 2.1.62.1 Description

Test mode allows to directly call the driver. This is expected to be called for tests. Test mode may be disabled in a device.

### 2.1.62.2 Return

QERR\_SUCCESS or error.

## 2.1.63 uwbmac\_trace\_init

enum qerr **uwbmac\_trace\_init**(void)  
Initialize trace management module.

### Parameters

- **void** – no arguments

### 2.1.63.1 Description

This API **must** be called by the user to initialize tracing.

### 2.1.63.2 NOTE

That API is only required for embedded systems.

### 2.1.63.3 Return

QERR\_SUCCESS or error.

## 2.1.64 enum uwbmac\_trace\_module\_ids

enum **uwbmac\_trace\_module\_ids**  
Unique ID for each trace module

### 2.1.64.1 Definition

```
enum uwbmac_trace_module_ids {
    UWBMAC_TRACE_MODULE_ID_MAIN,
    UWBMAC_TRACE_MODULE_ID_FBS,
    UWBMAC_TRACE_MODULE_ID_FIRA,
    UWBMAC_TRACE_MODULE_ID_LLD_COMMON,
    UWBMAC_TRACE_MODULE_ID_LLDD,
    UWBMAC_TRACE_MODULE_ID_LLDC,
    UWBMAC_TRACE_MODULE_ID_PCTT,
    UWBMAC_TRACE_MODULE_ID_RADAR,
    UWBMAC_TRACE_MODULE_ID_CCC,
    UWBMAC_TRACE_MODULE_NUMBER
};
```

### 2.1.64.2 Constants

#### UWBMAC\_TRACE\_MODULE\_ID\_MAIN

Main module.

#### UWBMAC\_TRACE\_MODULE\_ID\_FBS

FBS module.

#### UWBMAC\_TRACE\_MODULE\_ID\_FIRA

Fira module.

#### UWBMAC\_TRACE\_MODULE\_ID\_LLD\_COMMON

LLD Common module.

#### UWBMAC\_TRACE\_MODULE\_ID\_LLDD

LLDD module.

#### UWBMAC\_TRACE\_MODULE\_ID\_LLDC

LLDC module.

#### UWBMAC\_TRACE\_MODULE\_ID\_PCTT

PCTT module.

#### UWBMAC\_TRACE\_MODULE\_ID\_RADAR

Radar module.

#### UWBMAC\_TRACE\_MODULE\_ID\_CCC

CCC module.

#### UWBMAC\_TRACE\_MODULE\_NUMBER

Number of modules.

### 2.1.65 struct uwbmac\_trace\_info

struct **uwbmac\_trace\_info**

Trace module information

#### 2.1.65.1 Definition

```
struct uwbmac_trace_info {
    char name[UWBMAC_TRACE_MODULE_NAME_MAX_SIZE];
    bool enable;
}
```

#### 2.1.65.2 Members

##### name

name of the trace module

##### enable

true is trace module enabled, false otherwise

### 2.1.66 uwbmac\_trace\_module\_enable

enum qerr **uwbmac\_trace\_module\_enable**(const char \*module\_name, bool enable)

Enable/disable trace for a specific module

#### Parameters

- **module\_name** (const char\*) – Name of the module to set trace of.
- **enable** (bool) – true to enable, false to disable.

#### 2.1.66.1 Description

When the user wants to enable/disable a trace module, it sets true or false the **enable** parameter.

#### 2.1.66.2 NOTE

That API is only required for embedded systems.

#### 2.1.66.3 Return

QERR\_SUCCESS or error.

### 2.1.67 uwbmac\_trace\_module\_enable\_by\_id

enum qerr **uwbmac\_trace\_module\_enable\_by\_id**(enum [uwbmac\\_trace\\_module\\_ids](#) module\_id, bool enable)

Enable/disable trace for a specific module

#### Parameters

- **module\_id** (enum [uwbmac\\_trace\\_module\\_ids](#)) – ID of the module to set trace of.
- **enable** (bool) – true to enable, false to disable.

#### 2.1.67.1 Description

When the user wants to enable/disable a trace module, it sets true or false the **enable** parameter.

#### 2.1.67.2 NOTE

That API is only required for embedded systems.

### 2.1.67.3 Return

QERR\_SUCCESS or error.

### 2.1.68 uwbmactrace\_get\_trace\_modules

enum qerr uwbmactrace\_get\_trace\_modules(struct uwbmactrace\_info \*\*info, int \*nb\_modules)

Retrieve info of all trace modules available

#### Parameters

- **info** (struct uwbmactrace\_info\*\*) – output param where trace module informations are stored.
- **nb\_modules** (int\*) – output param where number of modules is stored.

#### 2.1.68.1 NOTE

That API is only required for embedded systems.

### 2.1.68.2 Return

QERR\_SUCCESS or error.

### 2.1.69 uwbmactrace\_is\_trace\_module\_enabled

bool uwbmactrace\_is\_trace\_module\_enabled(enum uwbmactrace\_module\_ids id)

Get trace enable status for a module.

#### Parameters

- **id** (enum uwbmactrace\_module\_ids) – unique ID of the module to get trace status of.

#### 2.1.69.1 NOTE

That API is only required for embedded systems.

### 2.1.69.2 Return

true if enable, false if disable or ID not found.

### 2.1.70 struct power\_state\_stats

struct power\_state\_stats

Contains power statistics details for one state

### 2.1.70.1 Definition

```
struct power_state_stats {
    uint32_t duration_ms;
    uint32_t count;
}
```

### 2.1.70.2 Members

**duration\_ms**  
total duration of the state in ms

**count**  
number of activations

### 2.1.71 struct uwbmact power\_stats

struct **uwbmact power\_stats**  
Contains power statistics about uwb

### 2.1.71.1 Definition

```
struct uwbmact power_stats {
    struct power_state_stats state_stats[UWBMAC_PWR_STATE_MAX];
    uint32_t interrupts;
}
```

### 2.1.71.2 Members

**state\_stats**  
statistics for each state

**interrupts**  
number of handled interrupts

### 2.1.72 struct uwbmact uwb\_device\_stats

struct **uwbmact uwb\_device\_stats**  
Contains device statistics about uwb



### 2.1.72.1 Definition

```
struct uwbmac_uwb_device_stats {
    int16_t temperature_hundredth_celsius;
}
```

### 2.1.72.2 Members

**temperature\_hundredth\_celsius**  
Temperature in hundredth of degree Celsius.

### 2.1.73 struct uwbmac\_device\_info

struct **uwbmac\_device\_info**  
Device information.

#### 2.1.73.1 Definition

```
struct uwbmac_device_info {
    uint64_t lot_id;
    uint32_t dev_id;
    uint32_t part_id;
}
```

#### 2.1.73.2 Members

**lot\_id**  
Lot ID.

**dev\_id**  
Device ID.

**part\_id**  
Part ID.

### 2.1.74 uwbmac\_set\_low\_power\_mode

enum qerr **uwbmac\_set\_low\_power\_mode**(struct uwbmac\_context \*context, bool enabled)  
Set low power mode.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **enabled** (bool) – True to enable low power mode state.

#### 2.1.74.1 Return

QERR\_SUCCESS or error.

#### 2.1.75 uwbmact\_get\_low\_power\_mode

bool **uwbmact\_get\_low\_power\_mode**(void)

Get low power mode S4 state.

##### Parameters

- **void** – no arguments

#### 2.1.75.1 Return

True if low power mode S4 is set, otherwise false.

#### 2.1.76 uwbmact\_set\_pm\_min\_inactivity\_s4

enum qerr **uwbmact\_set\_pm\_min\_inactivity\_s4**(struct uwbmact\_context \*context, uint32\_t time\_ms)

Set minimum inactivity time for S4.

##### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **time\_ms** (uint32\_t) – Minimum inactivity time to get into S4, in ms.

#### 2.1.76.1 Return

QERR\_SUCCESS or error.

#### 2.1.77 uwbmact\_get\_pm\_min\_inactivity\_s4

enum qerr **uwbmact\_get\_pm\_min\_inactivity\_s4**(struct uwbmact\_context \*context, uint32\_t \*time\_ms)

Get minimum inactivity time for S4.

##### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **time\_ms** (uint32\_t\*) – minimum inactivity time to get in S4, in ms.

#### 2.1.77.1 Return

QERR\_SUCCESS or error.

### 2.1.78 uwbmac\_se\_set\_key

enum qerr **uwbmac\_se\_set\_key**(uint32\_t session\_id, uint8\_t \*key, uint8\_t size, uint16\_t \*status)

*[Not supported in QM33 SDK]* Set a SE key for a given session.

#### Parameters

- **session\_id** (uint32\_t) – Id of the session.
- **key** (uint8\_t\*) – pointer to the session key
- **size** (uint8\_t) – length of the session key, can be 128 or 256 bits.
- **status** (uint16\_t\*) – SE status.

#### 2.1.78.1 Return

QERR\_SUCCESS or error.

### 2.1.79 uwbmac\_se\_derive\_key

enum qerr **uwbmac\_se\_derive\_key**(const uint8\_t \*key, const uint8\_t \*data, unsigned int data\_len, uint8\_t \*out)

*[Not supported in QM33 SDK]* Derive a key from a root key and derivation data.

#### Parameters

- **key** (const uint8\_t\*) – pointer to the root key
- **data** (const uint8\_t\*) – Derivation data.
- **data\_len** (unsigned int) – Derivation data length in bytes.
- **out** (uint8\_t\*) – pointer to the derived key

#### 2.1.79.1 Return

QERR\_SUCCESS or error.

### 2.1.80 uwbmac\_query\_gpio\_timestamp

enum qerr **uwbmac\_query\_gpio\_timestamp**(struct uwbmac\_context \*context, int64\_t \*timestamp\_us, uint8\_t \*sequence\_number)

*[Not supported in QM33 SDK]* Dequeue and return gpio timestamp and sequence number.

#### Parameters

- **context** (struct uwbmac\_context\*) – UWB MAC context.
- **timestamp\_us** (int64\_t\*) – Pointer to store the timestamp in microseconds.
- **sequence\_number** (uint8\_t\*) – Pointer to store the sequence number.

### 2.1.80.1 Return

QERR\_SUCCESS or error.

### 2.1.81 uwbmact\_get\_uwb\_device\_stats

enum qerr **uwbmact\_get\_uwb\_device\_stats**(struct uwbmact\_context \*context, struct [uwbmact\\_uwb\\_device\\_stats](#) \*uwb\_device\_stats)

[Not supported in QM33 SDK] Get uwb stats.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **uwb\_device\_stats** (struct [uwbmact\\_uwb\\_device\\_stats](#)\*) – Pointer to store the uwb stats.

### 2.1.81.1 Return

QERR\_SUCCESS or error.

### 2.1.82 uwbmact\_reinit\_crypto

enum qerr **uwbmact\_reinit\_crypto**(void)

Reinitialize crypto context from MCPS crypto.

#### Parameters

- **void** – no arguments

### 2.1.82.1 Return

QERR\_SUCCESS or error.

### 2.1.83 uwbmact\_get\_device\_info

enum qerr **uwbmact\_get\_device\_info**(struct uwbmact\_context \*context, struct [uwbmact\\_device\\_info](#) \*device\_info)

Get Device Information.

#### Parameters

- **context** (struct uwbmact\_context\*) – UWB MAC context.
- **device\_info** (struct [uwbmact\\_device\\_info](#)\*) – Pointer to store the device info.

### 2.1.83.1 Return

QERR\_SUCCESS or error.

### 2.1.84 macro UWBMAC\_BUF\_CB\_SIZE

**UWBMAC\_BUF\_CB\_SIZE()**

Size of the control block in a network buffer structure.

### 2.1.85 uwbmam\_buf\_alloc\_quota

struct uwbmam\_buf \***uwbmam\_buf\_alloc\_quota**(unsigned int size, enum mem\_quota\_id quota\_id)

Allocate a new network buffer with requested size.

#### Parameters

- **size** (unsigned int) – Size of buffer.
- **quota\_id** (enum mem\_quota\_id) – Quota to use for this allocation.

### 2.1.85.1 Return

Pointer to the new buffer, or NULL if no memory available.

### 2.1.86 uwbmam\_buf\_alloc

struct uwbmam\_buf \***uwbmam\_buf\_alloc**(unsigned int size)

Allocate a new network buffer with requested size.

#### Parameters

- **size** (unsigned int) – Size of buffer.

### 2.1.86.1 Return

Pointer to the new buffer, or NULL if no memory available.

### 2.1.87 uwbmam\_buf\_free

void **uwbmam\_buf\_free**(struct uwbmam\_buf \*buf)

Release a network buffer.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to release.

### 2.1.88 uwbmam\_buf\_reserve

void **uwbmam\_buf\_reserve**(struct uwbmam\_buf \*buf, unsigned int len)

Reserve some headroom on an empty buffer.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer where space needs to be reserved, must be empty.
- **len** (unsigned int) – Length to reserve.

### 2.1.89 uwbmam\_buf\_headroom

unsigned int **uwbmam\_buf\_headroom**(const struct uwbmam\_buf \*buf)

Return available space at start of buffer.

#### Parameters

- **buf** (const struct uwbmam\_buf\*) – Buffer.

#### 2.1.89.1 Return

Number of allocated free bytes before the data start.

### 2.1.90 uwbmam\_buf\_tailroom

unsigned int **uwbmam\_buf\_tailroom**(const struct uwbmam\_buf \*buf)

Return available space at end of buffer.

#### Parameters

- **buf** (const struct uwbmam\_buf\*) – Buffer.

#### 2.1.90.1 Return

Number of allocated free bytes after the data end.

### 2.1.91 uwbmam\_buf\_trim

void **uwbmam\_buf\_trim**(struct uwbmam\_buf \*buf, unsigned int len)

Trim data to the given length.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to trim.
- **len** (unsigned int) – New buffer length.

### 2.1.91.1 Description

If data is smaller than the trim length, the buffer is not modified.

### 2.1.91.2 NOTE

Use it only with not fragmented buffers.

## 2.1.92 uwbmam\_buf\_put

void **\*uwbmam\_buf\_put**(struct uwbmam\_buf \*buf, unsigned int len)

Prepare a buffer to append new data.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to prepare.
- **len** (unsigned int) – Length of data to add.

### 2.1.92.1 Description

This function returns a pointer to the first byte where data must be written. The caller must make sure that there is enough space before calling this function. If fragments are used, len must not exceed the tailroom of the last fragment of the buffer.

### 2.1.92.2 Return

Pointer to first new byte of data.

## 2.1.93 uwbmam\_buf\_put\_data

int **uwbmam\_buf\_put\_data**(struct uwbmam\_buf \*buf, const void \*data, unsigned int len)

Append data to a buffer.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to write to.
- **data** (const void\*) – Data to append.
- **len** (unsigned int) – Length of new data.

### 2.1.93.1 Description

The caller must make sure that there is enough space before calling this function.

### 2.1.93.2 Return

0 or error.

### 2.1.94 uwbmam\_buf\_put\_u8

void **uwbmam\_buf\_put\_u8**(struct uwbmam\_buf \*buf, uint8\_t data)

Append a single byte to a buffer.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to write to.
- **data** (uint8\_t) – Single byte to append.

#### 2.1.94.1 Description

The caller must make sure that there is enough space before calling this function.

### 2.1.95 uwbmam\_buf\_push

void \***uwbmam\_buf\_push**(struct uwbmam\_buf \*buf, unsigned int len)

Prepare a buffer to insert new data at buffer start.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to prepare.
- **len** (unsigned int) – Length of new data.

#### 2.1.95.1 Description

This function returns a pointer to the first byte where data must be written. The caller must make sure that there is enough space before calling this function.

### 2.1.95.2 Return

Pointer to first new byte of data.

### 2.1.96 uwbmam\_buf\_pull

void **uwbmam\_buf\_pull**(struct uwbmam\_buf \*buf, unsigned int len)

Extract data from the start of buffer.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to read.
- **len** (unsigned int) – Length of data to extract.



### 2.1.96.1 Description

The caller must make sure that there is enough data in the buffer before calling this function.

### 2.1.97 uwbmac\_buf\_queue\_init

```
void uwbmac_buf_queue_init(struct uwbmac_buf_queue *queue)
```

Initialize an empty queue.

#### Parameters

- **queue** (struct uwbmac\_buf\_queue\*) – Buffer queue to initialize.

### 2.1.98 uwbmac\_buf\_queue\_empty

```
bool uwbmac_buf_queue_empty(const struct uwbmac_buf_queue *queue)
```

Test whether a queue is empty.

#### Parameters

- **queue** (const struct uwbmac\_buf\_queue\*) – Buffer queue.

### 2.1.98.1 Return

true if the queue is empty.

### 2.1.99 uwbmac\_buf\_queue\_push

```
void uwbmac_buf_queue_push(struct uwbmac_buf_queue *queue, struct uwbmac_buf *buf)
```

Put a buffer at the start of a queue.

#### Parameters

- **queue** (struct uwbmac\_buf\_queue\*) – Buffer queue which will receive the buffer.
- **buf** (struct uwbmac\_buf\*) – Buffer to insert.

### 2.1.100 uwbmac\_buf\_queue\_put

```
void uwbmac_buf_queue_put(struct uwbmac_buf_queue *queue, struct uwbmac_buf *buf)
```

Put a buffer at the end of a queue.

#### Parameters

- **queue** (struct uwbmac\_buf\_queue\*) – Buffer queue which will receive the buffer.
- **buf** (struct uwbmac\_buf\*) – Buffer to insert.

### 2.1.101 uwbmac\_buf\_queue\_is\_last

```
bool uwbmac_buf_queue_is_last(const struct uwbmac_buf_queue *queue, const struct uwbmac_buf *buf)
```

Check if buf is the last entry in the queue.

#### Parameters

- **queue** (const struct uwbmac\_buf\_queue\*) – Queue head.
- **buf** (const struct uwbmac\_buf\*) – Current buffer.

#### 2.1.101.1 Return

True if buf is the last buffer on the list.

### 2.1.102 uwbmac\_buf\_queue\_next

```
struct uwbmac_buf *uwbmac_buf_queue_next(const struct uwbmac_buf_queue *queue, const struct uwbmac_buf *buf)
```

Return the next packet in the queue.

#### Parameters

- **queue** (const struct uwbmac\_buf\_queue\*) – Queue head.
- **buf** (const struct uwbmac\_buf\*) – Current buffer.

#### 2.1.102.1 Return

Next packet in the queue.

### 2.1.103 uwbmac\_buf\_queue\_peek

```
struct uwbmac_buf *uwbmac_buf_queue_peek(struct uwbmac_buf_queue *queue)
```

Peek a buffer from the start of a queue.

#### Parameters

- **queue** (struct uwbmac\_buf\_queue\*) – Buffer queue to peek the buffer from.

#### 2.1.103.1 Description

Buffer is left in the queue.

### 2.1.103.2 Return

The peeked buffer, or NULL if the queue is empty.

### 2.1.104 uwbmam\_buf\_queue\_pop

```
struct uwbmam_buf *uwbmam_buf_queue_pop(struct uwbmam_buf_queue *queue)
```

Get and remove a buffer from the start of a queue.

#### Parameters

- **queue** (struct uwbmam\_buf\_queue\*) – Buffer queue to extract the buffer from.

### 2.1.104.1 Return

The extracted buffer, or NULL if the queue is empty.

### 2.1.105 uwbmam\_buf\_queue\_purge

```
void uwbmam_buf_queue_purge(struct uwbmam_buf_queue *queue)
```

Release all buffers in a queue.

#### Parameters

- **queue** (struct uwbmam\_buf\_queue\*) – Buffer queue to purge

### 2.1.106 uwbmam\_buf\_get\_next\_frag

```
struct uwbmam_buf *uwbmam_buf_get_next_frag(struct uwbmam_buf *buf)
```

Retrieve next fragment data.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer.

### 2.1.106.1 Return

Pointer to next fragment or NULL.

### 2.1.107 uwbmam\_buf\_get\_data

```
uint8_t *uwbmam_buf_get_data(struct uwbmam_buf *buf)
```

Retrieve pointer to buffer data

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer.

### 2.1.107.1 Return

Pointer to first byte of buffer data.

### 2.1.108 uwbmactbuf\_get\_len

unsigned int **uwbmactbuf\_get\_len**(struct uwbmactbuf \*buf)  
Retrieve buffer data length.

#### Parameters

- **buf** (struct uwbmactbuf\*) – Buffer.

### 2.1.108.1 Return

Buffer data length.

### 2.1.109 uwbmactbuf\_get\_frag\_len

unsigned int **uwbmactbuf\_get\_frag\_len**(struct uwbmactbuf \*buf)  
Retrieve current fragment data length.

#### Parameters

- **buf** (struct uwbmactbuf\*) – Buffer/Fragment.

### 2.1.109.1 Return

Fragment data length.

### 2.1.110 uwbmactbuf\_get\_size

unsigned int **uwbmactbuf\_get\_size**(struct uwbmactbuf \*buf)  
Retrieve buffer size.

#### Parameters

- **buf** (struct uwbmactbuf\*) – Buffer.

### 2.1.110.1 Return

Buffer size.

### 2.1.111 uwbmam\_buf\_set\_queue\_mapping

void **uwbmam\_buf\_set\_queue\_mapping**(struct uwbmam\_buf \*buf, uint16\_t value)

Set queue mapping field of buffer.

#### Parameters

- **buf** (struct uwbmam\_buf\*) – Buffer to write to.
- **value** (uint16\_t) – queue\_mapping value to set.

### 2.1.112 uwbmam\_buf\_free\_msg\_priv

void **uwbmam\_buf\_free\_msg\_priv**(struct *uwbmam\_msg* \*msg)

Free priv member of msg.

#### Parameters

- **msg** (struct *uwbmam\_msg*\*) – Message to free.

### 2.1.113 struct uwbmam\_msg

struct **uwbmam\_msg**

Message container.

#### 2.1.113.1 Definition

```
struct uwbmam_msg {
    struct uwbmam_msg *parent;
    void *payload;
    uint8_t *position;
    void *priv;
    int length;
    int size;
    bool add_failed;
}
```

#### 2.1.113.2 Members

##### parent

Pointer to the parent, for nested messages.

##### payload

Pointer to the payload to be sent.

##### position

Pointer to the payload being written.

##### priv

Pointer to private data to keep around.

##### length

Length of the payload.

##### size

Capacity of the payload buffer.

## add\_failed

Set to true when an add call fails.

### 2.1.113.3 NOTE

do not access the fields directly, use the helper functions below.

### 2.1.114 uwbmactmsg\_free\_priv

void **uwbmactmsg\_free\_priv**(struct [uwbmactmsg](#) \*msg)

Free uwbmactmsg\_priv member.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message to free.

### 2.1.115 uwbmactmsg\_init

void **uwbmactmsg\_init**(struct [uwbmactmsg](#) \*msg, void \*payload, int length, int size)

Initialize message from payload/length

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message to initialize.
- **payload** (void\*) – Pointer to the payload.
- **length** (int) – Length of the payload.
- **size** (int) – Total size available in the payload.

### 2.1.116 uwbmactmsg\_copy

bool **uwbmactmsg\_copy**(struct [uwbmactmsg](#) \*msg, void \*payload, int length)

Copy a payload into a message

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message to initialize.
- **payload** (void\*) – Pointer to the payload.
- **length** (int) – Length of the payload.

#### 2.1.116.1 Return

true if there was enough space to do the copy, false otherwise.

### 2.1.117 uwbmac\_msg\_payload

void **uwbmac\_msg\_payload**(const struct [uwbmac\\_msg](#) \*msg)

Get the message payload

#### Parameters

- **msg** (const struct [uwbmac\\_msg](#)\*) – Message to use.

#### 2.1.117.1 Return

the message payload or NULL is msg is NULL.

### 2.1.118 uwbmac\_msg\_length

int **uwbmac\_msg\_length**(const struct [uwbmac\\_msg](#) \*msg)

Get the message payload length

#### Parameters

- **msg** (const struct [uwbmac\\_msg](#)\*) – Message to use.

#### 2.1.118.1 Return

the message payload length.

### 2.1.119 uwbmac\_msg\_size

int **uwbmac\_msg\_size**(const struct [uwbmac\\_msg](#) \*msg)

Get the message capacity

#### Parameters

- **msg** (const struct [uwbmac\\_msg](#)\*) – Message to use.

#### 2.1.119.1 Return

the message capacity.

### 2.1.120 enum uwbmac\_payload\_type

enum **uwbmac\_payload\_type**

UWB MAC serializable types.

### 2.1.120.1 Definition

```
enum uwbmac_payload_type {
    UWBMAC_PAYLOAD_TYPE_NONE,
    UWBMAC_PAYLOAD_TYPE_FLAG,
    UWBMAC_PAYLOAD_TYPE_BOOL,
    UWBMAC_PAYLOAD_TYPE_S8,
    UWBMAC_PAYLOAD_TYPE_S16,
    UWBMAC_PAYLOAD_TYPE_S32,
    UWBMAC_PAYLOAD_TYPE_S64,
    UWBMAC_PAYLOAD_TYPE_U8,
    UWBMAC_PAYLOAD_TYPE_U16,
    UWBMAC_PAYLOAD_TYPE_U32,
    UWBMAC_PAYLOAD_TYPE_U64,
    UWBMAC_PAYLOAD_TYPE_STRING,
    UWBMAC_PAYLOAD_TYPE_BINARY,
    UWBMAC_PAYLOAD_TYPE_NESTED
};
```

### 2.1.120.2 Constants

#### **UWBMAC\_PAYLOAD\_TYPE\_NONE**

No data to recover.

#### **UWBMAC\_PAYLOAD\_TYPE\_FLAG**

Flag - no data.

#### **UWBMAC\_PAYLOAD\_TYPE\_BOOL**

Boolean.

#### **UWBMAC\_PAYLOAD\_TYPE\_S8**

8 bit signed integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_S16**

16 bit signed integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_S32**

32 bit signed integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_S64**

64 bit signed integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_U8**

8 bit unsigned integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_U16**

16 bit unsigned integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_U32**

32 bit unsigned integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_U64**

64 bit unsigned integer.

#### **UWBMAC\_PAYLOAD\_TYPE\_STRING**

NULL terminated character string.

#### **UWBMAC\_PAYLOAD\_TYPE\_BINARY**

Binary object.



## UWBMAC\_PAYLOAD\_TYPE\_NESTED

Nested payload.

### 2.1.121 struct uwbmac\_parser\_element

struct uwbmac\_parser\_element

Helper to manipulate each UWB MAC elements.

#### 2.1.121.1 Definition

```
struct uwbmac_parser_element {
    void *data;
    uint16_t length;
    int *rlength;
    uint8_t type;
    uint8_t flags;
}
```

#### 2.1.121.2 Members

##### data

Pointer to data

##### length

Data length max

##### rlength

Data length found

##### type

Expected data type

##### flags

Tag mandatory/present

### 2.1.122 uwbmac\_parser\_init\_msg

void uwbmac\_parser\_init\_msg(struct uwbmac\_msg \*msg, void \*payload, int length)

Initialise on-stack uwbmac\_msg.

#### Parameters

- **msg** (struct uwbmac\_msg\*) – Message being initialized.
- **payload** (void\*) – Payload to parse.
- **length** (int) – Length of the payload to parse.

### 2.1.123 uwbmact\_parser\_read

```
enum qerr uwbmact_parser_read(struct uwbmact_msg *msg, struct uwbmact_parser_element elements, int
tag_max)
```

Read and parse payload.

#### Parameters

- **msg** (struct [uwbmact\\_msg](#)\*) – Message to parse.
- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag\_max** (int) – Maximum tag value (number of elements minus one).

#### 2.1.123.1 Return

QERR\_SUCCESS or error.

### 2.1.124 uwbmact\_parser\_read\_array

```
enum qerr uwbmact_parser_read_array(struct uwbmact_msg *msg, struct uwbmact_parser_element elements, int
tag_max, void *entry, int n, struct uwbmact_parser_read_array_info *info,
bool *keep_going)
```

Read and parse array payload.

#### Parameters

- **msg** (struct [uwbmact\\_msg](#)\*) – Message to parse.
- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag\_max** (int) – Maximum tag value (number of elements minus one).
- **entry** (void\*) – Array entry.
- **n** (int) – Number of array entries found so far.
- **info** (struct [uwbmact\\_parser\\_read\\_array\\_info](#)\*) – Internal loop data.
- **keep\_going** (bool\*) – Whether there are elements left in the array.

#### 2.1.124.1 Return

QERR\_SUCCESS or error.

### 2.1.125 uwbmact\_parser\_init\_nested\_loop

```
enum qerr uwbmact_parser_init_nested_loop(struct uwbmact_msg *msg, struct uwbmact_parser_read_array_info
*info)
```

Init internal loop data for nested iteration

#### Parameters

- **msg** (struct [uwbmact\\_msg](#)\*) – Message to parse.
- **info** (struct [uwbmact\\_parser\\_read\\_array\\_info](#)\*) – Internal loop data.

### 2.1.125.1 Description

This call is once before looping on each element with `uwbmac_parser_next_nested_loop_element`.

### 2.1.125.2 Return

QERR\_SUCCESS or error.

### 2.1.126 `uwbmac_parser_next_nested_loop_element`

```
enum qerr uwbmac_parser_next_nested_loop_element(struct uwbmac_msg *msg, struct uwbmac_msg *nested,
                                                struct uwbmac_parser_read_array_info *info, bool
                                                *keep_going)
```

Init nested with the next nested element

#### Parameters

- `msg` (struct `uwbmac_msg*`) – Message to parse.
- `nested` (struct `uwbmac_msg*`) – Message to init.
- `info` (struct `uwbmac_parser_read_array_info*`) – Internal loop data.
- `keep_going` (bool\*) – Whether there are elements left in the array.

### 2.1.126.1 Description

If `keep_going` is false, you've reached the end of the nested elements and `nested` was not initialized.

If `keep_going` is true, `nested` is initialized to point to the next element. You will need to setup your parsing and call `uwbmac_parser_read`.

### 2.1.126.2 Return

QERR\_SUCCESS or error.

### 2.1.127 `uwbmac_parser_is_present`

```
bool uwbmac_parser_is_present(struct uwbmac_parser_element elements, int tag)
```

Get tag presence status.

#### Parameters

- `elements` (struct `uwbmac_parser_element`) – Array of elements.
- `tag` (int) – Tag in the payload.

### 2.1.127.1 Return

true if present, false otherwise.

### 2.1.128 uwbmact\_parser\_add

void **uwbmact\_parser\_add**(struct [uwbmact\\_parser\\_element](#) elements, int tag, enum [uwbmact\\_payload\\_type](#) type, void \*data, int \*rlength, int length, bool mandatory)

Set elements.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **type** (enum [uwbmact\\_payload\\_type](#)) – Expected tag type.
- **data** (void\*) – Some pointer.
- **rlength** (int\*) – Actual payload's length.
- **length** (int) – Maximum expected length in the payload.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.128.1 NOTE

do not call directly, use one of the provided helpers.

### 2.1.129 uwbmact\_parser\_add\_none

void **uwbmact\_parser\_add\_none**(struct [uwbmact\\_parser\\_element](#) elements, int tag)

Set element to receive nothing.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.

### 2.1.130 uwbmact\_parser\_add\_flag

void **uwbmact\_parser\_add\_flag**(struct [uwbmact\\_parser\\_element](#) elements, int tag, bool \*data, bool mandatory)

Set element to receive an empty tag.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (bool\*) – Pointer to the boolean to set if present.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.131 uwbmact\_parser\_add\_bool

void **uwbmact\_parser\_add\_bool**(struct [uwbmact\\_parser\\_element](#) elements, int tag, bool \*data, bool mandatory)  
Set element to receive a boolean.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (bool\*) – Pointer to the boolean to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.132 uwbmact\_parser\_add\_s8

void **uwbmact\_parser\_add\_s8**(struct [uwbmact\\_parser\\_element](#) elements, int tag, int8\_t \*data, bool mandatory)  
Set element to receive a signed 8-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (int8\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.133 uwbmact\_parser\_add\_s16

void **uwbmact\_parser\_add\_s16**(struct [uwbmact\\_parser\\_element](#) elements, int tag, int16\_t \*data, bool mandatory)  
Set element to receive a signed 16-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (int16\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.134 uwbmact\_parser\_add\_s32

void **uwbmact\_parser\_add\_s32**(struct [uwbmact\\_parser\\_element](#) elements, int tag, int32\_t \*data, bool mandatory)  
Set element to receive a signed 32-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (int32\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.135 uwbmact\_parser\_add\_s64

void **uwbmact\_parser\_add\_s64**(struct [uwbmact\\_parser\\_element](#) elements, int tag, int64\_t \*data, bool mandatory)  
Set element to receive a signed 64-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (int64\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.136 uwbmact\_parser\_add\_u8

void **uwbmact\_parser\_add\_u8**(struct [uwbmact\\_parser\\_element](#) elements, int tag, uint8\_t \*data, bool mandatory)  
Set element to receive an unsigned 8-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (uint8\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.137 uwbmact\_parser\_add\_u16

void **uwbmact\_parser\_add\_u16**(struct [uwbmact\\_parser\\_element](#) elements, int tag, uint16\_t \*data, bool mandatory)  
Set element to receive an unsigned 16-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (uint16\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.138 uwbmact\_parser\_add\_u32

void **uwbmact\_parser\_add\_u32**(struct [uwbmact\\_parser\\_element](#) elements, int tag, uint32\_t \*data, bool mandatory)  
Set element to receive an unsigned 32-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (uint32\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.139 uwbmact\_parser\_add\_u64

void **uwbmact\_parser\_add\_u64**(struct [uwbmact\\_parser\\_element](#) elements, int tag, uint64\_t \*data, bool mandatory)

Set element to receive an unsigned 64-bit integer.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (uint64\_t\*) – Pointer to the integer to fill in.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.140 uwbmact\_parser\_add\_string

void **uwbmact\_parser\_add\_string**(struct [uwbmact\\_parser\\_element](#) elements, int tag, char \*data, int max\_length, bool mandatory)

Set element to receive a string.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (char\*) – Pointer to the string to fill in.
- **max\_length** (int) – Length available, including terminating NUL character.
- **mandatory** (bool) – Whether the element is mandatory in the message.

#### 2.1.140.1 NOTE

The payload string is copied from the payload to the provided string.

### 2.1.141 uwbmact\_parser\_add\_binary

void **uwbmact\_parser\_add\_binary**(struct [uwbmact\\_parser\\_element](#) elements, int tag, void \*data, int \*length, int max\_length, bool mandatory)

Set element to receive a binary object.

#### Parameters

- **elements** (struct [uwbmact\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **data** (void\*) – Pointer to the object to fill in.
- **length** (int\*) – Actual length.
- **max\_length** (int) – Length available.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.141.1 NOTE

The payload object is copied from the payload to the provided object.

### 2.1.142 uwbmac\_parser\_add\_nested

void **uwbmac\_parser\_add\_nested**(struct [uwbmac\\_parser\\_element](#) elements, int tag, struct [uwbmac\\_msg](#) \*nested, bool mandatory)

Set element to receive a nested message.

#### Parameters

- **elements** (struct [uwbmac\\_parser\\_element](#)) – Array of elements.
- **tag** (int) – Tag in the payload.
- **nested** (struct [uwbmac\\_msg](#)\*) – Pointer to the structure to keep internal data.
- **mandatory** (bool) – Whether the element is mandatory in the message.

### 2.1.143 uwbmac\_msg\_read\_tag

enum qerr **uwbmac\_msg\_read\_tag**(struct [uwbmac\\_msg](#) \*msg, int \*tag, bool \*is\_nested, int \*rem)

Tell current message element tag and data type.

#### Parameters

- **msg** (struct [uwbmac\\_msg](#)\*) – message in its current reading state
- **tag** (int\*) – output value
- **is\_nested** (bool\*) – output value telling whether data is a nested message to binary data.
- **rem** (int\*) – in/out remaining size to parse in message

#### 2.1.143.1 Description

Set message state to the next element. Used to serialize the message without knowing the meaning of its elements.

NB: *rem* parameter must be updated synchronously to *msg position* pointer.

#### 2.1.143.2 Return

status QERR\_SUCCESS or QERR\_EINVAL.

### 2.1.144 uwbmac\_msg\_read\_nested

enum qerr **uwbmac\_msg\_read\_nested**(struct [uwbmac\\_msg](#) \*msg, struct [uwbmac\\_msg](#) \*nested, int \*rem)

Get the current element as a nested msg

#### Parameters

- **msg** (struct [uwbmac\\_msg](#)\*) – message in its current reading state
- **nested** (struct [uwbmac\\_msg](#)\*) – messge to be initialized to point to the nested part of msg
- **rem** (int\*) – in/out remaining size to parse in message



### 2.1.144.1 Description

Actual type not checked, [uwbmac\\_msg\\_read\\_tag\(\)](#) should be used before. Set message state to the next element. Used to serialize the message without knowing the meaning of its elements.

NB: *rem* parameter must be updated synchronously to *msg position* pointer.

### 2.1.144.2 Return

status QERR\_SUCCESS or QERR\_EINVAL.

## 2.1.145 uwbmac\_msg\_read\_data

enum qerr **uwbmac\_msg\_read\_data**(struct [uwbmac\\_msg](#) \*msg, uint8\_t \*\*data, size\_t \*length, int \*rem)

Get the current element as binary data

### Parameters

- **msg** (struct [uwbmac\\_msg](#)\*) – message in its current reading state
- **data** (uint8\_t\*\*) – output, set to point to the data
- **length** (size\_t\*) – output, set to the data length
- **rem** (int\*) – in/out remaining size to parse in message

### 2.1.145.1 Description

Actual type not checked, [uwbmac\\_msg\\_read\\_tag\(\)](#) should be used before. Set message state to the next element. Used to serialize the message without knowing the meaning of its elements.

NB: *rem* parameter must be updated synchronously to *msg position* pointer.

### 2.1.145.2 Return

status QERR\_SUCCESS or QERR\_EINVAL.

## 2.1.146 uwbmac\_writer\_init\_msg

void **uwbmac\_writer\_init\_msg**(struct [uwbmac\\_msg](#) \*msg, void \*payload, int size)

Initialise on-stack uwbmac\_msg.

### Parameters

- **msg** (struct [uwbmac\\_msg](#)\*) – Message being initialized.
- **payload** (void\*) – Payload buffer to fill in.
- **size** (int) – Size of the payload buffer.

### 2.1.147 uwbbmac\_writer\_success

enum qerr uwbbmac\_writer\_success(const struct [uwbbmac\\_msg](#) \*msg)

Check that all 'add' operations succeeded.

#### Parameters

- **msg** (const struct [uwbbmac\\_msg](#)\*) – Message being written.

#### 2.1.147.1 Return

QERR\_SUCCESS on success, QERR\_EINVAL otherwise.

### 2.1.148 uwbbmac\_writer\_add

enum qerr uwbbmac\_writer\_add(struct [uwbbmac\\_msg](#) \*msg, int tag, const void \*data, int length)

Add tag and data to the message.

#### Parameters

- **msg** (struct [uwbbmac\\_msg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **data** (const void\*) – Payload related to tag.
- **length** (int) – Payload length.

#### 2.1.148.1 Return

QERR\_SUCCESS or error.

### 2.1.149 uwbbmac\_writer\_add\_flag

enum qerr uwbbmac\_writer\_add\_flag(struct [uwbbmac\\_msg](#) \*msg, int tag)

Add an empty tag to the message.

#### Parameters

- **msg** (struct [uwbbmac\\_msg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.

#### 2.1.149.1 Return

QERR\_SUCCESS or error.

### 2.1.150 uwbm\_writer\_add\_bool

enum qerr **uwbm\_writer\_add\_bool**(struct [uwbm\\_msg](#) \*msg, int tag, bool value)

Add a boolean to the message.

#### Parameters

- **msg** (struct [uwbm\\_msg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (bool) – Value to add.

#### 2.1.150.1 Return

QERR\_SUCCESS or error.

### 2.1.151 uwbm\_writer\_add\_s8

enum qerr **uwbm\_writer\_add\_s8**(struct [uwbm\\_msg](#) \*msg, int tag, int8\_t value)

Add a signed 8-bit integer to the message.

#### Parameters

- **msg** (struct [uwbm\\_msg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (int8\_t) – Value to add.

#### 2.1.151.1 Return

QERR\_SUCCESS or error.

### 2.1.152 uwbm\_writer\_add\_s16

enum qerr **uwbm\_writer\_add\_s16**(struct [uwbm\\_msg](#) \*msg, int tag, int16\_t value)

Add a signed 16-bit integer to the message.

#### Parameters

- **msg** (struct [uwbm\\_msg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (int16\_t) – Value to add.

### 2.1.152.1 Return

QERR\_SUCCESS or error.

### 2.1.153 uwbmactwriter\_add\_s32

enum qerr **uwbmactwriter\_add\_s32**(struct [uwbmactmsg](#) \*msg, int tag, int32\_t value)

Add a signed 32-bit integer to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (int32\_t) – Value to add.

### 2.1.153.1 Return

QERR\_SUCCESS or error.

### 2.1.154 uwbmactwriter\_add\_s64

enum qerr **uwbmactwriter\_add\_s64**(struct [uwbmactmsg](#) \*msg, int tag, int64\_t value)

Add a signed 64-bit integer to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (int64\_t) – Value to add.

### 2.1.154.1 Return

QERR\_SUCCESS or error.

### 2.1.155 uwbmactwriter\_add\_u8

enum qerr **uwbmactwriter\_add\_u8**(struct [uwbmactmsg](#) \*msg, int tag, uint8\_t value)

Add an unsigned 8-bit integer to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (uint8\_t) – Value to add.

### 2.1.155.1 Return

QERR\_SUCCESS or error.

### 2.1.156 uwbmactwriter\_add\_u16

enum qerr **uwbmactwriter\_add\_u16**(struct [uwbmactmsg](#) \*msg, int tag, uint16\_t value)

Add a unsigned 16-bit integer to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (uint16\_t) – Value to add.

### 2.1.156.1 Return

QERR\_SUCCESS or error.

### 2.1.157 uwbmactwriter\_add\_u32

enum qerr **uwbmactwriter\_add\_u32**(struct [uwbmactmsg](#) \*msg, int tag, uint32\_t value)

Add a unsigned 32-bit integer to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (uint32\_t) – Value to add.

### 2.1.157.1 Return

QERR\_SUCCESS or error.

### 2.1.158 uwbmactwriter\_add\_u64

enum qerr **uwbmactwriter\_add\_u64**(struct [uwbmactmsg](#) \*msg, int tag, uint64\_t value)

Add a unsigned 64-bit integer to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (uint64\_t) – Value to add.

### 2.1.158.1 Return

QERR\_SUCCESS or error.

### 2.1.159 uwbmactwriter\_add\_string

enum qerr **uwbmactwriter\_add\_string**(struct [uwbmactmsg](#) \*msg, int tag, const char \*value)

Add a string to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **value** (const char\*) – Pointer to the string to add.

### 2.1.159.1 Return

QERR\_SUCCESS or error.

### 2.1.160 uwbmactwriter\_add\_binary

enum qerr **uwbmactwriter\_add\_binary**(struct [uwbmactmsg](#) \*msg, int tag, const void \*data, int length)

Add a binary object to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **data** (const void\*) – Pointer to the object to add.
- **length** (int) – Length of the object to add.

### 2.1.160.1 Return

QERR\_SUCCESS or error.

### 2.1.161 uwbmactwriter\_start\_nested

enum qerr **uwbmactwriter\_start\_nested**(struct [uwbmactmsg](#) \*msg, int tag, struct [uwbmactmsg](#) \*nested)

Start adding a nested payload to the message.

#### Parameters

- **msg** (struct [uwbmactmsg](#)\*) – Message being written.
- **tag** (int) – Tag in the payload.
- **nested** (struct [uwbmactmsg](#)\*) – Pointer to the structure to keep internal data.

### 2.1.161.1 Return

QERR\_SUCCESS or error.

### 2.1.162 uwbmact\_writcr\_end\_nestcd

enum qerr uwbmact\_writcr\_end\_nestcd(struct uwbmact\_msg \*msg, struct uwbmact\_msg \*nestcd)  
Stop adding a nestcd payload to the message.

#### Parameters

- msg (struct uwbmact\_msg\*) – Message being writtcn.
- nestcd (struct uwbmact\_msg\*) – Pointer to the structure to keep internal data.

### 2.1.162.1 Return

QERR\_SUCCESS or error.

### 2.1.163 uwbmact\_writcr\_add\_singleton\_map

enum qerr uwbmact\_writcr\_add\_singleton\_map(struct uwbmact\_msg \*msg)  
Add a map containing a single pair to the message.

#### Parameters

- msg (struct uwbmact\_msg\*) – Message being writtcn.

### 2.1.163.1 Description

This function is only meant to be used at the beginning of an empty message, to produce a well-formed CBOR payload.

### 2.1.163.2 Return

QERR\_SUCCESS or error. QERR\_ENOTSUP if the message is not empty.

## 2.2 UWBMAC embedcd API

### 2.2.1 uwbmact\_device\_state\_report

void uwbmact\_device\_state\_report(struct ieee802154\_hw \*hw, enum uwbmact\_device\_state state)  
Report a device state change.

#### Parameters

- hw (struct ieee802154\_hw\*) – Pointer to MCPS hw instance.
- state (enum uwbmact\_device\_state) – New device state.

### 2.2.2 uwbmactestmode\_reply

enum qerr **uwbmactestmode\_reply**(struct ieee802154\_hw \*hw, struct sk\_buff \*reply)

Reply to a testmode call.

#### Parameters

- **hw** (struct ieee802154\_hw\*) – Pointer to MCPS hw instance.
- **reply** (struct sk\_buff\*) – Reply message.

#### 2.2.2.1 Return

QERR\_SUCCESS or error.

### 2.2.3 uwbmactestmode\_report

enum qerr **uwbmactestmode\_report**(struct ieee802154\_hw \*hw, uint32\_t port\_id, struct sk\_buff \*report)

Report an event.

#### Parameters

- **hw** (struct ieee802154\_hw\*) – Pointer to MCPS hw instance.
- **port\_id** (uint32\_t) – Port id to use to notify upper layer.
- **report** (struct sk\_buff\*) – Event report.

#### 2.2.3.1 Return

QERR\_SUCCESS or error.

### 2.2.4 uwbmactestmode\_reply

enum qerr **uwbmactestmode\_reply**(struct ieee802154\_hw \*hw, struct uwbmactestmode\_buf \*reply)

Reply to a testmode call.

#### Parameters

- **hw** (struct ieee802154\_hw\*) – Pointer to MCPS hw instance.
- **reply** (struct uwbmactestmode\_buf\*) – Reply message.

#### 2.2.4.1 NOTE

This method is only used by embedded flavor.



### 2.2.4.2 Return

QERR\_SUCCESS or error.

## 2.3 FiRa helper API

### 2.3.1 struct measurement\_sequence

struct **measurement\_sequence**  
Fira measurement sequence.

#### 2.3.1.1 Definition

```
struct measurement_sequence {
    size_t n_steps;
    struct fira_measurement_sequence_step steps[FIRA_MEASUREMENT_SEQUENCE_STEP_MAX];
}
```

#### 2.3.1.2 Members

**n\_steps**  
Number of steps in the schedule.

**steps**  
Steps of the schedule.

#### 2.3.1.3 Description

This structure contains the measurement sequence executed by the region.

### 2.3.2 struct session\_parameters

struct **session\_parameters**  
Fira session parameters.

#### 2.3.2.1 Definition

```
struct session_parameters {
    uint8_t device_type;
    uint8_t device_role;
    uint8_t ranging_round_usage;
    uint8_t sts_config;
    uint8_t multi_node_mode;
    uint16_t short_addr;
    uint16_t destination_short_address[FIRA_RESPONDERS_MAX];
    int n_destination_short_address;
    uint64_t time0_ns;
    uint32_t slot_duration_rstu;
}
```

(continues on next page)

(continued from previous page)

```

uint32_t round_duration_slots;
uint32_t block_duration_ms;
uint32_t block_stride_length;
bool round_hopping;
uint8_t priority;
uint8_t mac_address_mode;
uint8_t ranging_round_control;
uint8_t schedule_mode;
uint16_t max_number_of_measurements;
uint32_t max_rr_retry;
uint8_t channel_number;
uint8_t preamble_code_index;
uint8_t rframe_config;
uint8_t preamble_duration;
uint8_t sfd_id;
uint8_t psdu_data_rate;
uint8_t phr_data_rate;
union {
    struct {
        uint8_t static_sts_iv[FIRA_STATIC_STS_IV_SIZE];
        uint8_t vendor_id[FIRA_VENDOR_ID_SIZE];
    };
    uint8_t vupper64[FIRA_VUPPER64_SIZE];
};
uint8_t key_rotation;
uint8_t key_rotation_rate;
uint32_t sub_session_id;
uint8_t report_rssi;
uint8_t result_report_config;
uint8_t link_layer_mode;
uint8_t mac_fcs_type;
uint8_t prf_mode;
uint8_t cap_size_min;
uint8_t cap_size_max;
uint8_t number_of_sts_segments;
struct measurement_sequence meas_seq;
bool enable_diagnostics;
uint32_t diags_frame_reports_fields;
uint8_t sts_length;
uint8_t min_frames_per_rr;
uint16_t mtu_size;
uint8_t inter_frame_interval_ms;
uint8_t owr_aoa_measurement_ntf_period;
uint8_t session_info_ntf_config;
uint32_t near_proximity_config_cm;
uint32_t far_proximity_config_cm;
int32_t lower_aoa_bound_config_azimuth_2pi;
int32_t upper_aoa_bound_config_azimuth_2pi;
int16_t lower_aoa_bound_config_elevation_2pi;
int16_t upper_aoa_bound_config_elevation_2pi;
uint8_t termination_count;
}

```

### 2.3.2.2 Members

#### device\_type

Type of the device.

Possible values:

- 0x00: Controlee.
- 0x01: Controller.

See enum `quwbs_fbs_device_type`.

#### device\_role

Role played by the device.

Current implementation does not support decorrelation between the device's role and the device's type. The controller can only behave as the initiator and the controlee can only behave as responder.

Possible values:

- 0x00: Responder.
- 0x01: Initiator.
- 0x02: UT-Synchronization Anchor. *[Not supported in QM33 SDK]*
- 0x03: UT-Anchor. *[Not supported in QM33 SDK]*
- 0x04: UT-Tag. *[Not supported in QM33 SDK]*
- 0x05: Advertiser. *[Not supported in QM33 SDK]*
- 0x06: Observer. *[Not supported in QM33 SDK]*
- 0x07: DT-Anchor. *[Not supported in QM33 SDK]*
- 0x08: DT-Tag. *[Not supported in QM33 SDK]*

See enum `quwbs_fbs_device_role`.

#### ranging\_round\_usage

The ranging mode used during a round.

Possible values:

- 0x00: OWR UL-TDoA. *[Not supported in QM33 SDK]*
- 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: OWR DL-TDoA. *[Not supported in QM33 SDK]*
- 0x06: OWR for AoA. *[Not supported in QM33 SDK]*
- 0x07: eSS-TWR with Non-deferred Mode for Contention-based ranging. *[Not supported in QM33 SDK]*
- 0x08: aDS-TWR with Non-deferred Mode for Contention-based ranging. *[Not supported in QM33 SDK]*

See [enum `fira\_ranging\_round\_usage`](#).

### sts\_config

It configures how system shall generate the STS.

Possible values:

- 0x00: Static STS (default).
- 0x01: Dynamic STS. *[Not supported in QM33 SDK]*
- 0x02: Dynamic STS - Responder Specific Sub-session Key. *[Not supported in QM33 SDK]*
- 0x03: Provisioned STS.
- 0x04: Provisioned STS - Responder Specific Sub-session Key.

See enum `fbs_sts_mode`.

### multi\_node\_mode

The multi-node mode used during a round.

Possible values:

- 0x00: One-to-One.
- 0x01: One-to-Many.

See enum `struct fira_multi_node_mode`.

### short\_addr

Short address of the local device.

### destination\_short\_address

Array of destination short addresses.

### n\_destination\_short\_address

Number of destination short addresses.

### time0\_ns

Absolute value of the initiation time in nanoseconds.

### slot\_duration\_rstu

Duration of a slot in RSTU (1200RSTU=1ms).

### round\_duration\_slots

Number of slots per ranging round.

### block\_duration\_ms

Block size in unit of 1200 RSTU (same as ms).

### block\_stride\_length

Number of blocks to stride.

### round\_hopping

Enable FiRa round hopping.

### priority

Priority of the session.

### mac\_address\_mode

MAC addressing mode.

### ranging\_round\_control

Bit map of the following.

- b0: ranging result report phase is disabled(0) or enabled(1).
- b1: Control Message is sent in band(1) or not (0, not supported).

- b2: Control Message is sent separately(0) or piggybacked to RIM(1).

### schedule\_mode

Scheduling mode for the ranging session.

Possible values:

- 0x00 - Contention-based ranging. *[Not supported in QM33 SDK]*
- 0x01 - Time-scheduled ranging.
- 0x02 - Hybrid-based ranging. *[Not supported in QM33 SDK]*

### max\_number\_of\_measurements

Max number of measurements

### max\_rr\_retry

Number of failed ranging round attempts before stopping the session.

The value zero disable the feature.

### channel\_number

UWB channel for this session.

### preamble\_code\_index

UWB preamble code index.

Possible values:

- 9-24: BPRF
- 25-32: HPRF *[Not supported in QM33 SDK]*

### rframe\_config

The configuration of the frame.

see enum [struct fira\\_rframe\\_config](#).

### preamble\_duration

Possible values:

- 0x00: 32 symbols *[Not supported in QM33 SDK]*
- 0x01: 64 symbols (default)

See [enum fira\\_preamble\\_duration](#).

### sfd\_id

Possible values:

- 0 or 2 in BPRF
- 1-4 in HPRF *[Not supported in QM33 SDK]*

See [enum fira\\_sfd\\_id](#).

### psdu\_data\_rate

Possible values:

- 0: 6.81Mbps (default)
- 1: 7.80 Mbps *[Not supported in QM33 SDK]*
- 2: 27.2 Mbps *[Not supported in QM33 SDK]*
- 3: 31.2 Mbps *[Not supported in QM33 SDK]*

See [enum \*fira\\_psd\\_data\\_rate\*](#).

phr\_data\_rate

Possible values:

- 0: 850 kbit/s.
- 1: 6.81 Mbit/s.

See [enum \*fira\\_phr\\_data\\_rate\*](#).

**{unnamed\_union}**

anonymous

**{unnamed\_struct}**

anonymous

**static\_sts\_iv**

Static STS IV used in vUpper64.

**vendor\_id**

Vendor ID used in vUpper64.

**vupper64**

vUpper64 used during Static STS ranging.

**key\_rotation**

Enable/disable key rotation feature during Dynamic *[Not supported in QM33 SDK]* or Provisioned STS ranging.

Possible values:

- false: No key rotation.
- true: Key rotation enabled and period set by key\_rotation\_rate.

**key\_rotation\_rate**

Defines  $n$ , with  $2^n$  being the rotation rate of some keys used during Dynamic *[Not supported in QM33 SDK]* or Provisioned STS Ranging,  $n$  shall be in the range of  $0 \leq n \leq 15$ .

**sub\_session\_id**

Sub-session id for the controlee device. This configuration is applicable if STS\_CONFIG is set to 0x02 or 0x04.

**report\_rssi**

Activate rssi report

Possible values:

- 0: no rssi report
- 1: activate rssi report

**result\_report\_config**

Configure report information.

- b0: report ToF in result message, disabled(0) or enabled(1, default)
- b1: report AoA azimuth in result message, disabled (0, default) or enabled (1)
- b2: report AoA elevation in result message, disabled (0, default) or enabled (1)
- b3: report AoA FOM in result message, disabled (0, default) or enabled (1)

**link\_layer\_mode**

Used to define link layer behavior.

Possible values:

- 0x00: Bypass mode (default).

- 0x01: Connection less. *[Not supported in QM33 SDK]*
- Values 0x02 to 0xFF: RFU.

#### mac\_fcs\_type

**[NOT IMPLEMENTED]** The length of the Frame Check Sequence in the session.

Possible values:

- 0x00: CRC 16 (default)
- 0x01: CRC 32
- Values 0x02 to 0xFF: RFU

This parameter is not used in the current implementation.

See [enum \*fira\\_mac\\_fcs\\_type\*](#).

#### prf\_mode

Possible values:

- 0x00: 62.4 MHz PRF. BPRF mode (default)
- 0x01: 124.8 MHz PRF. HPRF mode. *[Not supported in QM33 SDK]*
- 0x02: 249.6 MHz PRF. HPRF mode with data rate 27.2 and 31.2 Mbps. *[Not supported in QM33 SDK]*

See [enum \*fira\\_prf\\_mode\*](#).

#### cap\_size\_min

*[Not supported in QM33 SDK]* Contention access period minimum value.

Default: 5

#### cap\_size\_max

*[Not supported in QM33 SDK]* Contention access period maximum value.

Default: round\_duration\_slots - 1

#### number\_of\_sts\_segments

**[NOT IMPLEMENTED]** Number of STS segments.

Possible values:

- 0x01: 1 STS Segment (default)
- 0x02: 2 STS Segments (HPRF only) *[Not supported in QM33 SDK]*
- 0x03: 3 STS Segments (HPRF only) *[Not supported in QM33 SDK]*
- 0x04: 4 STS Segments (HPRF only) *[Not supported in QM33 SDK]*
- Values 0x05 to 0xFF: RFU

This parameter is not used in the current implementation.

#### meas\_seq

*[Not supported in QM33 SDK]* Sequence of measurement sequence steps, configures the Antenna Flexibility features.

#### enable\_diagnostics

Activate the diagnostics for each round.

#### diags\_frame\_reports\_fields

Select the fields to activate in the frame reports stored in the diagnostics. Applicable only when enable\_diagnostics is set to true.

### sts\_length

Number of symbols in a STS segment.

Possible values:

- 0x00: 32 symbols
- 0x01: 64 symbols (default)
- 0x02: 128 symbols
- Values 0x03 to 0xFF: RFU

### min\_frames\_per\_rr

*[Not supported in QM33 SDK]* Minimal number of frames to be transmitted in OWR for AoA ranging round (block).

This parameter is only used in OWR for AoA Mode, see ranging\_round\_usage parameter

### mtu\_size

*[Not supported in QM33 SDK]* Maximum Transfer Unit, max size allowed to be transmitted in frame. The value shall be restricted to the maximum possible MTU size of the given frame which includes MHR, Variable IE size and FCS size.

### inter\_frame\_interval\_ms

*[Not supported in QM33 SDK]* Interval between RFRAMES transmitted in OWR for AoA (in units of 1200 RSTU)

This parameter is only used in OWR for AoA Mode, see ranging\_round\_usage parameter

### owr\_aoa\_measurement\_ntf\_period

*[Not supported in QM33 SDK]* Configure period of OWR for AoA measurement notifications.

Possible values:

- 0x00 = SESSION\_INFO\_NTF sent for every received OWR Advertisement frame (default)
- 0x01 = SESSION\_INFO\_NTF sent once after MIN\_FRAMES\_PER\_RR number of AoA measurements are aggregated

### session\_info\_ntf\_config

*[Not supported in QM33 SDK]* Configure session info notification.

Possible values:

- 0x00 = Disable session info notification (ntf)
- 0x01 = Enable session info notification (default)
- 0x02 = Enable session info ntf while inside proximity range
- 0x03 = Enable session info ntf while inside AoA upper and lower bounds
- 0x04 = Enable session info ntf while inside AoA upper and lower bounds as well as inside proximity range
- 0x05 = Enable session info ntf only when entering or leaving proximity range
- 0x06 = Enable session info ntf only when entering or leaving AoA upper and lower bounds
- 0x07 = Enable session info ntf only when entering or leaving AoA upper and lower bounds as well as entering or leaving proximity range

### near\_proximity\_config\_cm

*[Not supported in QM33 SDK]* Lower bound in cm above which the ranging notifications should be enabled.

Applicable when session\_info\_ntf\_config is set to 0x02, 0x04, 0x05 or 0x07. Should be less than or equal to far\_proximity\_config value.



#### far\_proximity\_config\_cm

[Not supported in QM33 SDK] Upper bound in cm above which the ranging notifications should be disabled.

Applicable when session\_info\_ntf\_config is set to 0x02, 0x04, 0x05 or 0x07. Should be greater than or equal to near\_proximity\_config value.

#### lower\_aoa\_bound\_config\_azimuth\_2pi

[Not supported in QM33 SDK] Represent degrees.

Applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07.

#### upper\_aoa\_bound\_config\_azimuth\_2pi

[Not supported in QM33 SDK] Represent degrees.

Applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07.

#### lower\_aoa\_bound\_config\_elevation\_2pi

[Not supported in QM33 SDK] Represent degrees

Applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07.

#### upper\_aoa\_bound\_config\_elevation\_2pi

[Not supported in QM33 SDK] Represent degrees.

Applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07.

#### termination\_count

in band termination attempt count.

### 2.3.2.3 Description

This structure contains the session parameters sent to the Fira region. Current implementation does not use all the parameters defined below.

### 2.3.3 struct controlee\_parameters

#### struct controlee\_parameters

Controlee parameters.

#### 2.3.3.1 Definition

```
struct controlee_parameters {
    uint32_t sub_session_id;
    uint16_t address;
    bool sub_session;
    uint8_t sub_session_key_len;
    uint8_t sub_session_key[FIRA_KEY_SIZE_MAX];
}
```

### 2.3.3.2 Members

- sub\_session\_id**  
Sub-session id for the controlee device.
- address**  
Controlee short address.
- sub\_session**  
To indicate whether or not the controlee has a sub-session.
- sub\_session\_key\_len**  
Size of the sub-session key, either 16 or 32 bytes.
- sub\_session\_key**  
Key used for sub-session's crypto calculations.

### 2.3.4 struct controlees\_parameters

struct **controlees\_parameters**  
Controlees list parameters.

#### 2.3.4.1 Definition

```
struct controlees_parameters {
    struct controlee_parameters controlees[FIRA_RESPONDERS_MAX];
    int n_controlees;
}
```

#### 2.3.4.2 Members

- controlees**  
List of controlees.
- n\_controlees**  
Number of controlees in the list.

### 2.3.5 struct dt\_anchor\_ranging\_round\_config

struct **dt\_anchor\_ranging\_round\_config**  
*[Not supported in QM33 SDK]* Configuration parameters of the ranging round for DT-Anchor.

#### 2.3.5.1 Definition

```
struct dt_anchor_ranging_round_config {
    uint8_t round_index;
    uint8_t acting_role;
    uint8_t n_responders;
    bool are_slots_present;
    uint16_t responders[FIRA_RESPONDERS_MAX];
    uint8_t slots[FIRA_RESPONDERS_MAX];
}
```

### 2.3.5.2 Members

#### round\_index

Round index.

#### acting\_role

Acting role (Initiator or Responder).

#### n\_responders

Total number of DT-Anchor Responders for this ranging round (applicable when the acting role is Initiator).

#### are\_slots\_present

Flag indicating if explicit slot scheduling will follow (applicable when the acting role is Initiator).

#### responders

Short addresses of DT-Anchor Responders for this ranging round (applicable when the acting role is Initiator).

#### slots

Slots for Response DTMs for consecutive DT-Anchor Responders (explicit slot scheduling, applicable when the acting role is Initiator).

### 2.3.6 struct update\_dt\_anchor\_ranging\_rounds\_cmd

struct update\_dt\_anchor\_ranging\_rounds\_cmd

[Not supported in QM33 SDK] Request to MAC with configuration of the ranging rounds for DT-Anchor.

#### 2.3.6.1 Definition

```
struct update_dt_anchor_ranging_rounds_cmd {
    int n_ranging_rounds;
    struct dt_anchor_ranging_round_config *ranging_rounds;
}
```

#### 2.3.6.2 Members

#### n\_ranging\_rounds

Total number of round configurations.

#### ranging\_rounds

Configuration parameters per ranging round.

### 2.3.7 struct update\_dt\_anchor\_ranging\_rounds\_rsp

struct update\_dt\_anchor\_ranging\_rounds\_rsp

[Not supported in QM33 SDK] Response from MAC including indexes of ranging rounds which failed to be configured for DT-Anchor.

### 2.3.7.1 Definition

```
struct update_dt_anchor_ranging_rounds_rsp {
    enum quwbs_fbs_status status;
    int n_round_indexes;
    uint8_t round_indexes[FIRA_DT_ANCHOR_MAX_ACTIVE_RR];
}
```

### 2.3.7.2 Members

#### status

Status of the config\_rsp.

#### n\_round\_indexes

Number of failing rounds.

#### round\_indexes

Failing round indexes.

## 2.3.8 struct dt\_tag\_ranging\_rounds\_config

struct dt\_tag\_ranging\_rounds\_config

[Not supported in QM33 SDK] Configuration parameters of the ranging rounds for DT-Tag.

### 2.3.8.1 Definition

```
struct dt_tag_ranging_rounds_config {
    uint8_t *round_indexes;
    int n_round_indexes;
}
```

### 2.3.8.2 Members

#### round\_indexes

Round indexes.

#### n\_round\_indexes

Total number of round indexes.

## 2.3.9 struct dt\_tag\_round\_indexes\_rsp

struct dt\_tag\_round\_indexes\_rsp

[Not supported in QM33 SDK] Indexes of ranging rounds which failed to be configured for DT-Tag.

### 2.3.9.1 Definition

```
struct dt_tag_round_indexes_rsp {
    enum quwbs_fbs_status status;
    int n_round_indexes;
    uint8_t round_indexes[FBS_DT_TAG_MAX_ACTIVE_RR];
}
```

### 2.3.9.2 Members

#### status

Status of the config\_rsp.

#### n\_round\_indexes

Total number of round indexes.

#### round\_indexes

Round indexes.

### 2.3.10 enum aoa\_measurements\_index

enum aoa\_measurements\_index

AOA measurements.

### 2.3.10.1 Definition

```
enum aoa_measurements_index {
    FIRA_HELPER_AOA_AZIMUTH,
    FIRA_HELPER_AOA,
    FIRA_HELPER_AOA_ELEVATION,
    FIRA_HELPER_AOA_NB
};
```

### 2.3.10.2 Constants

#### FIRA\_HELPER\_AOA\_AZIMUTH

Retrieve AOA azimuth.

#### FIRA\_HELPER\_AOA

Retrieve AOA (same as azimuth).

#### FIRA\_HELPER\_AOA\_ELEVATION

Retrieve AOA elevation. *[Not supported in QM33 SDK]*

#### FIRA\_HELPER\_AOA\_NB

Enum members number.

### 2.3.11 struct aoa\_measurements

struct **aoa\_measurements**  
Fira Angle of Arrival measurements.

#### 2.3.11.1 Definition

```
struct aoa_measurements {
    uint8_t rx_antenna_pair;
    uint8_t aoa_fom_100;
    int16_t aoa_2pi;
    int16_t pdoa_2pi;
}
```

#### 2.3.11.2 Members

**rx\_antenna\_pair**  
Antenna pair index.

**aoa\_fom\_100**  
Estimation of local AoA reliability.

**aoa\_2pi**  
Estimation of reception angle.

**pdoa\_2pi**  
Estimation of reception phase difference.

#### 2.3.11.3 Description

Contains the different results of the AOA measurements.

### 2.3.12 struct fira\_twr\_measurements

struct **fira\_twr\_measurements**  
Fira ranging measurements.

#### 2.3.12.1 Definition

```
struct fira_twr_measurements {
    uint16_t short_addr;
    uint8_t status;
    uint8_t slot_index;
    bool stopped;
    uint8_t nlos;
    int32_t distance_cm;
    int16_t remote_aoa_azimuth_2pi;
    int16_t remote_aoa_elevation_pi;
    uint8_t remote_aoa_azimuth_fom_100;
    uint8_t remote_aoa_elevation_fom_100;
}
```

(continues on next page)

(continued from previous page)

```
struct aoa_measurements local_aoa_measurements[FIRA_HELPER_AOA_NB];
uint8_t rssi;
}
```

### 2.3.12.2 Members

#### **short\_addr**

Address of the participating device.

#### **status**

Zero if ok, or error reason.

#### **slot\_index**

In case of error, slot index where the error was detected.

#### **stopped**

Ranging was stopped as requested [controller only].

#### **nlos**

Indicates if the ranging measurement was in Line of Sight (LoS) or Non-Line of Sight (NLoS): 0x00 = LoS, 0x01 = NLoS, 0xFF = Unable to determine.

#### **distance\_cm**

Distance in cm.

#### **remote\_aoa\_azimuth\_2pi**

Estimation of reception angle in the azimuth of the participating device.

#### **remote\_aoa\_elevation\_pi**

Estimation of reception angle in the elevation of the participating device.

#### **remote\_aoa\_azimuth\_fom\_100**

Estimation of azimuth reliability of the participating device.

#### **remote\_aoa\_elevation\_fom\_100**

Estimation of elevation of the participating device.

#### **local\_aoa\_measurements**

Table of estimations of local measurements.

#### **rssi**

Computed rssi

### 2.3.13 struct fira\_ranging\_info

#### struct **fira\_ranging\_info**

Common information on the ranging result.

### 2.3.13.1 Definition

```
struct fira_ranging_info {
    uint32_t session_handle;
    uint32_t sequence_number;
    uint32_t block_index;
    uint32_t ranging_interval_ms;
    uint64_t timestamp_ns;
    struct diagnostic_info *diagnostic;
    struct uwbmac_buf *psdus_report;
}
```

### 2.3.13.2 Members

#### session\_handle

Session handle of the ranging result.

#### sequence\_number

Session notification counter.

#### block\_index

Current block index.

#### ranging\_interval\_ms

Current ranging interval in unit of ms. formula: (block size \* (stride + 1))

#### timestamp\_ns

**[NOT IMPLEMENTED]** Timestamp in nanoseconds in the CLOCK\_MONOTONIC time reference.

The current implementation does not provide any timestamp.

#### diagnostic

Debug informations

#### psdus\_report

Report containing all the psdus.

## 2.3.14 struct fira\_twr\_ranging\_results

### struct fira\_twr\_ranging\_results

Ranging results for Fira SS-TWR/DS-TWR.

### 2.3.14.1 Definition

```
struct fira_twr_ranging_results {
    struct fira_ranging_info *info;
    int n_measurements;
    struct fira_twr_measurements measurements[FIRA_RESPONDERS_MAX];
}
```



### 2.3.14.2 Members

#### info

Common information on this ranging.

#### n\_measurements

Number of measurements stored in the measurements table.

#### measurements

Ranging measurements information.

### 2.3.15 struct fira\_owr\_aoa\_measurements

struct **fira\_owr\_aoa\_measurements**

[Not supported in QM33 SDK] Ranging measurement for Fira OWR AoA.

#### 2.3.15.1 Definition

```
struct fira_owr_aoa_measurements {
    uint16_t short_addr;
    uint8_t status;
    uint8_t nlos;
    uint8_t frame_sequence_number;
    uint16_t block_index;
    struct aoa_measurements local_aoa_measurements[FIRA_HELPER_AOA_NB];
}
```

#### 2.3.15.2 Members

##### short\_addr

Address of the participating device.

##### status

Zero if ok, or error reason.

##### nlos

Indicates if the reception of the message was in Line of Sight (LoS) or Non-Line of Sight (NLoS): 0x00 = LoS, 0x01 = NLoS, 0xFF = Unable to determine.

##### frame\_sequence\_number

Sequence number as received in MHR.

##### block\_index

Block Index number as received in the OWR message from the Advertiser.

##### local\_aoa\_measurements

Table of estimations of local measurements.

### 2.3.16 struct fira\_owr\_aoa\_ranging\_results

struct fira\_owr\_aoa\_ranging\_results

[Not supported in QM33 SDK] Ranging results for Fira OWR AOA.

#### 2.3.16.1 Definition

```
struct fira_owr_aoa_ranging_results {
    struct fira_ranging_info *info;
    int n_measurements;
    struct fira_owr_aoa_measurements measurements[FIRA_OWR_AOA_MEASUREMENTS_MAX];
}
```

#### 2.3.16.2 Members

##### info

Common information on this ranging.

##### n\_measurements

Number of measurements stored in the measurements table.

##### measurements

Ranging measurements information.

### 2.3.17 struct fira\_ul\_tdoa\_ranging\_results

struct fira\_ul\_tdoa\_ranging\_results

[Not supported in QM33 SDK] Ranging results for FiRa UL-TDoA. Will be extended with implementation of UT-Anchor.

#### 2.3.17.1 Definition

```
struct fira_ul_tdoa_ranging_results {
    struct fira_ranging_info *info;
}
```

#### 2.3.17.2 Members

##### info

Common information on this ranging.

### 2.3.18 struct fira\_dl\_tdoa\_measurements

struct fira\_dl\_tdoa\_measurements

[Not supported in QM33 SDK] DL-TDOA ranging measurements.

#### 2.3.18.1 Definition

```
struct fira_dl_tdoa_measurements {
    struct fira_dl_tdoa_measurements *next;
    uint16_t short_addr;
    enum quwbs_fbs_status status;
    enum fira_owr_message_type message_type;
    enum fira_owr_dtm_timestamp_type tx_timestamp_type;
    enum fira_owr_dtm_timestamp_len tx_timestamp_len;
    enum fira_owr_dtm_timestamp_len rx_timestamp_len;
    enum fira_dt_location_coord_system_type anchor_location_type;
    bool anchor_location_present;
    uint8_t active_ranging_round_indexes_len;
    uint8_t round_index;
    uint16_t block_index;
    int16_t local_aoa_azimuth_2pi;
    int16_t local_aoa_elevation_2pi;
    uint8_t local_aoa_azimuth_fom;
    uint8_t local_aoa_elevation_fom;
    uint8_t rx_rssi;
    uint8_t nlos;
    uint16_t local_cfo;
    uint16_t remote_cfo;
    uint64_t tx_timestamp_rctu;
    uint64_t rx_timestamp_rctu;
    uint32_t initiator_reply_time_rctu;
    uint32_t responder_reply_time_rctu;
    uint8_t anchor_location[FIRA_DL_TDOA_ANCHOR_LOCATION_SIZE_MAX];
    uint8_t active_ranging_round_indexes[FIRA_DL_TDOA_MAX_ROUNDS_PER_BLOCK];
    uint16_t initiator_responder_tof_rctu;
}
```

#### 2.3.18.2 Members

##### next

Pointer on next measurements if there is one, or NULL.

##### short\_addr

Address of the participating device.

##### status

Zero if ok, or error reason. See enum quwbs\_fbs\_status for all error codes.

##### message\_type

Type of the message which has been received.

##### tx\_timestamp\_type

Type of the TX timestamp (local time base vs common time base) included in the received message.

**tx\_timestamp\_len**

Length of the TX timestamp (40-bit vs 64-bit) included in the received message.

**rx\_timestamp\_len**

Length of the TX timestamp (40-bit vs 64-bit) calculated during the reception of the received message.

**anchor\_location\_type**

Type of the coordinate system of DT-Anchor location (0: WGS84, 1: relative) (if included).

**anchor\_location\_present**

True when the information about DT-Anchor location is included in the measurement, false otherwise.

**active\_ranging\_round\_indexes\_len**

Number of active ranging round indexes included in the measurement.

**round\_index**

Index of the current ranging round.

**block\_index**

Index of the current ranging block.

**local\_aoa\_azimuth\_2pi**

AoA Azimuth in degrees measured by the DT-Tag during the reception (encoded as Q9.7).

**local\_aoa\_elevation\_2pi**

AoA Elevation in degrees measured by the DT-Tag during the reception (encoded as Q9.7).

**local\_aoa\_azimuth\_fom**

Reliability of the estimated AoA Azimuth measured by the DT-Tag during the reception (range: 0-100).

**local\_aoa\_elevation\_fom**

Reliability of the estimated AoA Elevation measured by the DT-Tag during the reception (range: 0-100).

**rx\_rssi**

RSSI measured by the DT-Tag during the reception (encoded as Q7.1).

**nlos**

Indicates if the reception of the message was in Line of Sight (LoS) or Non-Line of Sight (NLoS): 0x00 = LoS, 0x01 = NLoS, 0xFF = Unable to determine.

**local\_cfo**

Clock frequency offset measured locally with respect to the DT-Anchor that sent the message received (encoded as Q6.10).

**remote\_cfo**

Clock frequency offset of a Responder DT-Anchor with respect to the Initiator DT-Anchor of the ranging round as included in the received message (encoded as Q6.10).

**tx\_timestamp\_rctu**

TX timestamp included in the received message (unit: RCTU).

**rx\_timestamp\_rctu**

RX timestamp calculated during the reception of the received message (unit: RCTU).

**initiator\_reply\_time\_rctu**

Reply time of the Initiator DT-Anchor measured between the reception of Response DTM and the transmission of Final DTM (used only in DS-TWR, unit: RCTU).

**responder\_reply\_time\_rctu**

Reply time of the Responder DT-Anchor measured between the reception of Poll DTM and the transmission of Response DTM (unit: RCTU).

**anchor\_location**

Location coordinates of DT-Anchor that sent the message received.

### active\_ranging\_round\_indexes

List of active ranging round indexes in which the DT-Anchor that sent the message received participates.

### initiator\_responder\_tof\_rctu

Time of Flight measured between the Initiator DT-Anchor and the Responder DT-Anchor (for SS-TWR it's calculated by Initiator DT-Anchor and included in Poll DTM and for DS-TWR it's calculated by Responder DT-Anchor and included in Response DTM, unit: RCTU)

## 2.3.19 struct fira\_dl\_tdoa\_ranging\_results

struct **fira\_dl\_tdoa\_ranging\_results**

[Not supported in QM33 SDK] Ranging results for Fira DL-TDOA.

### 2.3.19.1 Definition

```
struct fira_dl_tdoa_ranging_results {
    struct fira_ranging_info *info;
    int n_measurements;
    struct fira_dl_tdoa_measurements *measurements;
}
```

### 2.3.19.2 Members

#### info

Common information on this ranging.

#### n\_measurements

Number of measurements stored in the measurements table.

#### measurements

Linked list of the DL-TDOA measurements or NULL.

## 2.3.20 struct controlee\_status

struct **controlee\_status**

Controlee addition/deletion notification status.

### 2.3.20.1 Definition

```
struct controlee_status {
    uint16_t short_address;
    uint32_t sub_session_id;
    uint8_t status_code;
}
```

### 2.3.20.2 Members

#### short\_address

Controlee short address.

#### sub\_session\_id

Sub-session id of the current controlee.

#### status\_code

See [enum fira\\_multicast\\_update\\_status](#).

### 2.3.21 struct fira\_session\_multicast\_list\_ntf\_content

struct **fira\_session\_multicast\_list\_ntf\_content**

Necessary content to fill a session update controller multicast list notification.

#### 2.3.21.1 Definition

```
struct fira_session_multicast_list_ntf_content {
    uint32_t session_handle;
    uint8_t remaining_multicast_list_size;
    struct controlee_status controlees[FIRA_RESPONDERS_MAX];
    uint8_t n_controlees;
}
```

#### 2.3.21.2 Members

##### session\_handle

Session handle.

##### remaining\_multicast\_list\_size

New available size in the multicast list. Maximum size is defined by FIRA\_RESPONDERS\_MAX.

##### controlees

List of controlees with their corresponding multicast list update status.

##### n\_controlees

Number of controlees in the previous list.

### 2.3.22 struct data\_credit\_ntf\_content

struct **data\_credit\_ntf\_content**

[Not supported in QM33 SDK] Fira DATA\_CREDIT\_NFT content.

### 2.3.22.1 Definition

```
struct data_credit_ntf_content {
    uint32_t session_handle;
    uint8_t credit_avail;
}
```

### 2.3.22.2 Members

#### session\_handle

Session handle.

#### credit\_avail

Credit availability 0x00 Credit is not available 0x01 Credit is available

### 2.3.23 struct data\_transfer\_status\_ntf\_content

struct data\_transfer\_status\_ntf\_content

[Not supported in QM33 SDK] Fira SESSION\_DATA\_TRANSFER\_STATUS\_NTF content.

### 2.3.23.1 Definition

```
struct data_transfer_status_ntf_content {
    uint32_t session_handle;
    uint16_t uci_seq_nr;
    uint8_t status;
    uint8_t tx_count;
}
```

### 2.3.23.2 Members

#### session\_handle

Session handle.

#### uci\_seq\_nr

The Sequence Number identifying the UCI Data Message this NTF is for.

#### status

Status Code. See enum uci\_data\_transfer\_status\_code.

#### tx\_count

Indicates the number of times Application Data with the same UCI Sequence Number has been transmitted.

### 2.3.24 struct data\_message\_content

struct data\_message\_content

[Not supported in QM33 SDK] Fira DATA\_MESSAGE\_SND and DATA\_MESSAGE\_RCV content.

#### 2.3.24.1 Definition

```
struct data_message_content {
    uint32_t session_handle;
    uint16_t short_addr;
    uint16_t uci_seq_nr;
    uint8_t status;
    uint8_t data_segment_info;
    uint16_t data_len;
    uint8_t *data;
}
```

#### 2.3.24.2 Members

##### session\_handle

Session handle.

##### short\_addr

Short\_addr. For DATA\_MESSAGE\_SND: short\_addr of the Application Data recipient. For DATA\_MESSAGE\_RCV: short\_addr of the sender of the Application Data.

##### uci\_seq\_nr

Sequence Number for the UCI Data Message.

##### status

Status. Applicable only in case of DATA\_MESSAGE\_RCV. 0x00 STATUS\_SUCCESS 0x01 STATUS\_ERROR 0x02 STATUS\_UNKNOWN

##### data\_segment\_info

See [enum fira\\_data\\_segment\\_info](#).

##### data\_len

Length of the data.

##### data

Application Data.

### 2.3.25 struct fira\_hus\_controller\_phase\_config

struct fira\_hus\_controller\_phase\_config

[Not supported in QM33 SDK] Phase configuration parameters used by a HUS controller device to bind a secondary session to a primary session.



### 2.3.25.1 Definition

```
struct fira_hus_controller_phase_config {
    uint32_t session_id;
    uint16_t start_slot_index;
    uint16_t end_slot_index;
    uint16_t controller_short_addr;
    uint8_t control;
}
```

### 2.3.25.2 Members

#### session\_id

Session id of the targeted phase.

#### start\_slot\_index

Slot index of the first slot of the phase.

#### end\_slot\_index

Slot index of the last slot of the phase.

#### controller\_short\_addr

MAC short address of the controller of the phase.

#### control

Information about the current phase. b0: 0 = Short addressing mode, 1 = Extended addressing mode. b1: 0 = CAP phase, 1 = CFP phase.

### 2.3.26 struct fira\_hus\_controller\_config\_cmd

struct fira\_hus\_controller\_config\_cmd

[Not supported in QM33 SDK] List of secondary sessions to bind to a primary session. Only applicable to a HUS controller device.

### 2.3.26.1 Definition

```
struct fira_hus_controller_config_cmd {
    uint64_t update_time_us;
    uint32_t session_handle;
    struct fira_hus_controller_phase_config *phase_list;
    uint8_t number_of_phases;
}
```

### 2.3.26.2 Members

#### update\_time\_us

Time in microseconds when this configuration shall be applied.

#### session\_handle

Handle of the targeted session.

#### phase\_list

List of CAP or CFP phases.

#### number\_of\_phases

Number of CAP or CFP phases in the HUS ranging round.

### 2.3.27 struct fira\_hus\_controlee\_phase\_config

struct fira\_hus\_controlee\_phase\_config

[Not supported in QM33 SDK] Phase configuration parameters used by a HUS controlee device to bind a secondary session to a primary session.

#### 2.3.27.1 Definition

```
struct fira_hus_controlee_phase_config {
    uint32_t session_handle;
}
```

#### 2.3.27.2 Members

#### session\_handle

Session handle of the targeted phase.

### 2.3.28 struct fira\_hus\_controlee\_config\_cmd

struct fira\_hus\_controlee\_config\_cmd

[Not supported in QM33 SDK] Status of the configuration command binding secondary sessions to a primary session.

#### 2.3.28.1 Definition

```
struct fira_hus_controlee_config_cmd {
    uint32_t session_handle;
    struct fira_hus_controlee_phase_config *phase_list;
    uint8_t number_of_phases;
}
```

### 2.3.28.2 Members

#### session\_handle

Handle of the targeted session.

#### phase\_list

List of CAP or CFP phases.

#### number\_of\_phases

Number of CAP or CFP phases in the HUS ranging round.

### 2.3.29 enum fira\_helper\_cb\_type

#### enum fira\_helper\_cb\_type

Callback type. See [struct fira\\_helper\\_notification\\_cb\\_t](#)

#### 2.3.29.1 Definition

```
enum fira_helper_cb_type {
    FIRA_HELPER_CB_TYPE_UNSPEC,
    FIRA_HELPER_CB_TYPE_TWR_RANGE_NTF,
    FIRA_HELPER_CB_TYPE_OWR_AOA_NTF,
    FIRA_HELPER_CB_TYPE_UL_TDOA_NTF,
    FIRA_HELPER_CB_TYPE_DL_TDOA_NTF,
    FIRA_HELPER_CB_TYPE_SESSION_DATA_CREDIT_NTF,
    FIRA_HELPER_CB_TYPE_SESSION_DATA_TRANSFER_STATUS_NTF,
    FIRA_HELPER_CB_TYPE_DATA_MESSAGE_RCV,
    FIRA_HELPER_CB_TYPE_SESSION_STATUS_NTF,
    FIRA_HELPER_CB_TYPE_SESSION_UPDATE_CONTROLLER_MULTICAST_LIST_NTF
};
```

#### 2.3.29.2 Constants

##### FIRA\_HELPER\_CB\_TYPE\_UNSPEC

unspecified callback type

##### FIRA\_HELPER\_CB\_TYPE\_TWR\_RANGE\_NTF

Callback content is struct fira\_twr\_ranging\_results\*.

##### FIRA\_HELPER\_CB\_TYPE\_OWR\_AOA\_NTF

[Not supported in QM33 SDK] Callback content is struct fira\_owr\_aoa\_ranging\_results\*.

##### FIRA\_HELPER\_CB\_TYPE\_UL\_TDOA\_NTF

[Not supported in QM33 SDK] Callback content is struct fira\_ul\_tdoa\_ranging\_results\*.

##### FIRA\_HELPER\_CB\_TYPE\_DL\_TDOA\_NTF

[Not supported in QM33 SDK] Callback content is struct fira\_dl\_tdoa\_ranging\_results\*.

##### FIRA\_HELPER\_CB\_TYPE\_SESSION\_DATA\_CREDIT\_NTF

[Not supported in QM33 SDK] Callback content is struct data\_credit\_ntf\_content\*.

##### FIRA\_HELPER\_CB\_TYPE\_SESSION\_DATA\_TRANSFER\_STATUS\_NTF

[Not supported in QM33 SDK] Callback content is struct data\_transfer\_status\_ntf\_content\*.

##### FIRA\_HELPER\_CB\_TYPE\_DATA\_MESSAGE\_RCV

[Not supported in QM33 SDK] Callback content is struct data\_message\_content\*.

## FIRA\_HELPER\_CB\_TYPE\_SESSION\_STATUS\_NTF

Callback content is struct fbs\_helper\_session\_status\_ntf\*.

## FIRA\_HELPER\_CB\_TYPE\_SESSION\_UPDATE\_CONTROLLER\_MULTICAST\_LIST\_NTF

Callback content is struct fira\_session\_multicast\_list\_ntf\_content\*.

### 2.3.30 typedef fira\_helper\_notification\_cb\_t

void **fira\_helper\_notification\_cb\_t**(enum [fira\\_helper\\_cb\\_type](#) cb\_type, const void \*content, void \*user\_data)  
Notification callback type.

#### Parameters

- **cb\_type** (enum [fira\\_helper\\_cb\\_type](#)) – Type of callback depending on exact message to be sent.
- **content** (const void\*) – Generic content with results depending on the cb\_type.
- **user\_data** (void\*) – User data pointer given to fira\_helper\_open.

#### 2.3.30.1 Description

See enum fira\_helper\_cb\_type documentation for more information on the content.

### 2.3.31 fira\_helper\_open

enum qerr **fira\_helper\_open**(struct fira\_context \*ctx, struct uwbmactext \*uwbmactext, [fira\\_helper\\_notification\\_cb\\_t](#) notification\_cb, const char \*scheduler, int region\_id, void \*user\_data)

Initialize the fira helper context.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **uwbmactext** (struct uwbmactext\*) – UWB MAC context.
- **notification\_cb** ([fira\\_helper\\_notification\\_cb\\_t](#)) – Callback function for notifications feedback.
- **scheduler** (const char\*) – Scheduler name to use with the region.
- **region\_id** (int) – Region identifier to associate with the region.
- **user\_data** (void\*) – User data pointer to give back in callback.

#### 2.3.31.1 NOTE

This function must be called first. fira\_helper\_close must be called at the end of the application to ensure resources are freed. The channel will be managed by the helper, this means you should neither use uwbmactext\_channel\_create nor uwbmactext\_channel\_release.

### 2.3.31.2 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.32 `fira_helper_close`

void **fira\_helper\_close**(struct fira\_context \*ctx)

De-initialize the fira helper context.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.

### 2.3.33 `fira_helper_set_device_status_cb`

enum qerr **fira\_helper\_set\_device\_status\_cb**(struct fira\_context \*ctx, fbs\_helper\_device\_status\_ntf\_cb cb)

Set the device status callback.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **cb** (fbs\_helper\_device\_status\_ntf\_cb) – Callback for all device status notifications.

#### 2.3.33.1 NOTE

Temporary api before we inverse dependency with fbs\_helper. Once this is done client will have to directly use fbs\_helper\_set\_device\_status\_ntf\_cb.

#### 2.3.33.2 Return

QERR\_SUCCESS on success, on error otherwise.

### 2.3.34 `fira_helper_set_scheduler`

enum qerr **fira\_helper\_set\_scheduler**(struct fira\_context \*ctx)

Set the scheduler and open the MAC region.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.

#### 2.3.34.1 NOTE

This function must be called while the UWB MAC is stopped.

### 2.3.34.2 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.35 `fira_helper_get_capabilities`

enum qerr **fira\_helper\_get\_capabilities**(struct fira\_context \*ctx, struct fira\_capabilities \*capabilities)  
Get the FiRa region capabilities.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **capabilities** (struct fira\_capabilities\*) – Fira capabilities.

#### 2.3.35.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.36 `fira_helper_init_session`

enum qerr **fira\_helper\_init\_session**(struct fira\_context \*ctx, uint32\_t session\_id, enum quwbs\_fbs\_session\_type session\_type, struct fbs\_session\_init\_rsp \*rsp)  
Initialize a fira session.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_id** (uint32\_t) – Session identifier.
- **session\_type** (enum quwbs\_fbs\_session\_type) – Session type value.
- **rsp** (struct fbs\_session\_init\_rsp\*) – Session init response message information.

#### 2.3.36.1 Description

This function must be called first to create and initialize the fira session.

#### 2.3.36.2 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.37 `fira_helper_start_session`

enum qerr **fira\_helper\_start\_session**(struct fira\_context \*ctx, uint32\_t session\_handle)  
Start a fira session.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

### 2.3.37.1 Description

This function must be called after fira session was initialized.

### 2.3.37.2 Return

QERR\_SUCCESS on success, an error otherwise.

## 2.3.38 fira\_helper\_stop\_session

enum qerr **fira\_helper\_stop\_session**(struct fira\_context \*ctx, uint32\_t session\_handle)

Stop a fira session.

### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

### 2.3.38.1 Description

This function stop the session ranging.

### 2.3.38.2 Return

QERR\_SUCCESS on success, an error otherwise.

## 2.3.39 fira\_helper\_deinit\_session

enum qerr **fira\_helper\_deinit\_session**(struct fira\_context \*ctx, uint32\_t session\_handle)

Deinitialize a fira session.

### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

### 2.3.39.1 Description

This function is called to free all memory allocated by the session.

### 2.3.39.2 Return

**QERR\_SUCCESS or QERR\_EBUSY on success, an error otherwise.**

The QERR\_EBUSY is used to indicate that an active session has been deinit.

### 2.3.40 `fira_helper_get_session_parameters`

enum qerr **fira\_helper\_get\_session\_parameters**(struct fira\_context \*ctx, uint32\_t session\_handle, struct [session\\_parameters](#) \*session\_params)

Get session parameters.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **session\_params** (struct [session\\_parameters](#)\*) – Session parameters.

#### 2.3.40.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.41 `fira_helper_session_get_count`

enum qerr **fira\_helper\_session\_get\_count**(struct fira\_context \*ctx, int \*count)

Get sessions count, the number of active and inactive sessions.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **count** (int\*) – Session count.

#### 2.3.41.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.42 `fira_helper_session_get_state`

enum qerr **fira\_helper\_session\_get\_state**(struct fira\_context \*ctx, uint32\_t session\_handle, int \*state)

Get session state.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **state** (int\*) – Session state.



### 2.3.42.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.43 `fira_helper_get_ranging_count`

```
enum qerr fira_helper_get_ranging_count(struct fira_context *ctx, uint32_t session_handle, struct
                                         fbs_ranging_count_rsp *rsp)
```

Get ranging count, the number of times ranging has been attempted during the session.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **rsp** (struct fbs\_ranging\_count\_rsp\*) – Ranging count response message information.

### 2.3.43.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.44 `fira_helper_add_controlee`

```
int fira_helper_add_controlee(struct fira_context *ctx, uint32_t session_handle, const struct
                               controlee\_parameters *controlee)
```

Add one controlee to a specific session.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **controlee** (const struct [controlee\\_parameters](#)\*) – Controlee to add.

### 2.3.44.1 Return

0 or positive value on success, negative value on error.

### 2.3.45 `fira_helper_delete_controlee`

```
int fira_helper_delete_controlee(struct fira_context *ctx, uint32_t session_handle, const struct
                                   controlee\_parameters *controlee)
```

Delete one controlee from a specific session.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **controlee** (const struct [controlee\\_parameters](#)\*) – Controlee to delete.

### 2.3.45.1 Return

0 or positive value on success, negative value on error.

### 2.3.46 `fira_helper_get_controlees`

enum qerr `fira_helper_get_controlees`(struct fira\_context \*ctx, uint32\_t session\_handle, struct [controlees\\_parameters](#) \*controlees)

Get controlees list.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `controlees` (struct [controlees\\_parameters](#)\*) – List of controlees to write.

### 2.3.46.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.47 `fira_helper_get_controlees_count`

enum qerr `fira_helper_get_controlees_count`(struct fira\_context \*ctx, uint32\_t session\_handle, int \*count)

Get number of currently known controlees.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `count` (int\*) – Number of controlees known.

### 2.3.47.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.48 `fira_helper_data_message_send`

enum qerr `fira_helper_data_message_send`(struct fira\_context \*ctx, uint32\_t session\_handle, const struct [data\\_message\\_content](#) \*data\_content)

*[Not supported in QM33 SDK]* Send data message.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `data_content` (const struct [data\\_message\\_content](#)\*) – Data message.

### 2.3.48.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.49 `fira_helper_set_session_device_type`

enum qerr `fira_helper_set_session_device_type`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t device\_type)

Sets the device type.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `device_type` (uint8\_t) – 0 - CONTROLEE, 1 - CONTROLLER.

### 2.3.49.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.50 `fira_helper_set_session_dl_tdoa_time_reference_anchor`

enum qerr `fira_helper_set_session_dl_tdoa_time_reference_anchor`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t global\_time)

*[Not supported in QM33 SDK]* Set or reset the time reference anchor.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `global_time` (uint8\_t) – 0 - DISABLE, 1 - Set DT-ANCHOR as global time reference and sets its cost metric to zero.

### 2.3.50.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.51 `fira_helper_set_session_dl_tdoa_responder_tof`

enum qerr `fira_helper_set_session_dl_tdoa_responder_tof`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t responder\_tof)

*[Not supported in QM33 SDK]* Include or not the responder tof.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `responder_tof` (uint8\_t) – 0 - Do not include, 1 - include the estimated Responder ToF Result in a Response DTM.

### 2.3.51.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.52 `fira_helper_set_session_dl_tdoa_ranging_method`

enum qerr `fira_helper_set_session_dl_tdoa_ranging_method`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t method)

*[Not supported in QM33 SDK]* Set dl-tdoa ranging method.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `method` (uint8\_t) – 0 - SS-TWR, 1 - DS-TWR.

### 2.3.52.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.53 `fira_helper_set_session_dl_tdoa_tx_timestamp_type`

enum qerr `fira_helper_set_session_dl_tdoa_tx_timestamp_type`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t type)

*[Not supported in QM33 SDK]* Configure tx timestamp type.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `type` (uint8\_t) – Timestamp type.

### 2.3.53.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.54 `fira_helper_set_session_dl_tdoa_tx_timestamp_len`

enum qerr `fira_helper_set_session_dl_tdoa_tx_timestamp_len`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t len)

*[Not supported in QM33 SDK]* Configure tx timestamp length.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `len` (uint8\_t) – Timestamp length.

### 2.3.54.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.55 `fira_helper_set_session_dl_tdoa_hop_count`

enum qerr `fira_helper_set_session_dl_tdoa_hop_count`(struct `fira_context` \*ctx, uint32\_t session\_handle, uint8\_t count)

*[Not supported in QM33 SDK]* Set dl-tdoa hop count presence.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `count` (uint8\_t) – 0 - DEACTIVATED, 1 - ACTIVATED.

### 2.3.55.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.56 `fira_helper_set_session_dl_tdoa_anchor_cfo`

enum qerr `fira_helper_set_session_dl_tdoa_anchor_cfo`(struct `fira_context` \*ctx, uint32\_t session\_handle, uint8\_t cfo)

*[Not supported in QM33 SDK]* Set dl-tdoa presence of cfo.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `cfo` (uint8\_t) – 0 - DEACTIVATED, 1 - ACTIVATED.

### 2.3.56.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.57 `fira_helper_set_session_dl_tdoa_anchor_location_presence`

enum qerr `fira_helper_set_session_dl_tdoa_anchor_location_presence`(struct `fira_context` \*ctx, uint32\_t session\_handle, uint8\_t presence)

*[Not supported in QM33 SDK]* Set dl-tdoa presence of dt-anchor location.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `presence` (uint8\_t) – presence of the information about DT-Anchor location in DTMs 0 - DEACTIVATED, 1 - ACTIVATED.

### 2.3.57.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.58 `fira_helper_set_session_dl_tdoa_anchor_location`

enum qerr `fira_helper_set_session_dl_tdoa_anchor_location`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t len, uint8\_t \*data)

*[Not supported in QM33 SDK]* Set dl-tdoa dt-anchor location.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `len` (uint8\_t) – Length of the array according to the location type.
- `data` (uint8\_t\*) – data array that represents location.

### 2.3.58.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.59 `fira_helper_set_session_dl_tdoa_anchor_location_type`

enum qerr `fira_helper_set_session_dl_tdoa_anchor_location_type`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t type)

*[Not supported in QM33 SDK]* Set dl-tdoa type of dt-anchor location.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `type` (uint8\_t) – Type of the DT-Anchor location format: 0 - WGS84, 1 - relative.

### 2.3.59.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.60 `fira_helper_set_session_dl_tdoa_active_ranging_rounds`

enum qerr `fira_helper_set_session_dl_tdoa_active_ranging_rounds`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t rounds)

*[Not supported in QM33 SDK]* Set dl-tdoa presence of ranging rounds.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `rrounds` (uint8\_t) – 0 - deactivated, 1 - activated.

### 2.3.60.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.61 `fira_helper_set_session_dl_tdoa_block_skipping`

enum qerr `fira_helper_set_session_dl_tdoa_block_skipping`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t number)

*[Not supported in QM33 SDK]* Set dl-tdoa number of blocks that shall be skipped between 2 active ranging blocks.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `number` (uint8\_t) – Number of blocks to be skipped by the dt-tag.

#### 2.3.61.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.62 `fira_helper_set_session_report_psdus`

enum qerr `fira_helper_set_session_report_psdus`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t active)

*[Not supported in QM33 SDK]* Enable/disable psdus report.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `active` (uint8\_t) – True to enable psdus are reported, false otherwise.

#### 2.3.62.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.63 `fira_helper_get_session_in_band_termination_attempt_count`

enum qerr `fira_helper_get_session_in_band_termination_attempt_count`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*termination\_count)

Get the in band termination attempt count.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `termination_count` (uint8\_t\*) – Termination\_count.

### 2.3.63.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.64 `fira_helper_get_session_dl_tdoa_time_reference_anchor`

enum qerr `fira_helper_get_session_dl_tdoa_time_reference_anchor`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*global\_time)

*[Not supported in QM33 SDK]* Get the time reference anchor.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `global_time` (uint8\_t\*) – Time reference anchor.

### 2.3.64.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.65 `fira_helper_get_session_dl_tdoa_responder_tof`

enum qerr `fira_helper_get_session_dl_tdoa_responder_tof`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*responder\_tof)

*[Not supported in QM33 SDK]* Get the responder tof config.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `responder_tof` (uint8\_t\*) – Responder time of flight.

### 2.3.65.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.66 `fira_helper_get_session_dl_tdoa_ranging_method`

enum qerr `fira_helper_get_session_dl_tdoa_ranging_method`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*method)

*[Not supported in QM33 SDK]* Get dl-tdoa ranging method.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `method` (uint8\_t\*) – 0 - SS-TWR, 1 - DS-TWR.



### 2.3.66.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.67 `fira_helper_get_session_dl_tdoa_tx_timestamp_type`

enum qerr `fira_helper_get_session_dl_tdoa_tx_timestamp_type`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*type)

*[Not supported in QM33 SDK]* Get tx timestamp type.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `type` (uint8\_t\*) – Timestamp type.

### 2.3.67.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.68 `fira_helper_get_session_dl_tdoa_tx_timestamp_len`

enum qerr `fira_helper_get_session_dl_tdoa_tx_timestamp_len`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*len)

*[Not supported in QM33 SDK]* Get tx timestamp length.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `len` (uint8\_t\*) – Timestamp length.

### 2.3.68.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.69 `fira_helper_get_session_dl_tdoa_hop_count`

enum qerr `fira_helper_get_session_dl_tdoa_hop_count`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*count)

*[Not supported in QM33 SDK]* Get dl-tdoa hop count presence.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `count` (uint8\_t\*) – 0 - DEACTIVATED, 1 - ACTIVATED.

### 2.3.69.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.70 `fira_helper_get_session_dl_tdoa_anchor_cfo`

enum qerr `fira_helper_get_session_dl_tdoa_anchor_cfo`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*cfo)

*[Not supported in QM33 SDK]* Get dl-tdoa presence of cfo.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **cfo** (uint8\_t\*) – 0 - DEACTIVATED, 1 - ACTIVATED.

#### 2.3.70.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.71 `fira_helper_get_session_dl_tdoa_anchor_location_presence`

enum qerr `fira_helper_get_session_dl_tdoa_anchor_location_presence`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*presence)

*[Not supported in QM33 SDK]* Get dl-tdoa presence of dt-anchor location.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **presence** (uint8\_t\*) – presence of the information about DT-Anchor location in DTMs 0 - DEACTIVATED, 1 - ACTIVATED.

#### 2.3.71.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.72 `fira_helper_get_session_dl_tdoa_anchor_location_type`

enum qerr `fira_helper_get_session_dl_tdoa_anchor_location_type`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*type)

*[Not supported in QM33 SDK]* Get dl-tdoa type of dt-anchor location.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **type** (uint8\_t\*) – Type of the DT-Anchor location format: 0 - WGS84, 1 - RELATIVE.

### 2.3.72.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.73 `fira_helper_get_session_dl_tdoa_anchor_location`

enum qerr `fira_helper_get_session_dl_tdoa_anchor_location`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t len, uint8\_t \*data)

*[Not supported in QM33 SDK]* Get dl-tdoa dt-anchor location.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **len** (uint8\_t) – Length of the array according to the location type.
- **data** (uint8\_t\*) – data array that represents location.

### 2.3.73.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.74 `fira_helper_get_session_dl_tdoa_active_ranging_rounds`

enum qerr `fira_helper_get_session_dl_tdoa_active_ranging_rounds`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*rrounds)

*[Not supported in QM33 SDK]* Get dl-tdoa presence of ranging rounds.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **rrounds** (uint8\_t\*) – 0 - DEACTIVATED, 1 - ACTIVATED.

### 2.3.74.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.75 `fira_helper_get_session_dl_tdoa_block_skipping`

enum qerr `fira_helper_get_session_dl_tdoa_block_skipping`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*number)

*[Not supported in QM33 SDK]* Get dl-tdoa number of blocks that shall be skipped between 2 active ranging blocks.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **number** (uint8\_t\*) – Number of blocks to be skipped by the dt-tag.

### 2.3.75.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.76 `fira_helper_get_session_report_psdus`

enum qerr `fira_helper_get_session_report_psdus`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*active)

[Not supported in QM33 SDK] Get activation of psdus report.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `active` (uint8\_t\*) – True if psdus are reported, false otherwise

### 2.3.76.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.77 `fira_helper_bool_to_ranging_round_control`

uint8\_t `fira_helper_bool_to_ranging_round_control`(bool result\_report\_phase, bool skip\_ranging\_control\_phase)

get the ranging round control bitfield format.

#### Parameters

- `result_report_phase` (bool) – True if result report phase present.
- `skip_ranging_control_phase` (bool) – True if ranging control phase is skipped.

### 2.3.77.1 Return

ranging round control bitfield format.

### 2.3.78 `fira_helper_set_session_ranging_round_usage`

enum qerr `fira_helper_set_session_ranging_round_usage`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t ranging\_round\_usage)

Sets ranging round usage.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `ranging_round_usage` (uint8\_t) – See [enum fira\\_ranging\\_round\\_usage](#).

### 2.3.78.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.79 `fira_helper_set_session_device_role`

enum qerr **fira\_helper\_set\_session\_device\_role**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t device\_role)

Sets the device role

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **device\_role** (uint8\_t) – Role played by the device, accepted value are initiator for controller and responder for controlee.

### 2.3.79.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.80 `fira_helper_set_session_sts_config`

enum qerr **fira\_helper\_set\_session\_sts\_config**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t sts\_config)

scrambled timestamp sequence configuration.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **sts\_config** (uint8\_t) – Possible values: 0x01: Static STS (default). 0x02: Dynamic STS. *[Not supported in QM33 SDK]* 0x04: RFU (Dynamic STS - Individual Key). *[Not supported in QM33 SDK]* 0x08: Provisioned STS. 0x10: RFU (Provisioned STS - Individual Key).

### 2.3.80.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.81 `fira_helper_set_session_multi_node_mode`

enum qerr **fira\_helper\_set\_session\_multi\_node\_mode**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t multi\_node\_mode)

The multi-node mode used during a round.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **multi\_node\_mode** (uint8\_t) –

- FIRA\_MULTI\_NODE\_MODE\_UNICAST,
- FIRA\_MULTI\_NODE\_MODE\_ONE\_TO\_MANY,
- **[NOT IMPLEMENTED]** FIRA\_MULTI\_NODE\_MODE\_MANY\_TO\_MANY,

#### 2.3.81.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.82 `fira_helper_set_session_short_address`

enum qerr `fira_helper_set_session_short_address`(struct fira\_context \*ctx, uint32\_t session\_handle, uint16\_t short\_addr)

Sets short address.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **short\_addr** (uint16\_t) – Short\_addr.

#### 2.3.82.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.83 `fira_helper_set_session_destination_short_addresses`

enum qerr `fira_helper_set_session_destination_short_addresses`(struct fira\_context \*ctx, uint32\_t session\_handle, uint32\_t n\_dest\_short\_addr, uint16\_t \*dest\_short\_addr)

Sets destination short addresses.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **n\_dest\_short\_addr** (uint32\_t) – Number of destination short addresses.
- **dest\_short\_addr** (uint16\_t\*) – Array of destination short addresses.

#### 2.3.83.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.84 `fira_helper_set_session_time0_ns`

```
enum qerr fira_helper_set_session_time0_ns(struct fira_context *ctx, uint32_t session_handle, uint64_t
time0_ns)
```

Sets an absolute value of the initiation time [ns].

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **time0\_ns** (uint64\_t) – time0\_ns.

#### 2.3.84.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.85 `fira_helper_set_session_slot_duration_rstu`

```
enum qerr fira_helper_set_session_slot_duration_rstu(struct fira_context *ctx, uint32_t session_handle,
uint32_t slot_duration_rstu)
```

Sets slot duration rstu.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **slot\_duration\_rstu** (uint32\_t) – Slot\_duration\_rstu. - Duration of a slot in RSTU (1200RSTU=1ms)

#### 2.3.85.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.86 `fira_helper_set_session_round_duration_slots`

```
enum qerr fira_helper_set_session_round_duration_slots(struct fira_context *ctx, uint32_t session_handle,
uint32_t round_duration_slots)
```

Sets round duration slots.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **round\_duration\_slots** (uint32\_t) – Number of slots per ranging round.

### 2.3.86.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.87 `fira_helper_set_session_block_duration_ms`

```
enum qerr fira_helper_set_session_block_duration_ms(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t block_duration_ms)
```

Sets block duration.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **block\_duration\_ms** (uint32\_t) – Block size in unit of 1200 RSTU (same as ms).

### 2.3.87.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.88 `fira_helper_set_session_time_base`

```
enum qerr fira_helper_set_session_time_base(struct fira_context *ctx, uint32_t session_handle, const uint8_t
                                                    *time_base_param)
```

*[Not supported in QM33 SDK]* Set session time base configuration.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **time\_base\_param** (const uint8\_t\*) – Session time base parameter array. Expected array size equals to FIRA\_TIME\_BASE\_SIZE.

### 2.3.88.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.89 `fira_helper_set_session_block_stride_length`

```
enum qerr fira_helper_set_session_block_stride_length(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t block_stride_length)
```

Sets block stride length.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **block\_stride\_length** (uint32\_t) – Number of blocks to stride.



### 2.3.89.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.90 `fira_helper_set_session_round_hopping`

enum qerr `fira_helper_set_session_round_hopping`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t round\_hopping)

Enables or disable round hopping

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `round_hopping` (uint8\_t) – False - disabled, true - enabled

### 2.3.90.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.91 `fira_helper_set_session_priority`

enum qerr `fira_helper_set_session_priority`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t priority)  
sets the priority.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `priority` (uint8\_t) – Priority of the session.

### 2.3.91.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.92 `fira_helper_set_session_mac_address_mode`

enum qerr `fira_helper_set_session_mac_address_mode`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t mac\_address\_mode)

sets the MAC addressing mode.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `mac_address_mode` (uint8\_t) – MAC addressing mode.

### 2.3.92.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.93 `fira_helper_set_session_ranging_round_control`

enum qerr `fira_helper_set_session_ranging_round_control`(struct `fira_context` \*ctx, uint32\_t session\_handle, uint8\_t ranging\_round\_control)

Set the ranging round control bitfield.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `ranging_round_control` (uint8\_t) – Bitfield: - b0: ranging result report phase is disabled (0) or enabled (1) - b1: Control Message is sent in band (1) or not (0, not supported) - b2: Control Message is sent separately (0) or piggybacked to RIM (1)

### 2.3.93.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.94 `fira_helper_set_session_schedule_mode`

enum qerr `fira_helper_set_session_schedule_mode`(struct `fira_context` \*ctx, uint32\_t session\_handle, uint8\_t schedule\_mode)

Sets schedule mode parameter.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `schedule_mode` (uint8\_t) –
  - 0x00 - Contention-based ranging. *[Not supported in QM33 SDK]*
  - 0x01 - Time-scheduled ranging.
  - 0x02 - Hybrid-based ranging. *[Not supported in QM33 SDK]*

### 2.3.94.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.95 `fira_helper_set_session_max_number_of_measurements`

```
enum qerr fira_helper_set_session_max_number_of_measurements(struct fira_context *ctx, uint32_t
                                                                session_handle, uint32_t
                                                                max_number_of_measurements)
```

Sets the max number of measurements.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **max\_number\_of\_measurements** (uint32\_t) – Max\_number\_of\_measurements.

#### 2.3.95.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.96 `fira_helper_set_session_max_rr_retry`

```
enum qerr fira_helper_set_session_max_rr_retry(struct fira_context *ctx, uint32_t session_handle, uint32_t
                                                                max_rr_retry)
```

Sets the max rr retry.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **max\_rr\_retry** (uint32\_t) – Max\_rr\_retry. Number of failed ranging round attempts before stopping the session. The value zero disables the feature.

#### 2.3.96.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.97 `fira_helper_set_session_channel_number`

```
enum qerr fira_helper_set_session_channel_number(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                                channel_number)
```

Sets the channel number.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **channel\_number** (uint8\_t) – Channel\_number.

### 2.3.97.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.98 `fira_helper_set_session_preamble_code_index`

enum qerr **fira\_helper\_set\_session\_preamble\_code\_index**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t preamble\_code\_index)

Sets preamble code index.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **preamble\_code\_index** (uint8\_t) – Preamble code index.

Possible values:

- 9-24: BPRF
- 25-32: HPRF

### 2.3.98.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.99 `fira_helper_set_session_rframe_config`

enum qerr **fira\_helper\_set\_session\_rframe\_config**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t rframe\_config)

Sets rframe\_config.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **rframe\_config** (uint8\_t) – Ranging frame config.

### 2.3.99.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.100 `fira_helper_set_session_preamble_duration`

enum qerr **fira\_helper\_set\_session\_preamble\_duration**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t preamble\_duration)

Sets preamble duration.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

- **preamble\_duration** (uint8\_t) – 0x00: 32 symbols *[Not supported in QM33 SDK]* or 0x01: 64 symbols (default)

### 2.3.100.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.101 **fira\_helper\_set\_session\_sfd\_id**

enum qerr **fira\_helper\_set\_session\_sfd\_id**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t sfd\_id)  
Sets sfd\_id.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **sfd\_id** (uint8\_t) – 0 or 2 in BPRF, 1-4 in HPRF *[Not supported in QM33 SDK]*

### 2.3.101.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.102 **fira\_helper\_set\_session\_psdu\_data\_rate**

enum qerr **fira\_helper\_set\_session\_psdu\_data\_rate**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t psdu\_data\_rate)  
Sets psdu data rate.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **psdu\_data\_rate** (uint8\_t) – Possible values: - 0: 6.81Mbps (default) - 1: 7.80 Mbps *[Not supported in QM33 SDK]* - 2: 27.2 Mbps *[Not supported in QM33 SDK]* - 3: 31.2 Mbps *[Not supported in QM33 SDK]*

### 2.3.102.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.103 **fira\_helper\_set\_session\_sub\_session\_id**

enum qerr **fira\_helper\_set\_session\_sub\_session\_id**(struct fira\_context \*ctx, uint32\_t session\_handle, uint32\_t sub\_session\_id)  
Sets controlee' sub-session id.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

- **sub\_session\_id** (uint32\_t) – Controlee’ sub-session id used during Dynamic or Provisioned STS for Responder Specific Sub-session Key.

### 2.3.103.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.104 **fira\_helper\_set\_session\_vendor\_id**

enum qerr **fira\_helper\_set\_session\_vendor\_id**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t vendor\_id)

Sets Vendor ID.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **vendor\_id** (uint8\_t) – Vendor ID used for vUpper64.

### 2.3.104.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.105 **fira\_helper\_set\_session\_static\_sts\_iv**

enum qerr **fira\_helper\_set\_session\_static\_sts\_iv**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t static\_sts\_iv)

Sets Static STS IV.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **static\_sts\_iv** (uint8\_t) – Static STS IV used for vUpper64.

### 2.3.105.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.106 **fira\_helper\_set\_session\_vupper64**

enum qerr **fira\_helper\_set\_session\_vupper64**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t vupper64)

Sets vupper64.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **vupper64** (uint8\_t) – vUpper64 used during Static STS ranging.

### 2.3.106.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.107 `fira_helper_set_session_key_rotation`

```
enum qerr fira_helper_set_session_key_rotation(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                             key_rotation)
```

Enable/disable key rotation.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **key\_rotation** (uint8\_t) – 0 to disable key rotation, 1 to enable it. Enable/disable key rotation during Dynamic *[Not supported in QM33 SDK]* or Provisioned STS ranging. If enable the period will be set with `fira_helper_set_session_key_rotation_rate`.

### 2.3.107.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.108 `fira_helper_set_session_key_rotation_rate`

```
enum qerr fira_helper_set_session_key_rotation_rate(struct fira_context *ctx, uint32_t session_handle,
                                                    uint8_t key_rotation_rate)
```

Sets key rotation rate.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **key\_rotation\_rate** (uint8\_t) – Defines  $n$ , with  $2^n$  being the rotation rate of some keys used during Dynamic *[Not supported in QM33 SDK]* or Provisioned STS Ranging,  $n$  shall be in the range of  $0 \leq n \leq 15$ .

### 2.3.108.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.109 `fira_helper_set_session_mac_payload_encryption`

```
enum qerr fira_helper_set_session_mac_payload_encryption(struct fira_context *ctx, uint32_t
                                                         session_handle, uint8_t
                                                         mac_payload_encryption)
```

Enable or disable encryption of payload data.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

- `mac_payload_encryption` (uint8\_t) – Status of mac payload encryption.

### 2.3.109.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.110 `fira_helper_set_session_report_rssi`

enum qerr `fira_helper_set_session_report_rssi`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t report\_rssi)

Sets the report rssi.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `report_rssi` (uint8\_t) – Report\_rssi false - no report, true report

### 2.3.110.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.111 `fira_helper_bool_to_result_report_config`

uint8\_t `fira_helper_bool_to_result_report_config`(bool report\_tof, bool report\_aoa\_azimuth, bool report\_aoa\_elevation, bool report\_aoa\_fom)

get the result report config bitfield format.

#### Parameters

- `report_tof` (bool) – True if time of flight must be reported.
- `report_aoa_azimuth` (bool) – True if azimuth's angle of arrival must be reported.
- `report_aoa_elevation` (bool) – True if elevation's angle of arrival must be reported.
- `report_aoa_fom` (bool) – True if aoa figure of merit must be reported.

### 2.3.111.1 Return

result report config bitfield format.

### 2.3.112 `fira_helper_set_session_result_report_config`

enum qerr `fira_helper_set_session_result_report_config`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t result\_report\_config)

Enable/disable time of flight.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.



- **result\_report\_config** (uint8\_t) – See enum `fira_result_report_config`.

### 2.3.112.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.113 `fira_helper_set_session_link_layer_mode`

enum qerr **fira\_helper\_set\_session\_link\_layer\_mode**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t link\_layer\_mode)

Sets link layer mode.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **link\_layer\_mode** (uint8\_t) – Link layer configuration: 0x00: Bypass mode. 0x01: Connection less. *[Not supported in QM33 SDK]*

### 2.3.113.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.114 `fira_helper_set_session_data_repetition_count`

enum qerr **fira\_helper\_set\_session\_data\_repetition\_count**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t data\_repetition\_count)

*[Not supported in QM33 SDK]* Sets data repetition count.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_repetition\_count** (uint8\_t) – Number of times the current MDSDU shall be sent. 0x00: No repetition. 0x01 - 0xFE: Number of repetitions. 0xFF: Infinite number of times.

### 2.3.114.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.115 `fira_helper_set_session_data_transfer_status_ntf_config`

```
enum qerr fira_helper_set_session_data_transfer_status_ntf_config(struct fira_context *ctx, uint32_t
                                                                    session_handle, uint8_t config)
```

*[Not supported in QM33 SDK]* Sets config value for SESSION\_DATA\_TRANSFER\_STATUS\_NTF.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **config** (uint8\_t) – Value to set. 0x00: Disable 0x01: Enable

#### 2.3.115.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.116 `fira_helper_set_session_mac_fcs_type`

```
enum qerr fira_helper_set_session_mac_fcs_type(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                                    mac_fcs_type)
```

Sets the CRC type.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **mac\_fcs\_type** (uint8\_t) – CRC type: 0x00: CRC 16. 0x01: CRC 32.

#### 2.3.116.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.117 `fira_helper_set_session_number_of_sts_segments`

```
enum qerr fira_helper_set_session_number_of_sts_segments(struct fira_context *ctx, uint32_t
                                                                    session_handle, uint8_t
                                                                    number_of_sts_segments)
```

Sets the number of STS segments.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **number\_of\_sts\_segments** (uint8\_t) – Number of STS segments: 0x00: No STS Segments. 0x01: 1 STS Segment. 0x02: 2 STS Segments (HPRF only). *[Not supported in QM33 SDK]* 0x03: 3 STS Segments (HPRF only). *[Not supported in QM33 SDK]* 0x04: 4 STS Segments (HPRF only). *[Not supported in QM33 SDK]*

### 2.3.117.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.118 `fira_helper_set_session_phr_data_rate`

enum qerr `fira_helper_set_session_phr_data_rate`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t phr\_data\_rate)

Sets the PHR data rate.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `phr_data_rate` (uint8\_t) – PHR data rate: 0x00: 850 kbps. 0x01: 6.81 Mbps.

### 2.3.118.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.119 `fira_helper_set_session_prf_mode`

enum qerr `fira_helper_set_session_prf_mode`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t prf\_mode)

Sets prf mode.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `prf_mode` (uint8\_t) – Prf\_mode pulse repetition frequency. 0x00: 62.4 MHz PRF. BPRF mode (default) 0x01: 124.8 MHz PRF. HPRF mode. *[Not supported in QM33 SDK]* 0x02: 249.6 MHz PRF. HPRF mode with data rate 27.2 and 31.2 Mbps. *[Not supported in QM33 SDK]*

### 2.3.119.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.120 `fira_helper_set_session_cap_size_min`

enum qerr `fira_helper_set_session_cap_size_min`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t cap\_size\_min)

*[Not supported in QM33 SDK]* Sets the cap size min.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `cap_size_min` (uint8\_t) – Cap\_size\_min - default 5.

### 2.3.120.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.121 `fira_helper_set_session_cap_size_max`

enum qerr `fira_helper_set_session_cap_size_max`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t cap\_size\_max)

*[Not supported in QM33 SDK]* Sets the cap size max.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `cap_size_max` (uint8\_t) – Cap\_size\_max.

### 2.3.121.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.122 `fira_helper_set_session_measurement_sequence`

enum qerr `fira_helper_set_session_measurement_sequence`(struct fira\_context \*ctx, uint32\_t session\_handle, const struct [measurement\\_sequence](#) \*meas\_seq)

*[Not supported in QM33 SDK]* Sets the measurement sequence.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `meas_seq` (const struct [measurement\\_sequence](#)\*) – Sequence of measurement sequence steps, configures the Antenna Flexibility features.

### 2.3.122.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.123 `fira_helper_set_session_enable_diagnostics`

enum qerr `fira_helper_set_session_enable_diagnostics`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t enable\_diagnostics)

Enables diagnostics.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `enable_diagnostics` (uint8\_t) – Enable\_diagnostics 0 - no, 1 - yes.

### 2.3.123.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.124 `fira_helper_set_session_diags_frame_reports_fields`

```
enum qerr fira_helper_set_session_diags_frame_reports_fields(struct fira_context *ctx, uint32_t
                                                             session_handle, uint32_t
                                                             diags_frame_reports_fields)
```

Sets the diag frame fields.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **diags\_frame\_reports\_fields** (uint32\_t) – Select the fields to activate in the frame reports stored in the diags. Applicable only when enable\_diagnostics is set to true.

### 2.3.124.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.125 `fira_helper_set_session_sts_length`

```
enum qerr fira_helper_set_session_sts_length(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                              sts_length)
```

Sets sts length.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **sts\_length** (uint8\_t) – Values 0x00: 32 symbols 0x01: 64 symbols (default) 0x02: 128 symbols Values 0x03 to 0xFF: RFU

### 2.3.125.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.126 `fira_helper_set_session_min_frames_per_rr`

```
enum qerr fira_helper_set_session_min_frames_per_rr(struct fira_context *ctx, uint32_t session_handle,
                                                      uint8_t min_frames_per_rr)
```

*[Not supported in QM33 SDK]* Sets min\_frames\_per\_rr

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **min\_frames\_per\_rr** (uint8\_t) – Min\_frames\_per\_rr

### 2.3.126.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.127 `fira_helper_set_session_mtu_size`

```
enum qerr fira_helper_set_session_mtu_size(struct fira_context *ctx, uint32_t session_handle, uint16_t
                                          mtu_size)
```

*[Not supported in QM33 SDK]* Sets `mtu_size`

#### Parameters

- `ctx` (struct `fira_context*`) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `mtu_size` (uint16\_t) – `Mtu_size`, the value shall be restricted to the maximum possible MTU size of the given frame which includes MHR, Variable IE size and FCS size.

### 2.3.127.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.128 `fira_helper_set_session_inter_frame_interval_ms`

```
enum qerr fira_helper_set_session_inter_frame_interval_ms(struct fira_context *ctx, uint32_t
                                                         session_handle, uint8_t
                                                         inter_frame_interval_ms)
```

*[Not supported in QM33 SDK]* Sets `inter_frame_interval_ms`

#### Parameters

- `ctx` (struct `fira_context*`) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `inter_frame_interval_ms` (uint8\_t) – `Inter_frame_interval_ms`

### 2.3.128.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.129 `fira_helper_set_session_owr_aoa_measurement_ntf_period`

```
enum qerr fira_helper_set_session_owr_aoa_measurement_ntf_period(struct fira_context *ctx, uint32_t
                                                                session_handle, uint8_t
                                                                owr_aoa_measurement_ntf_period)
```

*[Not supported in QM33 SDK]* Sets OWR for AoA measurement notification period.

#### Parameters

- `ctx` (struct `fira_context*`) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.

- `owr_aoa_measurement_ntf_period` (uint8\_t) – 0 - send on every frame, 1 - send once after MIN\_FRAMES\_PER\_RR number of AoA measurements are aggregated.

### 2.3.129.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.130 `fira_helper_set_session_session_info_ntf_config`

```
enum qerr fira_helper_set_session_session_info_ntf_config(struct fira_context *ctx, uint32_t
                                                         session_handle, uint8_t
                                                         session_info_ntf_config)
```

*[Not supported in QM33 SDK]* Sets ntf config.

#### Parameters

- `ctx` (struct `fira_context*`) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `session_info_ntf_config` (uint8\_t) – Values; 0x00 = Disable session info notification (ntf) 0x01 = Enable session info notification (default) 0x02 = Enable session info ntf while inside proximity range 0x03 = Enable session info ntf while inside AoA upper and lower bounds 0x04 = Enable session info ntf while inside AoA upper and lower bounds as well as inside proximity range 0x05 = Enable session info ntf only when entering or leaving proximity range 0x06 = Enable session info ntf only when entering or leaving AoA upper and lower bounds 0x07 = Enable session info ntf only when entering or leaving AoA upper and lower bounds as well as entering or leaving proximity range.

### 2.3.130.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.131 `fira_helper_set_session_near_proximity_config_cm`

```
enum qerr fira_helper_set_session_near_proximity_config_cm(struct fira_context *ctx, uint32_t
                                                           session_handle, uint32_t
                                                           near_proximity_config_cm)
```

*[Not supported in QM33 SDK]* set proximity near cm.

#### Parameters

- `ctx` (struct `fira_context*`) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `near_proximity_config_cm` (uint32\_t) – Range\_data\_ntf\_proximity\_near\_cm. prerequisites: Applicable when `session_info_ntf_config` is set to 0x02, 0x04, 0x05 or 0x07. Should be less than or equal to `far_proximity_config` value.

### 2.3.131.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.132 `fira_helper_set_session_far_proximity_config_cm`

```
enum qerr fira_helper_set_session_far_proximity_config_cm(struct fira_context *ctx, uint32_t
                                                         session_handle, uint32_t
                                                         far_proximity_config_cm)
```

*[Not supported in QM33 SDK]* Sets `far_proximity_config_cm`. prerequisites, Applicable when `session_info_ntf_config` is set to 0x02, 0x04, 0x05 or 0x07. Should be greater than or equal to `near_proximity_config` value.

#### Parameters

- **ctx** (struct `fira_context*`) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **far\_proximity\_config\_cm** (uint32\_t) – `Range_data_ntf_proximity_far_cm`.

### 2.3.132.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.133 `fira_helper_get_session_device_type`

```
enum qerr fira_helper_get_session_device_type(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                         *device_type)
```

Gets the device type.

#### Parameters

- **ctx** (struct `fira_context*`) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **device\_type** (uint8\_t\*) – Variable to store the value.

### 2.3.133.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.134 `fira_helper_get_session_ranging_round_usage`

```
enum qerr fira_helper_get_session_ranging_round_usage(struct fira_context *ctx, uint32_t session_handle,
                                                         uint8_t *ranging_round_usage)
```

Gets the ranging round usage.

#### Parameters

- **ctx** (struct `fira_context*`) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **ranging\_round\_usage** (uint8\_t\*) – Variable to store the value.



### 2.3.134.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.135 `fira_helper_get_session_device_role`

enum qerr `fira_helper_get_session_device_role`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*device\_role)

Gets device role.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `device_role` (uint8\_t\*) – Device\_role.

### 2.3.135.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.136 `fira_helper_get_session_sts_config`

enum qerr `fira_helper_get_session_sts_config`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*sts\_config)

Gets the sts config.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `sts_config` (uint8\_t\*) – Variable to store the value.

### 2.3.136.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.137 `fira_helper_get_session_multi_node_mode`

enum qerr `fira_helper_get_session_multi_node_mode`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*multi\_node\_mode)

Gets the multi node mode.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `multi_node_mode` (uint8\_t\*) – Variable to store the value.

### 2.3.137.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.138 `fira_helper_get_session_short_address`

```
enum qerr fira_helper_get_session_short_address(struct fira_context *ctx, uint32_t session_handle, uint16_t
                                                *short_addr)
```

Gets the short address.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **short\_addr** (uint16\_t\*) – Variable to store the value.

### 2.3.138.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.139 `fira_helper_get_session_destination_short_addresses`

```
enum qerr fira_helper_get_session_destination_short_addresses(struct fira_context *ctx, uint32_t
                                                                session_handle, uint32_t
                                                                *n_dest_short_addr, uint16_t
                                                                *dest_short_addr)
```

Gets the destination short addresses.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **n\_dest\_short\_addr** (uint32\_t\*) – Number of destination short addresses.
- **dest\_short\_addr** (uint16\_t\*) – Array of destination short addresses.

### 2.3.139.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.140 `fira_helper_get_session_time0_ns`

```
enum qerr fira_helper_get_session_time0_ns(struct fira_context *ctx, uint32_t session_handle, uint64_t
                                                *time0_ns)
```

Gets an absolute value of the initiation time [ns].

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **time0\_ns** (uint64\_t\*) – Variable to store the value.

### 2.3.140.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.141 `fira_helper_get_session_slot_duration_rstu`

```
enum qerr fira_helper_get_session_slot_duration_rstu(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t *slot_duration_rstu)
```

Gets the slot duration rstu.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **slot\_duration\_rstu** (uint32\_t\*) – Variable to store the value.

### 2.3.141.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.142 `fira_helper_get_session_round_duration_slots`

```
enum qerr fira_helper_get_session_round_duration_slots(struct fira_context *ctx, uint32_t session_handle,
                                                         uint32_t *round_duration_slots)
```

Gets the number of round duration slots

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **round\_duration\_slots** (uint32\_t\*) – Variable to store the value.

### 2.3.142.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.143 `fira_helper_get_session_block_duration_ms`

```
enum qerr fira_helper_get_session_block_duration_ms(struct fira_context *ctx, uint32_t session_handle,
                                                       uint32_t *block_duration_ms)
```

Gets the block duration.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **block\_duration\_ms** (uint32\_t\*) – Variable to store the value. (Block size in unit of 1200 RSTU (same as ms))

### 2.3.143.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.144 `fira_helper_get_session_block_stride_length`

enum qerr **fira\_helper\_get\_session\_block\_stride\_length**(struct fira\_context \*ctx, uint32\_t session\_handle, uint32\_t \*block\_stride\_length)

Gets the block stride length.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **block\_stride\_length** (uint32\_t\*) – Variable to store the number of blocks to stride.

### 2.3.144.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.145 `fira_helper_get_session_round_hopping`

enum qerr **fira\_helper\_get\_session\_round\_hopping**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*round\_hopping)

Gets the round hopping 0 - disabled, 1 enabled.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **round\_hopping** (uint8\_t\*) – Variable to store the value.

### 2.3.145.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.146 `fira_helper_get_session_priority`

enum qerr **fira\_helper\_get\_session\_priority**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*priority)

Gets the priority.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **priority** (uint8\_t\*) – Variable to store the value.

### 2.3.146.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.147 `fira_helper_get_session_mac_address_mode`

enum qerr **fira\_helper\_get\_session\_mac\_address\_mode**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*mac\_address\_mode)

Gets the MAC addressing mode.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **mac\_address\_mode** (uint8\_t\*) – Variable to store the value.

### 2.3.147.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.148 `fira_helper_get_session_ranging_round_control`

enum qerr **fira\_helper\_get\_session\_ranging\_round\_control**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*ranging\_round\_control)

Gets the ranging round control.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **ranging\_round\_control** (uint8\_t\*) – Storage variable, where ranging round control is: - b0: ranging result report phase is disabled (0) or enabled (1) - b1: Control Message is sent in band (1) or not (0) - b2: Control Message is sent separately (0) or piggybacked to RIM (1) - b3-b6: RFUs, must be set to 0. - b7: MRM is sent from the initiator (0) or from the responder (1)

### 2.3.148.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.149 `fira_helper_get_session_schedule_mode`

enum qerr **fira\_helper\_get\_session\_schedule\_mode**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*schedule\_mode)

Gets schedule mode parameter.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

- **schedule\_mode** (uint8\_t\*) – Variable to store the value. - 0x00 - Contention-based ranging. *[Not supported in QM33 SDK]* - 0x01 - Time-scheduled ranging. - 0x02 - Hybrid-based ranging. *[Not supported in QM33 SDK]*

### 2.3.149.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.150 **fira\_helper\_get\_session\_max\_number\_of\_measurements**

```
enum qerr fira_helper_get_session_max_number_of_measurements(struct fira_context *ctx, uint32_t
                                                                session_handle, uint32_t
                                                                *max_number_of_measurements)
```

Gets the number of measurements.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **max\_number\_of\_measurements** (uint32\_t\*) – Variable to store the value.

### 2.3.150.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.151 **fira\_helper\_get\_session\_max\_rr\_retry**

```
enum qerr fira_helper_get_session_max_rr_retry(struct fira_context *ctx, uint32_t session_handle, uint32_t
                                                                *max_rr_retry)
```

Gets the maximum ranging rounds retry.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **max\_rr\_retry** (uint32\_t\*) – Max\_rr\_retry. Variable to store the value. Number of failed ranging round attempts before stopping the session. The value zero disables the feature.

### 2.3.151.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.152 `fira_helper_get_session_channel_number`

enum qerr `fira_helper_get_session_channel_number`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*channel\_number)

Gets the channel used in this session.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `channel_number` (uint8\_t\*) – Variable to store the value.

#### 2.3.152.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.153 `fira_helper_get_session_preamble_code_index`

enum qerr `fira_helper_get_session_preamble_code_index`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*preamble\_code\_index)

Gets preamble code index.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `preamble_code_index` (uint8\_t\*) – Variable to store the value, possible values: - 9-24: BPRF - 25-32: HPRF

#### 2.3.153.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.154 `fira_helper_get_session_rframe_config`

enum qerr `fira_helper_get_session_rframe_config`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*rframe\_config)

Gets the rframe\_config.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `rframe_config` (uint8\_t\*) – Variable to store the value.

### 2.3.154.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.155 `fira_helper_get_session_preamble_duration`

```
enum qerr fira_helper_get_session_preamble_duration(struct fira_context *ctx, uint32_t session_handle,
                                                    uint8_t *preamble_duration)
```

Gets the preamble duration.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **preamble\_duration** (uint8\_t\*) – Variable to store the value. 0x00: 32 symbols *[Not supported in QM33 SDK]* or 0x01: 64 symbols (default)a.

### 2.3.155.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.156 `fira_helper_get_session_sfd_id`

```
enum qerr fira_helper_get_session_sfd_id(struct fira_context *ctx, uint32_t session_handle, uint8_t *sfd_id)
```

Gets sfd\_id.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **sfd\_id** (uint8\_t\*) – Sfd\_id. Variable to store the value. possible values 0 or 2 in BPRF, 1-4 in HPRF *[Not supported in QM33 SDK]*

### 2.3.156.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.157 `fira_helper_get_session_psdu_data_rate`

```
enum qerr fira_helper_get_session_psdu_data_rate(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                    *psdu_data_rate)
```

Gets the psdu data rate.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **psdu\_data\_rate** (uint8\_t\*) – Psdu\_data\_rate.



### 2.3.157.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.158 `fira_helper_get_session_sub_session_id`

```
enum qerr fira_helper_get_session_sub_session_id(struct fira_context *ctx, uint32_t session_handle, uint32_t
                                                    *sub_session_id)
```

Gets controlee' sub-session id.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **sub\_session\_id** (uint32\_t\*) – Controlee' sub-session id used during Dynamic or Provisioned STS for Responder Specific Sub-session Key.

### 2.3.158.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.159 `fira_helper_get_session_vendor_id`

```
enum qerr fira_helper_get_session_vendor_id(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                              vendor_id)
```

Gets the Vendor ID.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **vendor\_id** (uint8\_t) – Vendor ID.

### 2.3.159.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.160 `fira_helper_get_session_static_sts_iv`

```
enum qerr fira_helper_get_session_static_sts_iv(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                  static_sts_iv)
```

Gets the Static STS IV.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **static\_sts\_iv** (uint8\_t) – Static STS IV.

### 2.3.160.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.161 `fira_helper_get_session_vupper64`

```
enum qerr fira_helper_get_session_vupper64(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                             vupper64)
```

Gets the vupper.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **vupper64** (uint8\_t) – vupper64.

### 2.3.161.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.162 `fira_helper_get_session_key_rotation`

```
enum qerr fira_helper_get_session_key_rotation(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                *key_rotation)
```

Gets the key rotation.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **key\_rotation** (uint8\_t\*) – False - no rotation, true rotation

### 2.3.162.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.163 `fira_helper_get_session_key_rotation_rate`

```
enum qerr fira_helper_get_session_key_rotation_rate(struct fira_context *ctx, uint32_t session_handle,
                                                      uint8_t *key_rotation_rate)
```

Gets the key rotation rate.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **key\_rotation\_rate** (uint8\_t\*) – Value to store the variable which. defines n, with  $2^n$  being the rotation rate of some keys used during Dynamic *[Not supported in QM33 SDK]* or Provisioned STS Ranging, n shall be in the range of  $0 \leq n \leq 15$ .

### 2.3.163.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.164 `fira_helper_get_session_mac_payload_encryption`

```
enum qerr fira_helper_get_session_mac_payload_encryption(struct fira_context *ctx, uint32_t
                                                         session_handle, uint8_t
                                                         *mac_payload_encryption)
```

Gets the status of encryption of payload data.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **mac\_payload\_encryption** (uint8\_t\*) – Status of mac payload encryption.

### 2.3.164.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.165 `fira_helper_get_session_report_rssi`

```
enum qerr fira_helper_get_session_report_rssi(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                *report_rssi)
```

Gets rssi report.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **report\_rssi** (uint8\_t\*) – Variable to store: false no report, true report.

### 2.3.165.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.166 `fira_helper_get_session_result_report_config`

```
enum qerr fira_helper_get_session_result_report_config(struct fira_context *ctx, uint32_t session_handle,
                                                         uint8_t *result_report_config)
```

Gets the report of

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **result\_report\_config** (uint8\_t\*) – Variable to store: false no report, true report.

### 2.3.166.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.167 `fira_helper_get_session_data_vendor_oui`

```
enum qerr fira_helper_get_session_data_vendor_oui(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t *data_vendor_oui)
```

gets the data vendor own unique id.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_vendor\_oui** (uint32\_t\*) – Variable to store: false no report, true report.

### 2.3.167.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.168 `fira_helper_get_session_link_layer_mode`

```
enum qerr fira_helper_get_session_link_layer_mode(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                    *link_layer_mode)
```

Gets link layer mode.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **link\_layer\_mode** (uint8\_t\*) – Link layer mode.

### 2.3.168.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.169 `fira_helper_get_session_data_repetition_count`

```
enum qerr fira_helper_get_session_data_repetition_count(struct fira_context *ctx, uint32_t session_handle,
                                                         uint8_t *data_repetition_count)
```

*[Not supported in QM33 SDK]* Gets data repetition count.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_repetition\_count** (uint8\_t\*) – Gets number of times each MDSDU shall be repeated.

### 2.3.169.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.170 `fira_helper_get_session_data_transfer_status_ntf_config`

enum qerr `fira_helper_get_session_data_transfer_status_ntf_config`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*config)

*[Not supported in QM33 SDK]* Gets config value for SESSION\_DATA\_TRANSFER\_STATUS\_NTF.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `config` (uint8\_t\*) – Gets config value for SESSION\_DATA\_TRANSFER\_STATUS\_NTF

### 2.3.170.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.171 `fira_helper_get_session_time_base`

enum qerr `fira_helper_get_session_time_base`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*time\_base\_param)

*[Not supported in QM33 SDK]* Get session time base configuration.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `time_base_param` (uint8\_t\*) – Gets session time base parameter array. Array size equals to FIRA\_TIME\_BASE\_SIZE.

### 2.3.171.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.172 `fira_helper_get_session_mac_fcs_type`

enum qerr `fira_helper_get_session_mac_fcs_type`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*mac\_fcs\_type)

Gets CRC type.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `mac_fcs_type` (uint8\_t\*) – CRC type: 0x00: CRC 16. 0x01: CRC 32.

### 2.3.172.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.173 `fira_helper_get_session_number_of_sts_segments`

```
enum qerr fira_helper_get_session_number_of_sts_segments(struct fira_context *ctx, uint32_t
                                                         session_handle, uint8_t
                                                         *number_of_sts_segments)
```

Gets the number of STS segments.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **number\_of\_sts\_segments** (uint8\_t\*) – Number of STS segments: 0x00: No STS Segments. 0x01: 1 STS Segment. 0x02: 2 STS Segments (HPRF only). *[Not supported in QM33 SDK]* 0x03: 3 STS Segments (HPRF only). *[Not supported in QM33 SDK]* 0x04: 4 STS Segments (HPRF only). *[Not supported in QM33 SDK]*

### 2.3.173.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.174 `fira_helper_get_session_phr_data_rate`

```
enum qerr fira_helper_get_session_phr_data_rate(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                         *phr_data_rate)
```

Gets the PHR data rate.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **phr\_data\_rate** (uint8\_t\*) – PHR data rate: 0x00: 850 kbps. 0x01: 6.81 Mbps.

### 2.3.174.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.175 `fira_helper_get_session_prf_mode`

```
enum qerr fira_helper_get_session_prf_mode(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                         *prf_mode)
```

gets the prf mode.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

- **prf\_mode** (uint8\_t\*) – Prf\_mode. pulse repetition frequency Variable to store the value.

### 2.3.175.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.176 **fira\_helper\_get\_session\_cap\_size\_min**

enum qerr **fira\_helper\_get\_session\_cap\_size\_min**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*cap\_size\_min)

*[Not supported in QM33 SDK]* Gets cap size min.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **cap\_size\_min** (uint8\_t\*) – Variable to store the value

### 2.3.176.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.177 **fira\_helper\_get\_session\_cap\_size\_max**

enum qerr **fira\_helper\_get\_session\_cap\_size\_max**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*cap\_size\_max)

*[Not supported in QM33 SDK]* Get cap size max.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **cap\_size\_max** (uint8\_t\*) – Variable to store the value.

### 2.3.177.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.178 **fira\_helper\_get\_session\_enable\_diagnostics**

enum qerr **fira\_helper\_get\_session\_enable\_diagnostics**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*enable\_diagnostics)

Enables diagnostics.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **enable\_diagnostics** (uint8\_t\*) – Enable\_diagnostics 0 - no, 1 - yes.

### 2.3.178.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.179 `fira_helper_get_session_diags_frame_reports_fields`

```
enum qerr fira_helper_get_session_diags_frame_reports_fields(struct fira_context *ctx, uint32_t
                                                                session_handle, uint32_t
                                                                *diags_frame_reports_fields)
```

Gets the diag frame fields.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `diags_frame_reports_fields` (uint32\_t\*) – Select the fields to activate in the frame reports stored in the diags. Applicable only when `enable_diagnostics` is set to true.

### 2.3.179.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.180 `fira_helper_get_session_measurement_sequence`

```
enum qerr fira_helper_get_session_measurement_sequence(struct fira_context *ctx, uint32_t session_handle,
                                                         struct measurement\_sequence *meas_seq)
```

*[Not supported in QM33 SDK]* Gets the measurement sequence.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `meas_seq` (struct [measurement\\_sequence](#)\*) – Variable to store the measurement sequence

### 2.3.180.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.181 `fira_helper_get_session_sts_length`

```
enum qerr fira_helper_get_session_sts_length(struct fira_context *ctx, uint32_t session_handle, uint8_t
                                                                *sts_length)
```

gets sts length.

#### Parameters

- `ctx` (struct `fira_context`\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `sts_length` (uint8\_t\*) – Variable to store the value. 0x00: 32 symbols 0x01: 64 symbols (default) 0x02: 128 symbols Values 0x03 to 0xFF: RFU



### 2.3.181.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.182 `fira_helper_get_session_min_frames_per_rr`

```
enum qerr fira_helper_get_session_min_frames_per_rr(struct fira_context *ctx, uint32_t session_handle,
                                                    uint8_t *min_frames_per_rr)
```

*[Not supported in QM33 SDK]* Gets min\_frames\_per\_rr

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **min\_frames\_per\_rr** (uint8\_t\*) – Min\_frames\_per\_rr

### 2.3.182.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.183 `fira_helper_get_session_mtu_size`

```
enum qerr fira_helper_get_session_mtu_size(struct fira_context *ctx, uint32_t session_handle, uint16_t
                                                    *mtu_size)
```

*[Not supported in QM33 SDK]* Gets mtu\_size

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **mtu\_size** (uint16\_t\*) – Mtu\_size, the value shall be restricted to the maximum possible MTU size of the given frame which includes MHR, Variable IE size and FCS size.

### 2.3.183.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.184 `fira_helper_get_session_inter_frame_interval_ms`

```
enum qerr fira_helper_get_session_inter_frame_interval_ms(struct fira_context *ctx, uint32_t
                                                            session_handle, uint8_t
                                                            *inter_frame_interval_ms)
```

*[Not supported in QM33 SDK]* Gets inter\_frame\_interval\_ms

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **inter\_frame\_interval\_ms** (uint8\_t\*) – Inter\_frame\_interval\_ms

### 2.3.184.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.185 `fira_helper_get_session_owr_aoa_measurement_ntf_period`

```
enum qerr fira_helper_get_session_owr_aoa_measurement_ntf_period(struct fira_context *ctx, uint32_t
                                                                session_handle, uint8_t
                                                                *owr_aoa_measurement_ntf_period)
```

*[Not supported in QM33 SDK]* Gets OWR for AoA measurement notification period.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **owr\_aoa\_measurement\_ntf\_period** (uint8\_t\*) – 0 - send on every frame, 1 - send once after MIN\_FRAMES\_PER\_RR number of AoA measurements are aggregated.

### 2.3.185.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.186 `fira_helper_get_session_session_info_ntf_config`

```
enum qerr fira_helper_get_session_session_info_ntf_config(struct fira_context *ctx, uint32_t
                                                         session_handle, uint8_t
                                                         *session_info_ntf_config)
```

*[Not supported in QM33 SDK]* Gets range notification.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **session\_info\_ntf\_config** (uint8\_t\*) – Variable to store the value: 0x00 = Disable session info notification (ntf). 0x01 = Enable session info notification (default). 0x02 = Enable session info ntf while inside proximity range. 0x03 = Enable session info ntf while inside AoA upper and lower bounds. 0x04 = Enable session info ntf while inside AoA upper and lower bounds as well as inside proximity range. 0x05 = Enable session info ntf only when entering or leaving proximity range. 0x06 = Enable session info ntf only when entering or leaving AoA upper and lower bounds. 0x07 = Enable session info ntf only when entering or leaving AoA upper and lower bounds as well as entering or leaving proximity range.

### 2.3.186.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.187 `fira_helper_get_session_near_proximity_config_cm`

```
enum qerr fira_helper_get_session_near_proximity_config_cm(struct fira_context *ctx, uint32_t
                                                         session_handle, uint32_t
                                                         *near_proximity_config_cm)
```

*[Not supported in QM33 SDK]* Gets ntf proximity near.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **near\_proximity\_config\_cm** (uint32\_t\*) – Variable to store the value.

### 2.3.187.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.188 `fira_helper_get_session_far_proximity_config_cm`

```
enum qerr fira_helper_get_session_far_proximity_config_cm(struct fira_context *ctx, uint32_t
                                                         session_handle, uint32_t
                                                         *far_proximity_config_cm)
```

*[Not supported in QM33 SDK]* Gets ntf proximity far.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **far\_proximity\_config\_cm** (uint32\_t\*) – Variable to store the value.

### 2.3.188.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.189 `fira_helper_set_session_lower_aoa_bound_config_azimuth_2pi`

```
enum qerr fira_helper_set_session_lower_aoa_bound_config_azimuth_2pi(struct fira_context *ctx, uint32_t
                                                                      session_handle, const int32_t
                                                                      lower_aoa_bound_config_azimuth_2pi)
```

*[Not supported in QM33 SDK]* Sets ntf lower bound aoa azimuth.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.

- **lower\_aoa\_bound\_config\_azimuth\_2pi** (const int32\_t) – Lower bound in rad\_2pi for AOA azimuth, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -180° to +180°. should be less than or equal to SESSION\_INFO\_NTF\_UPPER\_BOUND\_AOA\_AZIMUTH value. (default = -180).

### 2.3.189.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.190 **fira\_helper\_get\_session\_lower\_aoa\_bound\_config\_azimuth\_2pi**

```
enum qerr fira_helper_get_session_lower_aoa_bound_config_azimuth_2pi(struct fira_context *ctx, uint32_t
                                                                    session_handle, int32_t
                                                                    *lower_aoa_bound_config_azimuth_2pi)
```

*[Not supported in QM33 SDK]* Gets ntf lower bound aoa azimuth.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **lower\_aoa\_bound\_config\_azimuth\_2pi** (int32\_t\*) – Lower bound in rad\_2pi for AOA azimuth, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -180° to +180°. should be less than or equal to SESSION\_INFO\_NTF\_UPPER\_BOUND\_AOA\_AZIMUTH value. (default = -180).

### 2.3.190.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.191 **fira\_helper\_set\_session\_upper\_aoa\_bound\_config\_azimuth\_2pi**

```
enum qerr fira_helper_set_session_upper_aoa_bound_config_azimuth_2pi(struct fira_context *ctx, uint32_t
                                                                    session_handle, const int32_t
                                                                    data_ntf_upper_bound_aoa_azimuth_2pi)
```

*[Not supported in QM33 SDK]* Sets ntf upper bound aoa azimuth.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_ntf\_upper\_bound\_aoa\_azimuth\_2pi** (const int32\_t) – Upper bound in rad\_2pi for AOA azimuth, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -180° to +180°. Should be greater than or equal to SESSION\_INFO\_NTF\_LOWER\_BOUND\_AOA\_AZIMUTH value. (default = +180).

### 2.3.191.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.192 `fira_helper_get_session_upper_aoa_bound_config_azimuth_2pi`

```
enum qerr fira_helper_get_session_upper_aoa_bound_config_azimuth_2pi(struct fira_context *ctx, uint32_t
                                                                    session_handle, int32_t
                                                                    *data_ntf_upper_bound_aoa_azimuth_2pi)
```

*[Not supported in QM33 SDK]* Gets ntf upper bound aoa azimuth.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_ntf\_upper\_bound\_aoa\_azimuth\_2pi** (int32\_t\*) – Upper bound in rad\_2pi for AOA azimuth, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -180° to +180°. Should be greater than or equal to SESSION\_INFO\_NTF\_LOWER\_BOUND\_AOA\_AZIMUTH value. (default = +180).

### 2.3.192.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.193 `fira_helper_set_session_lower_aoa_bound_config_elevation_2pi`

```
enum qerr fira_helper_set_session_lower_aoa_bound_config_elevation_2pi(struct fira_context *ctx,
                                                                    uint32_t session_handle, const
                                                                    int16_t
                                                                    data_ntf_lower_bound_aoa_elevation_2pi)
```

*[Not supported in QM33 SDK]* Sets ntf lower bound aoa elevation.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_ntf\_lower\_bound\_aoa\_elevation\_2pi** (const int16\_t) – Lower bound in rad\_2pi for AOA elevation, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -90° to +90°. Should be less than or equal to SESSION\_INFO\_NTF\_UPPER\_BOUND\_AOA\_ELEVATION value. (default = -90).

### 2.3.193.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.194 `fira_helper_get_session_lower_aoa_bound_config_elevation_2pi`

```
enum qerr fira_helper_get_session_lower_aoa_bound_config_elevation_2pi(struct fira_context *ctx,
                                                                    uint32_t session_handle,
                                                                    int16_t
                                                                    *data_ntf_lower_bound_aoa_elevation_2pi)
```

*[Not supported in QM33 SDK]* Gets ntf lower bound aoa elevation.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_ntf\_lower\_bound\_aoa\_elevation\_2pi** (int16\_t\*) – Lower bound in rad\_2pi for AOA elevation, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -90° to +90°. Should be less than or equal to SESSION\_INFO\_NTF\_UPPER\_BOUND\_AOA\_ELEVATION value. (default = -90).

#### 2.3.194.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.195 `fira_helper_set_session_upper_aoa_bound_config_elevation_2pi`

```
enum qerr fira_helper_set_session_upper_aoa_bound_config_elevation_2pi(struct fira_context *ctx,
                                                                    uint32_t session_handle, const
                                                                    int16_t
                                                                    data_ntf_upper_bound_aoa_elevation_2pi)
```

*[Not supported in QM33 SDK]* Sets ntf upper bound aoa elevation.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_ntf\_upper\_bound\_aoa\_elevation\_2pi** (const int16\_t) – Upper bound in rad\_2pi for AOA elevation, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -90° to +90°. Should be greater than or equal to SESSION\_INFO\_NTF\_LOWER\_BOUND\_AOA\_ELEVATION value. (default = +90).

#### 2.3.195.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.196 `fira_helper_get_session_upper_aoa_bound_config_elevation_2pi`

```
enum qerr fira_helper_get_session_upper_aoa_bound_config_elevation_2pi(struct fira_context *ctx,
                                                                    uint32_t session_handle,
                                                                    int16_t
                                                                    *data_ntf_upper_bound_aoa_elevation_2
```

*[Not supported in QM33 SDK]* Gets ntf upper bound aoa elevation.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **data\_ntf\_upper\_bound\_aoa\_elevation\_2pi** (int16\_t\*) – Upper bound in rad\_2pi for AOA elevation, applicable when session\_info\_ntf\_config is set to 0x03, 0x04, 0x06 or 0x07. It is a signed value (rad\_2pi). Allowed values range from -90° to +90°. Should be greater than or equal to SESSION\_INFO\_NTF\_LOWER\_BOUND\_AOA\_ELEVATION value. (default = +90).

#### 2.3.196.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.197 `fira_helper_set_session_key`

```
enum qerr fira_helper_set_session_key(struct fira_context *ctx, uint32_t session_handle, const void *key,
                                                                    uint8_t size)
```

Sets this key for the session.

#### Parameters

- **ctx** (struct fira\_context\*) – FiRa helper context.
- **session\_handle** (uint32\_t) – Handle of the session to modify.
- **key** (const void\*) – Pointer to the session key
- **size** (uint8\_t) – Length of the session key, can be 128 or 256 bits.

#### 2.3.197.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.198 `fira_helper_set_sub_session_key`

```
enum qerr fira_helper_set_sub_session_key(struct fira_context *ctx, uint32_t session_handle, const void *key,
                                                                    uint8_t size)
```

Sets key for the current controlee sub-session.

#### Parameters

- **ctx** (struct fira\_context\*) – FiRa helper context.
- **session\_handle** (uint32\_t) – Handle of the session to modify.
- **key** (const void\*) – Pointer to the sub-session key

- **size** (uint8\_t) – Length of the sub-session key, can be 128 or 256 bits.

### 2.3.198.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.199 **fira\_helper\_set\_session\_in\_band\_termination\_attempt\_count**

enum qerr **fira\_helper\_set\_session\_in\_band\_termination\_attempt\_count**(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t termination\_count)

Sets in band termination attempt count.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **termination\_count** (uint8\_t) – In band termination attempt count to set

### 2.3.199.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.200 **fira\_helper\_update\_dt\_anchor\_ranging\_rounds**

enum qerr **fira\_helper\_update\_dt\_anchor\_ranging\_rounds**(struct fira\_context \*ctx, uint32\_t session\_handle, struct [update\\_dt\\_anchor\\_ranging\\_rounds\\_cmd](#) \*cmd, struct [update\\_dt\\_anchor\\_ranging\\_rounds\\_rsp](#) \*rsp)

[Not supported in QM33 SDK] Configure ranging rounds for DT-Anchor.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **cmd** (struct [update\\_dt\\_anchor\\_ranging\\_rounds\\_cmd](#)\*) – Command containing configuration parameters of the ranging rounds.
- **rsp** (struct [update\\_dt\\_anchor\\_ranging\\_rounds\\_rsp](#)\*) – Response containing indexes of ranging rounds which failed to be configured.

### 2.3.200.1 Return

Error code if the command cannot be executed.

The return code will be QERR\_SUCCESS if the command is valid, then the result of the command execution will be in the status field of the corresponding response.



### 2.3.201 `fira_helper_dt_tag_configure_ranging_rounds`

```
enum qerr fira_helper_dt_tag_configure_ranging_rounds(struct fira_context *ctx, uint32_t session_handle,
                                                    struct dt_tag_ranging_rounds_config
                                                    *ranging_rounds_config, struct
                                                    dt_tag_round_indexes_rsp *round_indexes_rsp)
```

*[Not supported in QM33 SDK]* Configure ranging rounds for DT-Tag.

#### Parameters

- **ctx** (struct `fira_context*`) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **ranging\_rounds\_config** (struct `dt_tag_ranging_rounds_config*`) – Configuration parameters of the ranging rounds.
- **round\_indexes\_rsp** (struct `dt_tag_round_indexes_rsp*`) – Indexes of ranging rounds which failed to be configured.

#### 2.3.201.1 Return

QERR\_SUCCESS or error if the command cannot be executed.

The return code will be QERR\_SUCCESS if the command is valid, then the result of the command execution will be in the status field of the corresponding response.

### 2.3.202 `fira_helper_set_session_ut_tx_interval_ms`

```
enum qerr fira_helper_set_session_ut_tx_interval_ms(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t interval)
```

*[Not supported in QM33 SDK]* Set time interval between UTMs.

#### Parameters

- **ctx** (struct `fira_context*`) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **interval** (uint32\_t) – Time interval between UTMs (in ms).

#### 2.3.202.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.203 `fira_helper_set_session_ut_random_window`

```
enum qerr fira_helper_set_session_ut_random_window(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t window)
```

*[Not supported in QM33 SDK]* Set duration of random window in which UTMs can be send.

#### Parameters

- **ctx** (struct `fira_context*`) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **window** (uint32\_t) – Random window for UTMs (in ms).

### 2.3.203.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.204 `fira_helper_set_session_ut_tx_timestamp_len`

enum qerr `fira_helper_set_session_ut_tx_timestamp_len`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t len)

*[Not supported in QM33 SDK]* Set length of timestamp in UTMs.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **len** (uint8\_t) – Length of timestamps included in UTMs not present (0, default), 40-bit timestamp (1), 64-bit timestamp (2).

### 2.3.204.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.205 `fira_helper_set_session_ut_device_id_len`

enum qerr `fira_helper_set_session_ut_device_id_len`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t len)

*[Not supported in QM33 SDK]* Set UL-TDoA device id length.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **len** (uint8\_t) – Length of device id not present (0, default), 16-bit (1), 32-bit timestamp (2), 64-bit timestamp (3).

### 2.3.205.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.206 `fira_helper_set_session_ut_device_id`

enum qerr `fira_helper_set_session_ut_device_id`(struct fira\_context \*ctx, uint32\_t session\_handle, uint64\_t id)

*[Not supported in QM33 SDK]* Set value of UL-TDoA device id.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **id** (uint64\_t) – Device id

### 2.3.206.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.207 `fira_helper_set_session_ut_report_config_interval`

enum qerr `fira_helper_set_session_ut_report_config_interval`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t interval)

*[Not supported in QM33 SDK]* Set value of UT-Anchor report config time interval.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **interval** (uint8\_t) – Time interval between reports.

### 2.3.207.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.208 `fira_helper_set_session_ut_report_config_count`

enum qerr `fira_helper_set_session_ut_report_config_count`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t count)

*[Not supported in QM33 SDK]* Set value of UT-Anchor report config count.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **count** (uint8\_t) – Measurement count between reports.

### 2.3.208.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.209 `fira_helper_set_session_ut_report_config_event`

enum qerr `fira_helper_set_session_ut_report_config_event`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t event)

*[Not supported in QM33 SDK]* Set value of UT-Anchor report config events type.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **event** (uint8\_t) – Rx event to be reported.

### 2.3.209.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.210 `fira_helper_get_session_ut_tx_interval_ms`

```
enum qerr fira_helper_get_session_ut_tx_interval_ms(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t *interval)
```

*[Not supported in QM33 SDK]* Get time interval between UTMs.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **interval** (uint32\_t\*) – Time interval between UTMs (in ms).

### 2.3.210.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.211 `fira_helper_get_session_ut_random_window`

```
enum qerr fira_helper_get_session_ut_random_window(struct fira_context *ctx, uint32_t session_handle,
                                                    uint32_t *window)
```

*[Not supported in QM33 SDK]* Set duration of random window in which UTMs can be send.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **window** (uint32\_t\*) – Random window for UTMs (in ms).

### 2.3.211.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.212 `fira_helper_get_session_ut_tx_timestamp_len`

```
enum qerr fira_helper_get_session_ut_tx_timestamp_len(struct fira_context *ctx, uint32_t session_handle,
                                                       uint8_t *len)
```

*[Not supported in QM33 SDK]* Get length of timestamp in UTMs.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **len** (uint8\_t\*) – Length of timestamps included in UTMs not present (0, default), 40-bit timestamp (1), 64-bit timestamp (2).

### 2.3.212.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.213 `fira_helper_get_session_ut_device_id_len`

enum qerr `fira_helper_get_session_ut_device_id_len`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*len)

*[Not supported in QM33 SDK]* Get UL-TDoA device id length.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **len** (uint8\_t\*) – Length of device id not present (0, default), 16-bit (1), 32-bit timestamp (2), 64-bit timestamp (3).

### 2.3.213.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.214 `fira_helper_get_session_ut_device_id`

enum qerr `fira_helper_get_session_ut_device_id`(struct fira\_context \*ctx, uint32\_t session\_handle, uint64\_t \*id)

*[Not supported in QM33 SDK]* Get value of UL-TDoA device id.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **id** (uint64\_t\*) – Device id

### 2.3.214.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.215 `fira_helper_get_session_ut_report_config_interval`

enum qerr `fira_helper_get_session_ut_report_config_interval`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*interval)

*[Not supported in QM33 SDK]* Get value of UT-Anchor report config time interval.

#### Parameters

- **ctx** (struct fira\_context\*) – Fira helper context.
- **session\_handle** (uint32\_t) – Session handle.
- **interval** (uint8\_t\*) – Time interval between reports.

### 2.3.215.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.216 `fira_helper_get_session_ut_report_config_count`

enum qerr `fira_helper_get_session_ut_report_config_count`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*count)

*[Not supported in QM33 SDK]* Get value of UT-Anchor report config count.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `count` (uint8\_t\*) – Measurement count between reports.

### 2.3.216.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.217 `fira_helper_get_session_ut_report_config_event`

enum qerr `fira_helper_get_session_ut_report_config_event`(struct fira\_context \*ctx, uint32\_t session\_handle, uint8\_t \*event)

*[Not supported in QM33 SDK]* Get value of UT-Anchor report config events type.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `event` (uint8\_t\*) – Rx event to be reported.

### 2.3.217.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.218 `fira_helper_session_get_data_size_in_ranging`

enum qerr `fira_helper_session_get_data_size_in_ranging`(struct fira\_context \*ctx, uint32\_t session\_handle, uint16\_t \*size\_in\_ranging)

*[Not supported in QM33 SDK]* Get maximum data size in ranging round.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle.
- `size_in_ranging` (uint16\_t\*) – Data size in ranging.

### 2.3.218.1 Return

QERR\_SUCCESS on success, an error otherwise.

### 2.3.219 `fira_helper_set_hus_controller_config`

enum qerr `fira_helper_set_hus_controller_config`(struct fira\_context \*ctx, uint32\_t session\_handle, struct [fira\\_hus\\_controller\\_config\\_cmd](#) \*cmd)

*[Not supported in QM33 SDK]* Configure phases of a HUS ranging round.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle of the targeted HUS controller.
- `cmd` (struct [fira\\_hus\\_controller\\_config\\_cmd](#)\*) – Command containing configuration parameters of each phase to bind.

### 2.3.219.1 Return

QERR\_SUCCESS or an error otherwise. The corresponding FiRa status code is present if return value > 0.

### 2.3.220 `fira_helper_set_hus_controlee_config`

enum qerr `fira_helper_set_hus_controlee_config`(struct fira\_context \*ctx, uint32\_t session\_handle, struct [fira\\_hus\\_controlee\\_config\\_cmd](#) \*cmd)

*[Not supported in QM33 SDK]* Configure phases of a HUS ranging round.

#### Parameters

- `ctx` (struct fira\_context\*) – Fira helper context.
- `session_handle` (uint32\_t) – Session handle of the targeted HUS controller.
- `cmd` (struct [fira\\_hus\\_controlee\\_config\\_cmd](#)\*) – Command containing configuration parameters of each phase to bind.

### 2.3.220.1 Return

QERR\_SUCCESS or an error otherwise. The corresponding FiRa status code is present if return value > 0.

### 2.3.221 `fira_helper_set_session_application_data_endpoint`

enum qerr `fira_helper_set_session_application_data_endpoint`(struct fira\_context \*context, uint32\_t session\_handle, uint8\_t application\_data\_endpoint)

*[Not supported in QM33 SDK]* Application data endpoint setter.

#### Parameters

- `context` (struct fira\_context\*) – Fira context.
- `session_handle` (uint32\_t) – Session handle.

- **application\_data\_endpoint** (uint8\_t) – Endpoint for non-secure/secure data message exchange.

### 2.3.221.1 Return

QERR\_SUCCESS or error.

### 2.3.222 **fira\_helper\_get\_session\_application\_data\_endpoint**

enum qerr **fira\_helper\_get\_session\_application\_data\_endpoint**(struct fira\_context \*context, uint32\_t session\_handle, uint8\_t \*application\_data\_endpoint)

*[Not supported in QM33 SDK]* Application data endpoint getter.

#### Parameters

- **context** (struct fira\_context\*) – Fira context.
- **session\_handle** (uint32\_t) – Session handle.
- **application\_data\_endpoint** (uint8\_t\*) – Endpoint for non-secure/secure data message exchange.

### 2.3.222.1 Return

QERR\_SUCCESS or error.

### 2.3.223 **enum fira\_dt\_anchor\_acting\_role**

enum **fira\_dt\_anchor\_acting\_role**

*[Not supported in QM33 SDK]* Internal role played by a DT-Anchor during the particular ranging round.

### 2.3.223.1 Definition

```
enum fira_dt_anchor_acting_role {
    FIRA_DT_ANCHOR_ACTING_RESPONDER,
    FIRA_DT_ANCHOR_ACTING_INITIATOR
};
```

### 2.3.223.2 Constants

#### **FIRA\_DT\_ANCHOR\_ACTING\_RESPONDER**

The DT-Anchor acts as a responder.

#### **FIRA\_DT\_ANCHOR\_ACTING\_INITIATOR**

The DT-Anchor acts as an initiator.



### 2.3.224 enum fira\_ranging\_round\_usage

enum fira\_ranging\_round\_usage

Ranging mode.

#### 2.3.224.1 Definition

```
enum fira_ranging_round_usage {
    FIRA_RANGING_ROUND_USAGE_OWR_UL_TDOA,
    FIRA_RANGING_ROUND_USAGE_SSTWR_DEFERRED,
    FIRA_RANGING_ROUND_USAGE_DSTWR_DEFERRED,
    FIRA_RANGING_ROUND_USAGE_SSTWR_NON_DEFERRED,
    FIRA_RANGING_ROUND_USAGE_DSTWR_NON_DEFERRED,
    FIRA_RANGING_ROUND_USAGE_OWR_DL_TDOA,
    FIRA_RANGING_ROUND_USAGE_OWR_AOA,
    FIRA_RANGING_ROUND_USAGE_ESS_TWR_NON_DEFERRED_CONTENTION_BASED,
    FIRA_RANGING_ROUND_USAGE_ADS_TWR_CONTENTION_BASED,
    FIRA_RANGING_ROUND_USAGE_ASSIGNED,
    FIRA_RANGING_ROUND_USAGE_HYBRID_RANGING,
    FIRA_RANGING_ROUND_USAGE_MAX,
    FIRA_RANGING_ROUND_USAGE_UNDEFINED
};
```

#### 2.3.224.2 Constants

##### **FIRA\_RANGING\_ROUND\_USAGE\_OWR\_UL\_TDOA**

[Not supported in QM33 SDK] One Way Ranging UL-TDoA.

##### **FIRA\_RANGING\_ROUND\_USAGE\_SSTWR\_DEFERRED**

Single-Sided Two Way Ranging with Deferred Mode.

##### **FIRA\_RANGING\_ROUND\_USAGE\_DSTWR\_DEFERRED**

Dual-Sided Two Way Ranging with Deferred Mode.

##### **FIRA\_RANGING\_ROUND\_USAGE\_SSTWR\_NON\_DEFERRED**

Single-Sided Two Way Ranging with Non-Deferred Mode.

##### **FIRA\_RANGING\_ROUND\_USAGE\_DSTWR\_NON\_DEFERRED**

Dual-Sided Two Way Ranging with Non-Deferred Mode.

##### **FIRA\_RANGING\_ROUND\_USAGE\_OWR\_DL\_TDOA**

[Not supported in QM33 SDK] One Way Ranging for DownLink Time Difference of Arrival measurement.

##### **FIRA\_RANGING\_ROUND\_USAGE\_OWR\_AOA**

[Not supported in QM33 SDK] One Way Ranging for Angle of Arrival measurement

##### **FIRA\_RANGING\_ROUND\_USAGE\_ESS\_TWR\_NON\_DEFERRED\_CONTENTION\_BASED**

[Not supported in QM33 SDK] Enhanced SingleSided Two Way Ranging with Non-deferred Mode for Contention-based ranging.

##### **FIRA\_RANGING\_ROUND\_USAGE\_ADS\_TWR\_CONTENTION\_BASED**

[Not supported in QM33 SDK] Alternative Dual-Sided Two Way Ranging for Contention-based ranging.

##### **FIRA\_RANGING\_ROUND\_USAGE\_ASSIGNED**

RFU.

##### **FIRA\_RANGING\_ROUND\_USAGE\_HYBRID\_RANGING**

[Not supported in QM33 SDK] Hybrid Ranging Mode.

## FIRA\_RANGING\_ROUND\_USAGE\_MAX

Max value for the range check.

## FIRA\_RANGING\_ROUND\_USAGE\_UNDEFINED

Ranging round usage is undefined.

### 2.3.225 enum fira\_dl\_tdoa\_ranging\_method

enum **fira\_dl\_tdoa\_ranging\_method**

[Not supported in QM33 SDK] Ranging method used by DT-Anchors in DL-TDoA.

#### 2.3.225.1 Definition

```
enum fira_dl_tdoa_ranging_method {
    FIRA_DL_TDOA_RANGING_METHOD_SSTWR,
    FIRA_DL_TDOA_RANGING_METHOD_DSTWR
};
```

#### 2.3.225.2 Constants

## FIRA\_DL\_TDOA\_RANGING\_METHOD\_SSTWR

Single-Sided Two Way Ranging.

## FIRA\_DL\_TDOA\_RANGING\_METHOD\_DSTWR

Dual-Sided Two Way Ranging.

### 2.3.226 enum fira\_multi\_node\_mode

enum **fira\_multi\_node\_mode**

Multi-node mode.

#### 2.3.226.1 Definition

```
enum fira_multi_node_mode {
    FIRA_MULTI_NODE_MODE_UNICAST,
    FIRA_MULTI_NODE_MODE_ONE_TO_MANY,
    FIRA_MULTI_NODE_MODE_MANY_TO_MANY,
    FIRA_MULTI_NODE_MODE_UNDEFINED
};
```

### 2.3.226.2 Constants

#### **FIRA\_MULTI\_NODE\_MODE\_UNICAST**

Ranging between one initiator and one responder.

#### **FIRA\_MULTI\_NODE\_MODE\_ONE\_TO\_MANY**

Ranging between one initiator and multiple responders.

#### **FIRA\_MULTI\_NODE\_MODE\_MANY\_TO\_MANY**

*[Not supported in QM33 SDK]* Ranging between multiple initiators and multiple responders.

#### **FIRA\_MULTI\_NODE\_MODE\_UNDEFINED**

Ranging mode is undefined.

### 2.3.227 enum `fira_measurement_report_originator`

enum `fira_measurement_report_originator`

Originator (author) of the Ranging Measurement Report Message (MRM).

#### 2.3.227.1 Definition

```
enum fira_measurement_report_originator {
    FIRA_MEASUREMENT_REPORT_FROM_INITIATOR,
    FIRA_MEASUREMENT_REPORT_FROM_RESPONDER
};
```

### 2.3.227.2 Constants

#### **FIRA\_MEASUREMENT\_REPORT\_FROM\_INITIATOR**

The initiator sends a measurement report message.

#### **FIRA\_MEASUREMENT\_REPORT\_FROM\_RESPONDER**

The responder sends a measurement report message.

### 2.3.228 enum `fira_measurement_report_type`

enum `fira_measurement_report_type`

Type of the current Ranging Measurement Report Message (MRM).

#### 2.3.228.1 Definition

```
enum fira_measurement_report_type {
    FIRA_MEASUREMENT_REPORT_TYPE_1,
    FIRA_MEASUREMENT_REPORT_TYPE_2,
    FIRA_MEASUREMENT_REPORT_TYPE_3
};
```

### 2.3.228.2 Constants

#### FIRA\_MEASUREMENT\_REPORT\_TYPE\_1

Measurement report message type 1.

#### FIRA\_MEASUREMENT\_REPORT\_TYPE\_2

Measurement report message type 2.

#### FIRA\_MEASUREMENT\_REPORT\_TYPE\_3

[Not supported in QM33 SDK] Measurement report message type 3.

### 2.3.229 enum `fira_owr_message_type`

enum `fira_owr_message_type`

[Not supported in QM33 SDK] Type of the current One Way Ranging (OWR) Message (FiRa v2.0 5.9.13 OWR Message).

#### 2.3.229.1 Definition

```
enum fira_owr_message_type {
    FIRA_OWR_MESSAGE_TYPE_UT_BLINK,
    FIRA_OWR_MESSAGE_TYPE_UT_SYNC,
    FIRA_OWR_MESSAGE_TYPE_DT_POLL,
    FIRA_OWR_MESSAGE_TYPE_DT_RESPONSE,
    FIRA_OWR_MESSAGE_TYPE_DT_FINAL,
    FIRA_OWR_MESSAGE_TYPE_AOA_MEASUREMENT
};
```

### 2.3.229.2 Constants

#### FIRA\_OWR\_MESSAGE\_TYPE\_UT\_BLINK

UL-TDOA Blink Message.

#### FIRA\_OWR\_MESSAGE\_TYPE\_UT\_SYNC

UL-TDOA Synchronization Message.

#### FIRA\_OWR\_MESSAGE\_TYPE\_DT\_POLL

DL-TDOA Poll Message.

#### FIRA\_OWR\_MESSAGE\_TYPE\_DT\_RESPONSE

DL-TDOA Response Message.

#### FIRA\_OWR\_MESSAGE\_TYPE\_DT\_FINAL

DL-TDOA Final Message.

#### FIRA\_OWR\_MESSAGE\_TYPE\_AOA\_MEASUREMENT

AoA Measurement Message.

### 2.3.230 enum fira\_schedule\_mode

enum **fira\_schedule\_mode**

Slot scheduling mode used during the ranging session.

#### 2.3.230.1 Definition

```
enum fira_schedule_mode {
    FIRA_SCHEDULE_MODE_CONTENTION_BASED,
    FIRA_SCHEDULE_MODE_TIME_SCHEDULED,
    FIRA_SCHEDULE_MODE_HYBRID_BASED,
    FIRA_SCHEDULE_MODE_MAX,
    FIRA_SCHEDULE_MODE_UNDEFINED
};
```

#### 2.3.230.2 Constants

**FIRA\_SCHEDULE\_MODE\_CONTENTION\_BASED**

*[Not supported in QM33 SDK]* Contention-based ranging.

**FIRA\_SCHEDULE\_MODE\_TIME\_SCHEDULED**

Time-scheduled ranging.

**FIRA\_SCHEDULE\_MODE\_HYBRID\_BASED**

*[Not supported in QM33 SDK]* Hybrid-based ranging.

**FIRA\_SCHEDULE\_MODE\_MAX**

Max value defined

**FIRA\_SCHEDULE\_MODE\_UNDEFINED**

Scheduled mode is undefined.

### 2.3.231 enum fira\_rframe\_config

enum **fira\_rframe\_config**

Rframe configuration used to transmit/receive ranging messages.

#### 2.3.231.1 Definition

```
enum fira_rframe_config {
    FIRA_RFRAME_CONFIG_SP0,
    FIRA_RFRAME_CONFIG_SP1,
    FIRA_RFRAME_CONFIG_SP2,
    FIRA_RFRAME_CONFIG_SP3
};
```

### 2.3.231.2 Constants

#### FIRA\_RFRAME\_CONFIG\_SP0

Use SP0 mode. (Applicable only for PCTT)

#### FIRA\_RFRAME\_CONFIG\_SP1

Use SP1 mode.

#### FIRA\_RFRAME\_CONFIG\_SP2

RFU

#### FIRA\_RFRAME\_CONFIG\_SP3

Use SP3 mode.

### 2.3.232 enum fira\_prf\_mode

enum fira\_prf\_mode

Pulse Repetition Frequency mode

#### 2.3.232.1 Definition

```
enum fira_prf_mode {
    FIRA_PRF_MODE_BPRF,
    FIRA_PRF_MODE_HPRF,
    FIRA_PRF_MODE_HPRF_HIGH_RATE
};
```

#### 2.3.232.2 Constants

##### FIRA\_PRF\_MODE\_BPRF

Base Pulse Repetition Frequency.

##### FIRA\_PRF\_MODE\_HPRF

[Not supported in QM33 SDK] Higher Pulse Repetition Frequency.

##### FIRA\_PRF\_MODE\_HPRF\_HIGH\_RATE

[Not supported in QM33 SDK] Higher Pulse Repetition Frequency allows high data rate (27.2 Mbps and 31.2 Mbps).

#### 2.3.232.3 Description

This enum is not used in the current implementation.

### 2.3.233 enum fira\_preamble\_duration

enum **fira\_preamble\_duration**  
Duration of preamble in symbols.

#### 2.3.233.1 Definition

```
enum fira_preamble_duration {
    FIRA_PREAMBLE_DURATION_32,
    FIRA_PREAMBLE_DURATION_64
};
```

#### 2.3.233.2 Constants

**FIRA\_PREAMBLE\_DURATION\_32**  
*[Not supported in QM33 SDK]* 32 symbols duration.

**FIRA\_PREAMBLE\_DURATION\_64**  
64 symbols duration.

### 2.3.234 enum fira\_sfd\_id

enum **fira\_sfd\_id**  
Start-of-frame delimiter.

#### 2.3.234.1 Definition

```
enum fira_sfd_id {
    FIRA_SFD_ID_0,
    FIRA_SFD_ID_1,
    FIRA_SFD_ID_2,
    FIRA_SFD_ID_3,
    FIRA_SFD_ID_4
};
```

#### 2.3.234.2 Constants

**FIRA\_SFD\_ID\_0**  
Delimiter is [0 +1 0 -1 +1 0 0 -1]

**FIRA\_SFD\_ID\_1**  
*[Not supported in QM33 SDK]* Delimiter is [ -1 -1 +1 -1 ]

**FIRA\_SFD\_ID\_2**  
Delimiter is [ -1 -1 -1 +1 -1 -1 +1 -1 ]

**FIRA\_SFD\_ID\_3**  
*[Not supported in QM33 SDK]* Delimiter is [ -1 -1 -1 -1 -1 +1 +1 -1 -1 +1 -1 +1 -1 +1 -1 ]

**FIRA\_SFD\_ID\_4**  
*[Not supported in QM33 SDK]* Delimiter is [ -1 -1 -1 -1 -1 -1 +1 -1 -1 +1 -1 -1 +1 -1 +1 -1 +1 -1 -1 -1 +1 +1 -1 -1 +1 -1 +1 +1 -1 -1 ]

### 2.3.235 enum `fira_sts_segments`

enum `fira_sts_segments`  
Number of STS segments.

#### 2.3.235.1 Definition

```
enum fira_sts_segments {
    FIRA_STS_SEGMENTS_0,
    FIRA_STS_SEGMENTS_1,
    FIRA_STS_SEGMENTS_2,
    FIRA_STS_SEGMENTS_3,
    FIRA_STS_SEGMENTS_4
};
```

#### 2.3.235.2 Constants

**FIRA\_STS\_SEGMENTS\_0**  
No STS Segment (Rframe config SP0).

**FIRA\_STS\_SEGMENTS\_1**  
1 STS Segment.

**FIRA\_STS\_SEGMENTS\_2**  
*[Not supported in QM33 SDK]* 2 STS Segments.

**FIRA\_STS\_SEGMENTS\_3**  
*[Not supported in QM33 SDK]* 3 STS Segments.

**FIRA\_STS\_SEGMENTS\_4**  
*[Not supported in QM33 SDK]* 4 STS Segments.

### 2.3.236 enum `fira_psdu_data_rate`

enum `fira_psdu_data_rate`  
Data rate used to exchange PSDUs.

#### 2.3.236.1 Definition

```
enum fira_psdu_data_rate {
    FIRA_PSDU_DATA_RATE_6M81,
    FIRA_PSDU_DATA_RATE_7M80,
    FIRA_PSDU_DATA_RATE_27M2,
    FIRA_PSDU_DATA_RATE_31M2
};
```



### 2.3.236.2 Constants

#### FIRA\_PSDU\_DATA\_RATE\_6M81

6.8Mb/s rate.

#### FIRA\_PSDU\_DATA\_RATE\_7M80

[Not supported in QM33 SDK] 7.8Mb/s rate.

#### FIRA\_PSDU\_DATA\_RATE\_27M2

[Not supported in QM33 SDK] 27.2Mb/s rate.

#### FIRA\_PSDU\_DATA\_RATE\_31M2

[Not supported in QM33 SDK] 31.2Mb/s rate.

### 2.3.237 enum fira\_phr\_data\_rate

enum **fira\_phr\_data\_rate**

Data rate used to exchange PHR.

#### 2.3.237.1 Definition

```
enum fira_phr_data_rate {
    FIRA_PHR_DATA_RATE_850K,
    FIRA_PHR_DATA_RATE_6M81
};
```

### 2.3.237.2 Constants

#### FIRA\_PHR\_DATA\_RATE\_850K

850kb/s rate.

#### FIRA\_PHR\_DATA\_RATE\_6M81

6.8Mb/s rate.

### 2.3.238 enum fira\_mac\_fcs\_type

enum **fira\_mac\_fcs\_type**

Length of Frame Check Sequence.

#### 2.3.238.1 Definition

```
enum fira_mac_fcs_type {
    FIRA_MAC_FCS_TYPE_CRC_16,
    FIRA_MAC_FCS_TYPE_CRC_32
};
```

### 2.3.238.2 Constants

#### FIRA\_MAC\_FCS\_TYPE\_CRC\_16

2 bytes sequence.

#### FIRA\_MAC\_FCS\_TYPE\_CRC\_32

4 bytes sequence.

### 2.3.239 enum fira\_data\_transfer\_status

enum fira\_data\_transfer\_status

[Not supported in QM33 SDK] Data transfer status.

#### 2.3.239.1 Definition

```
enum fira_data_transfer_status {
    FIRA_DATA_TRANSFER_STATUS_REPETITION_OK,
    FIRA_DATA_TRANSFER_STATUS_OK,
    FIRA_DATA_TRANSFER_STATUS_ERROR_DATA_TRANSFER,
    FIRA_DATA_TRANSFER_STATUS_ERROR_NO_CREDIT_AVAILABLE,
    FIRA_DATA_TRANSFER_STATUS_ERROR_REJECTED,
    FIRA_DATA_TRANSFER_STATUS_SESSION_TYPE_NOT_SUPPORTED,
    FIRA_DATA_TRANSFER_STATUS_ERROR_DATA_TRANSFER_IS_ONGOING,
    FIRA_DATA_TRANSFER_STATUS_INVALID_FORMAT
};
```

### 2.3.239.2 Constants

#### FIRA\_DATA\_TRANSFER\_STATUS\_REPETITION\_OK

If DATA\_REPETITION\_COUNT>0 and if SESSION\_DATA\_TRANSFER\_STATUS\_NTF\_CONFIG = Enable; it indicates that one Data transmission is completed in a RR.

#### FIRA\_DATA\_TRANSFER\_STATUS\_OK

For TWR - it indicates that the Application Data transmission is completed. For OWR - it indicates that the Application Data transmission and its repetitions of DATA\_REPETITION\_COUNT is completed.

#### FIRA\_DATA\_TRANSFER\_STATUS\_ERROR\_DATA\_TRANSFER

Application Data couldn't be sent due to an unrecoverable error.

#### FIRA\_DATA\_TRANSFER\_STATUS\_ERROR\_NO\_CREDIT\_AVAILABLE

DATA\_MESSAGE\_SND is not accepted as no credit is available.

#### FIRA\_DATA\_TRANSFER\_STATUS\_ERROR\_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.

#### FIRA\_DATA\_TRANSFER\_STATUS\_SESSION\_TYPE\_NOT\_SUPPORTED

Data transfer is not supported for given session type.

#### FIRA\_DATA\_TRANSFER\_STATUS\_ERROR\_DATA\_TRANSFER\_IS\_ONGOING

Application Data is being transmitted and the number of configured DATA\_REPETITION\_COUNT transmissions is not yet completed.

#### FIRA\_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).

### 2.3.240 enum `fira_measurement_type`

enum `fira_measurement_type`

The different type of available measurements.

#### 2.3.240.1 Definition

```
enum fira_measurement_type {
    FIRA_MEASUREMENT_TYPE_RANGE,
    FIRA_MEASUREMENT_TYPE_AOA,
    FIRA_MEASUREMENT_TYPE_AOA_AZIMUTH,
    FIRA_MEASUREMENT_TYPE_AOA_ELEVATION,
    FIRA_MEASUREMENT_TYPE_AOA_AZIMUTH_ELEVATION,
    __FIRA_MEASUREMENT_TYPE_AFTER_LAST
};
```

#### 2.3.240.2 Constants

**FIRA\_MEASUREMENT\_TYPE\_RANGE**

Measure only range.

**FIRA\_MEASUREMENT\_TYPE\_AOA**

Measure range + unspecified AoA.

**FIRA\_MEASUREMENT\_TYPE\_AOA\_AZIMUTH**

Measure range + azimuth.

**FIRA\_MEASUREMENT\_TYPE\_AOA\_ELEVATION**

Measure range + elevation.

**FIRA\_MEASUREMENT\_TYPE\_AOA\_AZIMUTH\_ELEVATION**

Measure range+azimuth+elevation.

**\_\_FIRA\_MEASUREMENT\_TYPE\_AFTER\_LAST**

Internal use.

### 2.3.241 struct `fira_session_time_base`

struct `fira_session_time_base`

[Not supported in QM33 SDK] Session time base information.

#### 2.3.241.1 Definition

```
struct fira_session_time_base {
    uint8_t config;
    uint32_t session_handle;
    uint32_t time_offset_us;
};
```

### 2.3.241.2 Members

#### config

Time base configuration bitfield. b0: 1:enable - 0:disable b1: 1:continue - 0:stop b2: 1:resync - 0:no resync

#### session\_handle

Session handle of the reference session.

#### time\_offset\_us

Time offset in microseconds.

### 2.3.242 struct fira\_measurement\_sequence\_step

struct **fira\_measurement\_sequence\_step**

One measurement step configuration.

#### 2.3.242.1 Definition

```
struct fira_measurement_sequence_step {
    enum fira_measurement_type type;
    uint8_t n_measurements;
    int8_t rx_ant_set_nonranging;
    int8_t rx_ant_sets_ranging[2];
    int8_t tx_ant_set_nonranging;
    int8_t tx_ant_set_ranging;
}
```

#### 2.3.242.2 Members

##### type

The type of measurement.

##### n\_measurements

The number of measurement to do.

##### rx\_ant\_set\_nonranging

RX antenna set for non-ranging frames.

##### rx\_ant\_sets\_ranging

RX antenna sets for ranging frames.

##### tx\_ant\_set\_nonranging

TX antenna set for non-ranging frames.

##### tx\_ant\_set\_ranging

TX antenna set for ranging frames.

### 2.3.243 struct `fira_measurement_sequence`

struct `fira_measurement_sequence`  
Measurement sequence configuration.

#### 2.3.243.1 Definition

```
struct fira_measurement_sequence {
    uint8_t n_steps;
    struct fira_measurement_sequence_step steps[];
}
```

#### 2.3.243.2 Members

**n\_steps**  
Number of steps in the sequence.

**steps**  
Array of step configuration.

### 2.3.244 enum `fira_ranging_diagnostics_frame_report_flags`

enum `fira_ranging_diagnostics_frame_report_flags`  
Activation flags for different frame diagnostics information.

#### 2.3.244.1 Definition

```
enum fira_ranging_diagnostics_frame_report_flags {
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORT_NONE,
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORT_AOAS,
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORT_CFO,
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORT_EMITTER_SHORT_ADDR,
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORT_SEGMENT_METRICS,
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORT_CIRS,
    __FIRA_RANGING_DIAGNOSTICS_FRAME_REPORT_AFTER_LAST
};
```

#### 2.3.244.2 Constants

**FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORT\_NONE**  
No specific frame diagnostic report requested.

**FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORT\_AOAS**  
Report AOA in frame diagnostics.

**FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORT\_CFO**  
Report Clock Frequency Offset in report, as measured on the first Rx RFRAME in the round.

**FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORT\_EMITTER\_SHORT\_ADDR**  
Report the MAC emitter short address in frame diagnostics.

## FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORT\_SEGMENT\_METRICS

Report Segment Metrics in frame diagnostics.

## FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORT\_CIRS

Report CIR in frame diagnostics.

## \_\_FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORT\_AFTER\_LAST

Internal use.

### 2.3.245 enum fira\_ranging\_diagnostics\_frame\_reports\_status\_flags

enum **fira\_ranging\_diagnostics\_frame\_reports\_status\_flags**

Flags for the individual frame report's status bitfield.

#### 2.3.245.1 Definition

```
enum fira_ranging_diagnostics_frame_reports_status_flags {
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORTS_STATUS_FLAGS_SUCCESS,
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORTS_STATUS_FLAGS_WIFI_COEX,
    FIRA_RANGING_DIAGNOSTICS_FRAME_REPORTS_STATUS_FLAGS_GRANT_DURATION_EXCEEDED
};
```

#### 2.3.245.2 Constants

### FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORTS\_STATUS\_FLAGS\_SUCCESS

False when the frame Rx has failed for some reason. Always true for Tx.

### FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORTS\_STATUS\_FLAGS\_WIFI\_COEX

True if the Wifi Coex GPIO was on while transmitting the frame.

### FIRA\_RANGING\_DIAGNOSTICS\_FRAME\_REPORTS\_STATUS\_FLAGS\_GRANT\_DURATION\_EXCEEDED

True if the MAX\_GRANT\_DURATION has been exceeded.

### 2.3.246 enum fira\_sts\_length

enum **fira\_sts\_length**

Number of symbols in a STS segment.

#### 2.3.246.1 Definition

```
enum fira_sts_length {
    FIRA_STS_LENGTH_32,
    FIRA_STS_LENGTH_64,
    FIRA_STS_LENGTH_128
};
```

### 2.3.246.2 Constants

#### FIRA\_STS\_LENGTH\_32

The STS length is 32 symbols.

#### FIRA\_STS\_LENGTH\_64

The STS length is 64 symbols.

#### FIRA\_STS\_LENGTH\_128

The STS length is 128 symbols.

### 2.3.247 enum `fira_session_info_ntf_config`

enum `fira_session_info_ntf_config`

[Not supported in QM33 SDK] Configure session info notification.

#### 2.3.247.1 Definition

```
enum fira_session_info_ntf_config {
    FIRA_SESSION_INFO_NTF_CONFIG_DISABLED,
    FIRA_SESSION_INFO_NTF_CONFIG_ALWAYS,
    FIRA_SESSION_INFO_NTF_CONFIG_PROXIMITY,
    FIRA_SESSION_INFO_NTF_CONFIG_AOA,
    FIRA_SESSION_INFO_NTF_CONFIG_PROXIMITY_AND_AOA,
    FIRA_SESSION_INFO_NTF_CONFIG_PROXIMITY_CROSSING,
    FIRA_SESSION_INFO_NTF_CONFIG_AOA_CROSSING,
    FIRA_SESSION_INFO_NTF_CONFIG_PROXIMITY_AND_AOA_CROSSING
};
```

### 2.3.247.2 Constants

#### FIRA\_SESSION\_INFO\_NTF\_CONFIG\_DISABLED

Do not report range data.

#### FIRA\_SESSION\_INFO\_NTF\_CONFIG\_ALWAYS

Report range data.

#### FIRA\_SESSION\_INFO\_NTF\_CONFIG\_PROXIMITY

Report range data if it is within proximity range defined by proximity parameters (NEAR\_PROXIMITY\_CONFIG/FAR).

#### FIRA\_SESSION\_INFO\_NTF\_CONFIG\_AOA

Report range data in AoA upper and lower bound. defined by AOA parameters (FIRA\_SESSION\_PARAM\_ATTR\_SESSION\_INFO\_NTF\_UPPER/LOWER\_BOUND\_AOA\_AZIMUTH/ELEVATION)

#### FIRA\_SESSION\_INFO\_NTF\_CONFIG\_PROXIMITY\_AND\_AOA

Report range data in AoA upper and lower bound as well as in proximity range.

#### FIRA\_SESSION\_INFO\_NTF\_CONFIG\_PROXIMITY\_CROSSING

Same as FIRA\_SESSION\_INFO\_NTF\_CONFIG\_PROXIMITY, but issues notification on crossing of boundaries. As for now, same notif is sent for “enter” and “exit” events.

#### FIRA\_SESSION\_INFO\_NTF\_CONFIG\_AOA\_CROSSING

Same as FIRA\_SESSION\_INFO\_NTF\_CONFIG\_AOA, but issues notification on crossing of boundaries. As for now, same notif is sent for “enter” and “exit” events.

## FIRA\_SESSION\_INFO\_NTF\_CONFIG\_PROXIMITY\_AND\_AOA\_CROSSING

Same as FIRA\_SESSION\_INFO\_NTF\_CONFIG\_PROXIMITY\_AND\_AOA, but issues notification on crossing of boundaries. As for now, same notif is sent for “enter” and “exit” events.

### 2.3.248 enum fira\_link\_layer\_mode

enum **fira\_link\_layer\_mode**

Link layer behavior.

#### 2.3.248.1 Definition

```
enum fira_link_layer_mode {
    FIRA_LINK_LAYER_MODE_BYPASS,
    FIRA_LINK_LAYER_MODE_CONNECTION_LESS
};
```

#### 2.3.248.2 Constants

##### FIRA\_LINK\_LAYER\_MODE\_BYPASS

No link layer overhead is added to the MDSDU.

##### FIRA\_LINK\_LAYER\_MODE\_CONNECTION\_LESS

*[Not supported in QM33 SDK]* Link layer header is needed to provide addressing capabilities for data transmission.

### 2.3.249 enum fira\_message\_id

enum **fira\_message\_id**

Message identifiers, used in report and in messages.

#### 2.3.249.1 Definition

```
enum fira_message_id {
    FIRA_MESSAGE_ID_RANGING_INITIATION,
    FIRA_MESSAGE_ID_RANGING_RESPONSE,
    FIRA_MESSAGE_ID_RANGING_FINAL,
    FIRA_MESSAGE_ID_CONTROL,
    FIRA_MESSAGE_ID_MEASUREMENT_REPORT,
    FIRA_MESSAGE_ID_RESULT_REPORT,
    FIRA_MESSAGE_ID_CONTROL_UPDATE,
    FIRA_MESSAGE_ID_ONE_WAY_RANGING,
    FIRA_MESSAGE_ID_DATA,
    FIRA_MESSAGE_ID_RFRAME_MAX,
    FIRA_MESSAGE_ID_MAX
};
```



### 2.3.249.2 Constants

#### **FIRA\_MESSAGE\_ID\_RANGING\_INITIATION**

Ranging Initiation Message.

#### **FIRA\_MESSAGE\_ID\_RANGING\_RESPONSE**

Ranging Response Message.

#### **FIRA\_MESSAGE\_ID\_RANGING\_FINAL**

Ranging Final Message, only for DS-TWR.

#### **FIRA\_MESSAGE\_ID\_CONTROL**

Control Message, sent by the controller.

#### **FIRA\_MESSAGE\_ID\_MEASUREMENT\_REPORT**

Measurement Report Message.

#### **FIRA\_MESSAGE\_ID\_RESULT\_REPORT**

Ranging Result Report Message.

#### **FIRA\_MESSAGE\_ID\_CONTROL\_UPDATE**

Control Update Message.

#### **FIRA\_MESSAGE\_ID\_ONE\_WAY\_RANGING**

*[Not supported in QM33 SDK]* One Way Ranging Message (see internal types).

#### **FIRA\_MESSAGE\_ID\_DATA**

*[Not supported in QM33 SDK]* Data Message.

#### **FIRA\_MESSAGE\_ID\_RFRAME\_MAX**

Maximum identifier of messages transmitted as an RFRAME (without a payload).

#### **FIRA\_MESSAGE\_ID\_MAX**

Maximum identifier of all messages.

### 2.3.250 enum `fira_result_report_config_flags`

enum `fira_result_report_config_flags`

result report config flags.

#### 2.3.250.1 Definition

```
enum fira_result_report_config_flags {
    FIRA_RESULT_REPORT_CONFIG_REPORT_TOF,
    FIRA_RESULT_REPORT_CONFIG_REPORT_AOA_AZIMUTH,
    FIRA_RESULT_REPORT_CONFIG_REPORT_AOA_ELEVATION,
    FIRA_RESULT_REPORT_CONFIG_REPORT_AOA_FOM,
    FIRA_RESULT_REPORT_CONFIG_REPORT_ALL
};
```

### 2.3.250.2 Constants

#### FIRA\_RESULT\_REPORT\_CONFIG\_REPORT\_TOF

Report Time of flight.

#### FIRA\_RESULT\_REPORT\_CONFIG\_REPORT\_AOA\_AZIMUTH

Report azimuth angle.

#### FIRA\_RESULT\_REPORT\_CONFIG\_REPORT\_AOA\_ELEVATION

Report elevation angle.

#### FIRA\_RESULT\_REPORT\_CONFIG\_REPORT\_AOA\_FOM

report AoA figure of merit.

#### FIRA\_RESULT\_REPORT\_CONFIG\_REPORT\_ALL

Maximum value of the parameter.

### 2.3.251 enum fira\_data\_message\_status

enum fira\_data\_message\_status

[Not supported in QM33 SDK] status of data message receive notification.

#### 2.3.251.1 Definition

```
enum fira_data_message_status {
    FIRA_DATA_MESSAGE_STATUS_SUCCESS,
    FIRA_DATA_MESSAGE_STATUS_FAILED,
    FIRA_DATA_MESSAGE_STATUS_UNKNOWN
};
```

### 2.3.251.2 Constants

#### FIRA\_DATA\_MESSAGE\_STATUS\_SUCCESS

All data segments in the round were successfully received.

#### FIRA\_DATA\_MESSAGE\_STATUS\_FAILED

Intended operation failed to complete, e.g. incomplete ranging round with piggyback data.

#### FIRA\_DATA\_MESSAGE\_STATUS\_UNKNOWN

Failure due to unknown reason.

### 2.3.252 enum fira\_dt\_location\_coord\_system\_type

enum fira\_dt\_location\_coord\_system\_type

[Not supported in QM33 SDK] Coordinate System Type of a DT-Anchor.

### 2.3.252.1 Definition

```
enum fira_dt_location_coord_system_type {
    FIRA_DT_LOCATION_COORD_WGS84,
    FIRA_DT_LOCATION_COORD_RELATIVE,
    FIRA_DT_LOCATION_COORD_INVALID
};
```

### 2.3.252.2 Constants

#### FIRA\_DT\_LOCATION\_COORD\_WGS84

The location is given in WGS84 coordinate system (longitude, latitude, altitude,(see struct fira\_wgs84\_location).

#### FIRA\_DT\_LOCATION\_COORD\_RELATIVE

The location is given in relative coordinates system (see struct fira\_relative\_location).

#### FIRA\_DT\_LOCATION\_COORD\_INVALID

is a value in RSU range for test.

### 2.3.253 enum fira\_owr\_dtm\_timestamp\_type

enum fira\_owr\_dtm\_timestamp\_type

[Not supported in QM33 SDK] DTM TX Timestamp type.

### 2.3.253.1 Definition

```
enum fira_owr_dtm_timestamp_type {
    FIRA_OWR_DTM_TIMESTAMP_LOCAL_TIME_BASE,
    FIRA_OWR_DTM_TIMESTAMP_COMMON_TIME_BASE
};
```

### 2.3.253.2 Constants

#### FIRA\_OWR\_DTM\_TIMESTAMP\_LOCAL\_TIME\_BASE

timestamp in local time base.

#### FIRA\_OWR\_DTM\_TIMESTAMP\_COMMON\_TIME\_BASE

timestamp in common time base of the Initiator DT-Anchor.

### 2.3.254 enum fira\_owr\_dtm\_timestamp\_len

enum fira\_owr\_dtm\_timestamp\_len

[Not supported in QM33 SDK] DTM TX Timestamp length.

### 2.3.254.1 Definition

```
enum fira_owr_dtm_timestamp_len {
    FIRA_OWR_DTM_TIMESTAMP_40BITS,
    FIRA_OWR_DTM_TIMESTAMP_64BITS
};
```

### 2.3.254.2 Constants

**FIRA\_OWR\_DTM\_TIMESTAMP\_40BITS**  
40 bits timestamp.

**FIRA\_OWR\_DTM\_TIMESTAMP\_64BITS**  
64 bits timestamp.

### 2.3.255 enum fira\_owr\_utm\_timestamp\_len

enum **fira\_owr\_utm\_timestamp\_len**  
[Not supported in QM33 SDK] UTM TX Timestamp length.

### 2.3.255.1 Definition

```
enum fira_owr_utm_timestamp_len {
    FIRA_OWR_UTM_TIMESTAMP_NOT_PRESENT,
    FIRA_OWR_UTM_TIMESTAMP_40BITS,
    FIRA_OWR_UTM_TIMESTAMP_64BITS
};
```

### 2.3.255.2 Constants

**FIRA\_OWR\_UTM\_TIMESTAMP\_NOT\_PRESENT**  
0 bits timestamp.

**FIRA\_OWR\_UTM\_TIMESTAMP\_40BITS**  
40 bits timestamp.

**FIRA\_OWR\_UTM\_TIMESTAMP\_64BITS**  
64 bits timestamp.

### 2.3.256 enum fira\_ranging\_round\_control\_flags

enum **fira\_ranging\_round\_control\_flags**  
Ranging round control flags. Below bits make sense when SCHEDULE\_MODE is set to Time scheduled.

### 2.3.256.1 Definition

```
enum fira_ranging_round_control_flags {
    FIRA_RANGING_ROUND_CONTROL_RRRM_EXPECTED,
    FIRA_RANGING_ROUND_CONTROL_CM_EXPECTED,
    FIRA_RANGING_ROUND_CONTROL_RCP_EXCLUDED,
    FIRA_RANGING_ROUND_CONTROL_MEASUREMENT_REPORT_PHASE,
    FIRA_RANGING_ROUND_CONTROL_MEASUREMENT_REPORT_MESSAGE,
    FIRA_RANGING_ROUND_CONTROL_ALL
};
```

### 2.3.256.2 Constants

#### **FIRA\_RANGING\_ROUND\_CONTROL\_RRRM\_EXPECTED**

If set to 1, a Controller shall schedule an RRRM in the Ranging Device Management List (RDML). If set to 0, a Controller shall not schedule an RRRM in the RDML.

#### **FIRA\_RANGING\_ROUND\_CONTROL\_CM\_EXPECTED**

If set to 1, a Controller shall send a CM in-band and a Controlee shall expect a CM in-band. If set to 0, a Controller shall not send a CM in-band and a Controlee shall not expect a CM in-band.

#### **FIRA\_RANGING\_ROUND\_CONTROL\_RCP\_EXCLUDED**

If set to 1, RCP is excluded in Ranging Round, means CM is piggybacked with the RIM. If set to 0, RCP is included in Ranging Round.

#### **FIRA\_RANGING\_ROUND\_CONTROL\_MEASUREMENT\_REPORT\_PHASE**

UWBS shall ignore this bit.

#### **FIRA\_RANGING\_ROUND\_CONTROL\_MEASUREMENT\_REPORT\_MESSAGE**

If set to 1, the controller shall schedule the MRM to be sent from the responder(s) to the initiator in the RDML. If set to 0, the controller shall schedule the MRM to be sent from the initiator to the Responder(s) in the RDML.

#### **FIRA\_RANGING\_ROUND\_CONTROL\_ALL**

Maximum value of the parameter.

### 2.3.257 enum fira\_owr\_utm\_device\_id\_len

```
enum fira_owr_utm_device_id_len
```

[Not supported in QM33 SDK] UTM Device ID length.

### 2.3.257.1 Definition

```
enum fira_owr_utm_device_id_len {
    FIRA_OWR_UTM_DEVICE_ID_NOT_PRESENT,
    FIRA_OWR_UTM_DEVICE_ID_16BITS,
    FIRA_OWR_UTM_DEVICE_ID_32BITS,
    FIRA_OWR_UTM_DEVICE_ID_64BITS
};
```

### 2.3.257.2 Constants

#### **FIRA\_OWR\_UTM\_DEVICE\_ID\_NOT\_PRESENT**

0 bits device id.

#### **FIRA\_OWR\_UTM\_DEVICE\_ID\_16BITS**

16 bits device id.

#### **FIRA\_OWR\_UTM\_DEVICE\_ID\_32BITS**

32 bits device id.

#### **FIRA\_OWR\_UTM\_DEVICE\_ID\_64BITS**

64 bits device id.

### 2.3.258 enum fira\_multicast\_update\_status

enum **fira\_multicast\_update\_status**

controlee change status after update controller multicast list command.

#### 2.3.258.1 Definition

```
enum fira_multicast_update_status {
    FIRA_MULTICAST_UPDATE_STATUS_OK_MULTICAST_LIST_UPDATE,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_MULTICAST_LIST_FULL,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_KEY_FETCH_FAIL,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_SUB_SESSION_ID_NOT_FOUND,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_SUB_SESSION_KEY_NOT_FOUND,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_SUB_SESSION_KEY_NOT_APPLICABLE,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_SESSION_KEY_NOT_FOUND,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_ADDRESS_NOT_FOUND,
    FIRA_MULTICAST_UPDATE_STATUS_ERROR_ADDRESS_ALREADY_PRESENT
};
```

#### 2.3.258.2 Constants

##### **FIRA\_MULTICAST\_UPDATE\_STATUS\_OK\_MULTICAST\_LIST\_UPDATE**

it shall be reported if the multicast list is updated (Add/Delete) successfully for the given Controlee.

##### **FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_MULTICAST\_LIST\_FULL**

it shall be reported for a Controlee if the multicast is full.

##### **FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_KEY\_FETCH\_FAIL**

it shall be reported for a Controlee if Session Key fetch from Secure Component is failed.

##### **FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_SUB\_SESSION\_ID\_NOT\_FOUND**

it shall be reported for a Controlee if Sub-Session ID is not found in Secure Component.

##### **FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_SUB\_SESSION\_KEY\_NOT\_FOUND**

it shall be reported for a Controlee if Sub-Session Key is not found in Secure Component.

##### **FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_SUB\_SESSION\_KEY\_NOT\_APPLICABLE**

it shall be reported for a Controlee if Sub-Session Key is configured with STS config is other than 0x04 (Provisioned STS for Responder specific Sub-session Key).

#### FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_SESSION\_KEY\_NOT\_FOUND

it shall be reported for a Controlee if Sub-Session Key is configured but SESSION\_KEY App configuration parameter is not programmed.

#### FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_ADDRESS\_NOT\_FOUND

it shall be reported for a Controlee if its short address is not found while deleting its entry from multicast list.

#### FIRA\_MULTICAST\_UPDATE\_STATUS\_ERROR\_ADDRESS\_ALREADY\_PRESENT

it shall be reported for a Controlee if its short address is already present in the multicast list.

### 2.3.259 enum fira\_data\_segment\_info

enum **fira\_data\_segment\_info**

[Not supported in QM33 SDK] information about data packet received.

#### 2.3.259.1 Definition

```
enum fira_data_segment_info {
    FIRA_DATA_SEGMENT_FIRST,
    FIRA_DATA_SEGMENT_LAST,
    __FIRA_DATA_SEGMENT_AFTER_LAST
};
```

#### 2.3.259.2 Constants

##### FIRA\_DATA\_SEGMENT\_FIRST

packet is the first one in data message.

##### FIRA\_DATA\_SEGMENT\_LAST

packet is the last one in data message.

##### \_\_FIRA\_DATA\_SEGMENT\_AFTER\_LAST

Internal use.

### 2.3.260 enum fira\_owr\_aoa\_measurement\_ntf\_period

enum **fira\_owr\_aoa\_measurement\_ntf\_period**

[Not supported in QM33 SDK] period of sending SESSION\_INFO\_NTF.

#### 2.3.260.1 Definition

```
enum fira_owr_aoa_measurement_ntf_period {
    FIRA_OWR_AOA_MEASUREMENT_NTF_PERIOD_SINGLE,
    FIRA_OWR_AOA_MEASUREMENT_NTF_PERIOD_AGGREGATED
};
```

### 2.3.260.2 Constants

#### FIRA\_OWR\_AOA\_MEASUREMENT\_NTF\_PERIOD\_SINGLE

notification sent after every received frames.

#### FIRA\_OWR\_AOA\_MEASUREMENT\_NTF\_PERIOD\_AGGREGATED

notification sent after MIN\_FRAMES\_PER\_RR aggregated frames.

### 2.3.261 fira\_free\_frame\_report

void **fira\_free\_frame\_report**(struct frame\_report \*head)

Free the frame report created by diagnostic.

#### Parameters

- **head** (struct frame\_report\*) – Head of the frame report.

## 2.4 PCTT helper API

### 2.4.1 struct pctl\_parameters

struct **pctl\_parameters**

PCTT parameters.

#### 2.4.1.1 Definition

```
struct pctl_parameters {
    uint32_t num_packets;
    uint32_t t_gap;
    uint32_t t_start;
    uint32_t t_win;
    uint8_t randomize_psdu;
    uint8_t phr_ranging_bit;
    uint32_t rmarker_tx_start;
    uint32_t rmarker_rx_start;
    uint8_t sts_index_auto_incr;
    uint16_t rssi_outliers;
}
```

#### 2.4.1.2 Members

##### num\_packets

number of packets.

##### t\_gap

Gap between start of one packet to the next in micro seconds.

##### t\_start

Max. time from the start of T\_GAP to SFD found state in micro seconds.

##### t\_win

Max. time for which RX is looking for a packet from the start of T\_GAP in micro seconds.



### randomize\_psdu

0 no randomization, 1 take first byte of data supplied by cmd is the seed.

### phr\_ranging\_bit

0 disable, 1 enable. Configures ranging bit field of PHR.

### rmarker\_tx\_start

Start time of TX in  $1/(128 \times 499.2\text{MHz})$  units.

### rmarker\_rx\_start

Start time of RX in  $1/(128 \times 499.2\text{MHz})$  units.

### sts\_index\_auto\_incr

0x00: STS\_INDEX config value is used for all PER Rx/ Periodic TX. 0x01: STS\_INDEX value SHALL be incremented for every frame in PER Rx/Periodic TX test.

### rss\_i\_outliers

number of outliers to remove from Rssi values.

## 2.4.2 struct ptt\_test\_payload

### struct ptt\_test\_payload

PCTT test payload.

#### 2.4.2.1 Definition

```
struct ptt_test_payload {
    uint8_t payload[PCTT_PAYLOAD_MAX_LEN];
    int payload_len;
}
```

#### 2.4.2.2 Members

### payload

PSDU data bytes used by certain cmd tests.

### payload\_len

payload length.

## 2.4.3 struct ptt\_session\_parameters

### struct ptt\_session\_parameters

PCTT session parameters.

### 2.4.3.1 Definition

```
struct ptt_session_parameters {
    uint8_t device_role;
    uint16_t short_addr;
    uint16_t destination_short_address;
    uint8_t tx_antenna_selection;
    uint8_t rx_antenna_selection;
    uint32_t slot_duration_rstu;
    uint8_t channel_number;
    uint8_t preamble_code_index;
    uint8_t rframe_config;
    uint8_t prf_mode;
    uint8_t preamble_duration;
    uint8_t sfd_id;
    uint8_t number_of_sts_segments;
    uint8_t psdu_data_rate;
    uint8_t phr_data_rate;
    uint8_t mac_fcs_type;
    uint32_t sts_index;
    uint8_t sts_length;
}
```

### 2.4.3.2 Members

#### device\_role

see enum device\_role.

#### short\_addr

Device short address

#### destination\_short\_address

Address of controller.

#### tx\_antenna\_selection

Selection of TX antenna configuration for this session.

#### rx\_antenna\_selection

Selection of RX antenna configuration for this session.

#### slot\_duration\_rstu

Duration of a slot in RSTU. (1200RSTU=1ms)

#### channel\_number

Uwb channel for this session.

#### preamble\_code\_index

Uwb preamble code index. BPRF (9-24), HPRF (25-32) *[Not supported in QM33 SDK]*

#### rframe\_config

see enum fira\_rframe\_config.

#### prf\_mode

prf\_mode 0x00 = 62.4 MHz PRF. BPRF mode (default) 0x01 = 124.8 MHz PRF. HPRF mode. *[Not supported in QM33 SDK]* 0x02 – 249.6 MHz PRF. HPRF mode with data rate 27.2 and 31.2 Mbps *[Not supported in QM33 SDK]* Values 0x03 to 0xFF = RFU

### preamble\_duration

preamble\_duration 0x00 = 32 symbols *[Not supported in QM33 SDK]* 0x01 = 64 symbols (default) Values 0x02 to 0xFF = RFU

### sfd\_id

BPRF (0 or 2), HPRF (1-4) *[Not supported in QM33 SDK]*.

### number\_of\_sts\_segments

number\_of\_sts\_segments 0x01 = 1 STS Segment (default) 0x02 = 2 STS Segments (HPRF only) *[Not supported in QM33 SDK]* 0x03 = 3 STS Segments (HPRF only) *[Not supported in QM33 SDK]* 0x04 = 4 STS Segments (HPRF only) *[Not supported in QM33 SDK]* Values 0x05 to 0xFF = RFU

### psdu\_data\_rate

psdu\_data\_rate 0x00 - > 6.81Mbps (Default) 0x01 = 7.80 Mbps *[Not supported in QM33 SDK]* 0x02 = 27.2 Mbps *[Not supported in QM33 SDK]* 0x03 = 31.2 Mbps *[Not supported in QM33 SDK]* 0x04 = 850 Kbps *[Not supported in QM33 SDK]* Values 0x00, 0x02, 0x04 map to K=3 and 0x01, 0x03 map to K=7. Values 0x05 to 0xFF = RFU

### phr\_data\_rate

BPRF PHR data rate 0x00 = 850kb/s rate. 0x01 = 6.8Mb/s rate.

### mac\_fcs\_type

mac\_fcs\_type 0x00 = CRC 16 (default) 0x01 = CRC 32 Values 0x02 to 0xFF = RFU

### sts\_index

sts\_index. default = 0.

### sts\_length

Number of symbols in a STS segment.

Possible values:

- 0x00: 32 symbols
- 0x01: 64 symbols (default)
- 0x02: 128 symbols
- Values 0x03 to 0xFF: RFU

## 2.4.4 struct pttt\_result\_data

struct pttt\_result\_data

PCTT result data.

### 2.4.4.1 Definition

```
struct pttt_result_data {
    uint8_t status;
    uint32_t attempts;
    uint32_t acq_detect;
    uint32_t acq_reject;
    uint32_t rx_fail;
    uint32_t sync_cir_ready;
    uint32_t sfd_fail;
    uint32_t sfd_found;
    uint32_t phr_dec_error;
    uint32_t phr_bit_error;
```

(continues on next page)

(continued from previous page)

```
uint32_t psdu_dec_error;
uint32_t psdu_bit_error;
uint32_t sts_found;
uint32_t eof;
uint32_t rx_done_ts_int;
uint16_t rx_done_ts_frac;
uint8_t toa_gap;
uint8_t phr;
uint8_t psdu_data[PCTT_PAYLOAD_MAX_LEN];
uint16_t psdu_data_len;
uint32_t tx_ts_int;
uint16_t tx_ts_frac;
uint32_t rx_ts_int;
uint16_t rx_ts_frac;
int16_t noise_value;
uint32_t measurement;
int16_t pdoa_azimuth_2pi;
int16_t pdoa_elevation_2pi;
uint8_t nb_rssi;
uint16_t rssid_q8[MAX_RSSI];
int16_t aoa_azimuth_2pi;
int16_t aoa_elevation_2pi;
int32_t cfo_q26;
}
```

## 2.4.4.2 Members

### status

Generic status code.

### attempts

Number of RX attempts.

### acq\_detect

Number of times signal was detected.

### acq\_reject

Number of times signal was rejected.

### rx\_fail

Number of times RX did not go beyond ACQ stage.

### sync\_cir\_ready

Number of times sync. CIR ready event was received.

### sfd\_fail

Number of times RX was stuck at either ACQ detect or sync CIR ready.

### sfd\_found

Number of times SFD was found.

### phr\_dec\_error

No. of times PHR decode failed.

### phr\_bit\_error

No. of times PHR bits in error.

**psdu\_dec\_error**

No. of times payload decode failed.

**psdu\_bit\_error**

No. of times payload bits in error.

**sts\_found**

No. of times STS detection was successful.

**eof**

No. of times end of frame event was triggered

**rx\_done\_ts\_int**

Integer part of timestamp 1/124.8Mhz ticks.

**rx\_done\_ts\_frac**

Fractional part of timestamp 1/(128\*499.2Mhz) ticks.

**toa\_gap**

ToA of main path minus ToA of first path in nanoseconds.

**phr**

Received PHR (bits 0-12 as per IEEE spec).

**psdu\_data**

Length of PSDU Data(N) to follow.

**psdu\_data\_len**

Length of psdu\_data.

**tx\_ts\_int**

Integer part of timestamp in 1/124.8 us resolution.

**tx\_ts\_frac**

Fractional part of timestamp in 1/124.8/512 us resolution.

**rx\_ts\_int**

Integer part of timestamp in 1/124.8 us resolution.

**rx\_ts\_frac**

Fractional part of timestamp in 1/124.8/512 us resolution.

**noise\_value**

General noise value in dB, allowing to compute SNR.

**measurement**

For TEST\_SS\_TWR\_NTF. Contains Tround time of Initiator or Treply time of Responder depending on DE-VICE\_ROLE option. This is expressed in 1/(128 \* 499.2Mhz) ticks.

**pdoa\_azimuth\_2pi**

Estimation of reception phase difference in the azimuth.

**pdoa\_elevation\_2pi**

Estimation of reception phase difference in the elevation.

**nb\_rssi**

The number of RSSIs in the array *rssis\_q8*.

**rssis\_q8**

Calculated RSSIs (Received Signal Strength Indicator), encoded as Q8.8.

**aoa\_azimuth\_2pi**

Estimation of reception angle in the azimuth.

**aoa\_elevation\_2pi**

Estimation of reception angle in the elevation.

## cfo\_q26

Carrier Frequency Offset, encoded as signed q8.26.

### 2.4.5 typedef ptt\_helper\_notification\_cb\_t

void ptt\_helper\_notification\_cb\_t(void \*user\_data, uint8\_t cmd\_id, const struct [ptt\\_result\\_data](#) \*results)  
Notification callback type.

#### Parameters

- **user\_data** (void\*) – User data pointer given to ptt\_helper\_open.
- **cmd\_id** (uint8\_t) – The type of cmd corresponding to the results.
- **results** (const struct [ptt\\_result\\_data](#)\*) – ptt results.

#### 2.4.5.1 Return

nothing

### 2.4.6 ptt\_helper\_open

enum qerr ptt\_helper\_open(struct ptt\_context \*context, struct [uwbmac\\_context](#) \*uwbmac\_context,  
[ptt\\_helper\\_notification\\_cb\\_t](#) notification\_cb, const char \*scheduler, int region\_id,  
void \*user\_data)

Initialize the internal resources of the helper.

#### Parameters

- **context** (struct ptt\_context\*) – Context to initialize.
- **uwbmac\_context** (struct [uwbmac\\_context](#)\*) – UWB MAC context.
- **notification\_cb** ([ptt\\_helper\\_notification\\_cb\\_t](#)) – Callback to call when a notification is available.
- **scheduler** (const char\*) – In which scheduler the region will be
- **region\_id** (int) – Which id the region will have in the scheduler
- **user\_data** (void\*) – User data pointer to give back in callbacks.

#### 2.4.6.1 NOTE

This function must be called first. ptt\_helper\_close must be called at the end of the application to ensure resources are freed.

### 2.4.6.2 Return

QERR\_SUCCESS or error.

### 2.4.7 pctl\_helper\_set\_scheduler

enum qerr **pctl\_helper\_set\_scheduler**(struct pctl\_context \*context)

Set the scheduler and the region.

#### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.

#### 2.4.7.1 NOTE

This function must be called while the UWB MAC is stopped.

### 2.4.7.2 Return

QERR\_SUCCESS or error.

### 2.4.8 pctl\_helper\_close

void **pctl\_helper\_close**(struct pctl\_context \*context)

Free all internal resources of the helper.

#### Parameters

- **context** (struct pctl\_context\*) – Context to release.

### 2.4.9 pctl\_helper\_session\_init

int **pctl\_helper\_session\_init**(struct pctl\_context \*context)

Initialize a pctl session.

#### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.

#### 2.4.9.1 Description

This function must be called first to create and initialize the pctl session.

### 2.4.9.2 Return

0 or error.

### 2.4.10 pctl\_helper\_session\_start

```
int pctl_helper_session_start(struct pctl_context *context, uint8_t cmd_id, const struct pctl\_parameters
                             *params)
```

Start a pctl session.

#### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.
- **cmd\_id** (uint8\_t) – The cmd being executed.
- **params** (const struct [pctl\\_parameters](#)\*) – specific pctl\_parameters of the test.

#### 2.4.10.1 Description

This function must be called after pctl session was initialized.

#### 2.4.10.2 Return

0 or error.

### 2.4.11 pctl\_helper\_set\_test\_payload

```
int pctl_helper_set_test_payload(struct pctl_context *context, const struct pctl\_test\_payload *test_payload)
```

Set payload.

#### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.
- **test\_payload** (const struct [pctl\\_test\\_payload](#)\*) – data payload for the test.

#### 2.4.11.1 Description

This function must be called after pctl session was initialized.

#### 2.4.11.2 Return

0 or error.



## 2.4.12 pctl\_helper\_session\_deinit

```
int pctl_helper_session_deinit(struct pctl_context *context)
```

Deinitialize a pctl session.

### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.

### 2.4.12.1 Description

This function is called to free all memory allocated by the session. This function must be called when the session is stopped.

### 2.4.12.2 Return

0 or error.

## 2.4.13 pctl\_helper\_session\_get\_state

```
int pctl_helper_session_get_state(struct pctl_context *context, int *state)
```

Request to get a state.

### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.
- **state** (int\*) – The result state of the session.

### 2.4.13.1 Return

0 or error.

## 2.4.14 pctl\_helper\_session\_get\_params

```
int pctl_helper_session_get_params(struct pctl_context *context, struct pctl\_session\_parameters *params)
```

Request pctl parameters.

### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.
- **params** (struct [pctl\\_session\\_parameters](#)\*) – the output parameters.

#### 2.4.14.1 Return

0 or error.

#### 2.4.15 pctl\_helper\_session\_set\_params

int pctl\_helper\_session\_set\_params(struct pctl\_context \*context, struct [pctl\\_session\\_parameters](#) \*params)

Set pctl session parameters.

##### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.
- **params** (struct [pctl\\_session\\_parameters](#)\*) – the session parameters to set.

#### 2.4.15.1 Return

0 or error.

#### 2.4.16 pctl\_helper\_tx\_cw

enum qerr pctl\_helper\_tx\_cw(struct pctl\_context \*context, struct [pctl\\_session\\_parameters](#) \*params, bool start)

Start/stop continuous wave test.

##### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.
- **params** (struct [pctl\\_session\\_parameters](#)\*) – Specific pctl\_parameters of the test, used to set channel\_number and TX antenna to transmit from.
- **start** (bool) – Whether to start or not.

#### 2.4.16.1 Return

QERR\_SUCCESS or error.

#### 2.4.17 pctl\_helper\_pll\_lock

enum qerr pctl\_helper\_pll\_lock(struct pctl\_context \*context, struct [pctl\\_session\\_parameters](#) \*params, uint8\_t \*errcode, uint32\_t \*pll\_status)

Run PLL Lock test.

##### Parameters

- **context** (struct pctl\_context\*) – Context of this helper.
- **params** (struct [pctl\\_session\\_parameters](#)\*) – Specific pctl\_parameters, used to set channel\_number.
- **errcode** (uint8\_t\*) – Returned error code.
- **pll\_status** (uint32\_t\*) – Returned PLL status in case of test success.

### 2.4.17.1 Return

QERR\_SUCCESS or error.

### 2.4.18 enum pctl\_device\_role

enum pctl\_device\_role

[NOT IMPLEMENTED] Role played by a device.

#### 2.4.18.1 Definition

```
enum pctl_device_role {
    PCTL_DEVICE_ROLE_RESPONDER,
    PCTL_DEVICE_ROLE_INITIATOR
};
```

#### 2.4.18.2 Constants

##### PCTL\_DEVICE\_ROLE\_RESPONDER

The device acts as a responder.

##### PCTL\_DEVICE\_ROLE\_INITIATOR

The device acts as an initiator.

#### 2.4.18.3 Description

Current implementation does not support decorrelation between the device's role and the device's type. The controller is always the initiator and the controlee is always the responder.

This enum is not used in the current implementation.

### 2.4.19 enum pctl\_rframe\_config

enum pctl\_rframe\_config

Rframe configuration used to transmit/receive ranging messages.

#### 2.4.19.1 Definition

```
enum pctl_rframe_config {
    PCTL_RFRAME_CONFIG_SP0,
    PCTL_RFRAME_CONFIG_SP1,
    PCTL_RFRAME_CONFIG_SP2,
    PCTL_RFRAME_CONFIG_SP3
};
```

### 2.4.19.2 Constants

**PCTT\_RFRAME\_CONFIG\_SP0**  
Use SP0 mode.

**PCTT\_RFRAME\_CONFIG\_SP1**  
Use SP1 mode.

**PCTT\_RFRAME\_CONFIG\_SP2**  
RFU

**PCTT\_RFRAME\_CONFIG\_SP3**  
Use SP3 mode.

### 2.4.20 enum pctl\_prf\_mode

enum **pctl\_prf\_mode**  
Pulse Repetition Frequency mode.

#### 2.4.20.1 Definition

```
enum pctl_prf_mode {
    PCTT_PRF_MODE_BPRF,
    PCTT_PRF_MODE_HPRF,
    PCTT_PRF_MODE_HPRF_HIGH_RATE
};
```

#### 2.4.20.2 Constants

**PCTT\_PRF\_MODE\_BPRF**  
Base Pulse Repetition Frequency.

**PCTT\_PRF\_MODE\_HPRF**  
*[Not supported in QM33 SDK]* Higher Pulse Repetition Frequency.

**PCTT\_PRF\_MODE\_HPRF\_HIGH\_RATE**  
*[Not supported in QM33 SDK]* Higher Pulse Repetition Frequency allowing higher data rates (27M2 and 31M2).

#### 2.4.20.3 Description

This enum is not used in the current implementation.

### 2.4.21 enum pctl\_preamble\_duration

enum **pctl\_preamble\_duration**  
Duration of preamble in symbols.

### 2.4.21.1 Definition

```
enum ptt_preamble_duration {
    PCTT_PREAMBLE_DURATION_32,
    PCTT_PREAMBLE_DURATION_64
};
```

### 2.4.21.2 Constants

#### PCTT\_PREAMBLE\_DURATION\_32

[Not supported in QM33 SDK] 32 symbols duration.

#### PCTT\_PREAMBLE\_DURATION\_64

64 symbols duration.

### 2.4.22 enum ptt\_sfd\_id

enum ptt\_sfd\_id  
Start-of-frame delimiter.

### 2.4.22.1 Definition

```
enum ptt_sfd_id {
    PCTT_SFD_ID_0,
    PCTT_SFD_ID_1,
    PCTT_SFD_ID_2,
    PCTT_SFD_ID_3,
    PCTT_SFD_ID_4
};
```

### 2.4.22.2 Constants

#### PCTT\_SFD\_ID\_0

Delimiter is [0 +1 0 -1 +1 0 0 -1]

#### PCTT\_SFD\_ID\_1

[Not supported in QM33 SDK] Delimiter is [-1 -1 +1 -1 ]

#### PCTT\_SFD\_ID\_2

Delimiter is [-1 -1 -1 +1 -1 -1 +1 -1 ]

#### PCTT\_SFD\_ID\_3

[Not supported in QM33 SDK] Delimiter is [-1 -1 -1 -1 -1 +1 +1 -1 -1 +1 -1 +1 -1 -1 +1 -1 ]

#### PCTT\_SFD\_ID\_4

[Not supported in QM33 SDK] Delimiter is [-1 -1 -1 -1 -1 -1 -1 +1 -1 -1 +1 -1 -1 +1 -1 +1 -1 -1 -1 -1 +1 +1 -1 -1 +1 -1 +1 +1 -1 -1 ]

### 2.4.23 enum pctl\_number\_of\_sts\_segments

enum **pctl\_number\_of\_sts\_segments**  
Number of STS segments.

#### 2.4.23.1 Definition

```
enum pctl_number_of_sts_segments {
    PCTL_NUMBER_OF_STS_SEGMENTS_NONE,
    PCTL_NUMBER_OF_STS_SEGMENTS_1_SEGMENT,
    PCTL_NUMBER_OF_STS_SEGMENTS_2_SEGMENTS,
    PCTL_NUMBER_OF_STS_SEGMENTS_3_SEGMENTS,
    PCTL_NUMBER_OF_STS_SEGMENTS_4_SEGMENTS
};
```

#### 2.4.23.2 Constants

**PCTL\_NUMBER\_OF\_STS\_SEGMENTS\_NONE**  
No STS Segment (Rframe config SP0).

**PCTL\_NUMBER\_OF\_STS\_SEGMENTS\_1\_SEGMENT**  
1 STS Segment.

**PCTL\_NUMBER\_OF\_STS\_SEGMENTS\_2\_SEGMENTS**  
*[Not supported in QM33 SDK]* 2 STS Segments.

**PCTL\_NUMBER\_OF\_STS\_SEGMENTS\_3\_SEGMENTS**  
*[Not supported in QM33 SDK]* 3 STS Segments.

**PCTL\_NUMBER\_OF\_STS\_SEGMENTS\_4\_SEGMENTS**  
*[Not supported in QM33 SDK]* 4 STS Segments.

### 2.4.24 enum pctl\_psdu\_data\_rate

enum **pctl\_psdu\_data\_rate**  
Data rate used to exchange PSDUs.

#### 2.4.24.1 Definition

```
enum pctl_psdu_data_rate {
    PCTL_PSDU_DATA_RATE_6M81,
    PCTL_PSDU_DATA_RATE_7M80,
    PCTL_PSDU_DATA_RATE_27M2,
    PCTL_PSDU_DATA_RATE_31M2
};
```

#### 2.4.24.2 Constants

##### PCTT\_PSDU\_DATA\_RATE\_6M81

6.8Mb/s rate.

##### PCTT\_PSDU\_DATA\_RATE\_7M80

[Not supported in QM33 SDK] 7.8Mb/s rate.

##### PCTT\_PSDU\_DATA\_RATE\_27M2

[Not supported in QM33 SDK] 27.2Mb/s rate.

##### PCTT\_PSDU\_DATA\_RATE\_31M2

[Not supported in QM33 SDK] 31.2Mb/s rate.

#### 2.4.25 enum pctl\_phr\_data\_rate

##### enum pctl\_phr\_data\_rate

Data rate used to exchange PHR.

##### 2.4.25.1 Definition

```
enum pctl_phr_data_rate {
    PCTT_PHR_DATA_RATE_850K,
    PCTT_PHR_DATA_RATE_6M81
};
```

#### 2.4.25.2 Constants

##### PCTT\_PHR\_DATA\_RATE\_850K

850kb/s rate.

##### PCTT\_PHR\_DATA\_RATE\_6M81

6.8Mb/s rate.

#### 2.4.25.3 Description

This enum is not used in the current implementation.

#### 2.4.26 enum pctl\_status\_ranging

##### enum pctl\_status\_ranging

Ranging status: success or failure reason.

### 2.4.26.1 Definition

```
enum pctl_status_ranging {
    PCTL_STATUS_RANGING_INTERNAL_ERROR,
    PCTL_STATUS_RANGING_SUCCESS,
    PCTL_STATUS_RANGING_TX_FAILED,
    PCTL_STATUS_RANGING_RX_TIMEOUT,
    PCTL_STATUS_RANGING_RX_PHY_DEC_FAILED,
    PCTL_STATUS_RANGING_RX_PHY_TOA_FAILED,
    PCTL_STATUS_RANGING_RX_PHY_STS_FAILED,
    PCTL_STATUS_RANGING_RX_MAC_DEC_FAILED,
    PCTL_STATUS_RANGING_RX_MAC_IE_DEC_FAILED,
    PCTL_STATUS_RANGING_RX_MAC_IE_MISSING
};
```

### 2.4.26.2 Constants

#### **PCTL\_STATUS\_RANGING\_INTERNAL\_ERROR**

Implementation specific error.

#### **PCTL\_STATUS\_RANGING\_SUCCESS**

Ranging info are valid.

#### **PCTL\_STATUS\_RANGING\_TX\_FAILED**

Failed to transmit UWB packet.

#### **PCTL\_STATUS\_RANGING\_RX\_TIMEOUT**

No UWB packet detected by the receiver.

#### **PCTL\_STATUS\_RANGING\_RX\_PHY\_DEC\_FAILED**

UWB packet channel decoding error.

#### **PCTL\_STATUS\_RANGING\_RX\_PHY\_TOA\_FAILED**

Failed to detect time of arrival of the UWB packet from CIR samples.

#### **PCTL\_STATUS\_RANGING\_RX\_PHY\_STS\_FAILED**

UWB packet STS segment mismatch.

#### **PCTL\_STATUS\_RANGING\_RX\_MAC\_DEC\_FAILED**

MAC CRC or syntax error.

#### **PCTL\_STATUS\_RANGING\_RX\_MAC\_IE\_DEC\_FAILED**

IE syntax error.

#### **PCTL\_STATUS\_RANGING\_RX\_MAC\_IE\_MISSING**

Expected IE missing in the packet.



## 2.4.27 enum pctl\_session\_state

enum **pctl\_session\_state**  
Session state.

### 2.4.27.1 Definition

```
enum pctl_session_state {
    PCTL_SESSION_STATE_INIT,
    PCTL_SESSION_STATE_DEINIT,
    PCTL_SESSION_STATE_ACTIVE,
    PCTL_SESSION_STATE_IDLE
};
```

### 2.4.27.2 Constants

**PCTL\_SESSION\_STATE\_INIT**  
Initial state, session is not ready yet.

**PCTL\_SESSION\_STATE\_DEINIT**  
Session does not exist.

**PCTL\_SESSION\_STATE\_ACTIVE**  
Session is currently active.

**PCTL\_SESSION\_STATE\_IDLE**  
Session is ready to start, but not currently active.

## 2.4.28 enum pctl\_sts\_length

enum **pctl\_sts\_length**  
Number of symbols in a STS segment.

### 2.4.28.1 Definition

```
enum pctl_sts_length {
    PCTL_STS_LENGTH_32,
    PCTL_STS_LENGTH_64,
    PCTL_STS_LENGTH_128
};
```

2.4.28.2 Constants

**PCTT\_STS\_LENGTH\_32**  
The STS length is 32 symbols.

**PCTT\_STS\_LENGTH\_64**  
The STS length is 64 symbols.

**PCTT\_STS\_LENGTH\_128**  
The STS length is 128 symbols.

## Index

### A

aoa\_measurements (C struct), 78  
 aoa\_measurements\_index (C enum), 77

### C

controlee\_parameters (C struct), 73  
 controlee\_status (C struct), 85  
 controlees\_parameters (C struct), 74

### D

data\_credit\_ntf\_content (C struct), 86  
 data\_message\_content (C struct), 88  
 data\_transfer\_status\_ntf\_content (C struct), 87  
 dt\_anchor\_ranging\_round\_config (C struct), 74  
 dt\_tag\_ranging\_rounds\_config (C struct), 76  
 dt\_tag\_round\_indexes\_rsp (C struct), 76

### F

fira\_data\_message\_status (C enum), 178  
 fira\_data\_segment\_info (C enum), 183  
 fira\_data\_transfer\_status (C enum), 170  
 fira\_dl\_tdoa\_measurements (C struct), 83  
 fira\_dl\_tdoa\_ranging\_method (C enum), 162  
 fira\_dl\_tdoa\_ranging\_results (C struct), 85  
 fira\_dt\_anchor\_acting\_role (C enum), 160  
 fira\_dt\_location\_coord\_system\_type (C enum), 178  
 fira\_free\_frame\_report (C function), 184  
 fira\_helper\_add\_controlee (C function), 97  
 fira\_helper\_bool\_to\_ranging\_round\_control (C function), 108  
 fira\_helper\_bool\_to\_result\_report\_config (C function), 120  
 fira\_helper\_cb\_type (C enum), 91  
 fira\_helper\_close (C function), 93  
 fira\_helper\_data\_message\_send (C function), 98  
 fira\_helper\_deinit\_session (C function), 95  
 fira\_helper\_delete\_controlee (C function), 97  
 fira\_helper\_dt\_tag\_configure\_ranging\_rounds (C function), 153  
 fira\_helper\_get\_capabilities (C function), 94  
 fira\_helper\_get\_controlees (C function), 98  
 fira\_helper\_get\_controlees\_count (C function), 98  
 fira\_helper\_get\_ranging\_count (C function), 97  
 fira\_helper\_get\_session\_application\_data\_endpoint (C function), 160  
 fira\_helper\_get\_session\_block\_duration\_ms (C function), 131  
 fira\_helper\_get\_session\_block\_stride\_length (C function), 132  
 fira\_helper\_get\_session\_cap\_size\_max (C function), 143

fira\_helper\_get\_session\_cap\_size\_min (C function), 143  
 fira\_helper\_get\_session\_channel\_number (C function), 135  
 fira\_helper\_get\_session\_data\_repetition\_count (C function), 140  
 fira\_helper\_get\_session\_data\_transfer\_status\_ntf\_config (C function), 141  
 fira\_helper\_get\_session\_data\_vendor\_oui (C function), 140  
 fira\_helper\_get\_session\_destination\_short\_addresses (C function), 130  
 fira\_helper\_get\_session\_device\_role (C function), 129  
 fira\_helper\_get\_session\_device\_type (C function), 128  
 fira\_helper\_get\_session\_diags\_frame\_reports\_fields (C function), 144  
 fira\_helper\_get\_session\_dl\_tdoa\_active\_ranging\_rounds (C function), 107  
 fira\_helper\_get\_session\_dl\_tdoa\_anchor\_cfo (C function), 106  
 fira\_helper\_get\_session\_dl\_tdoa\_anchor\_location (C function), 107  
 fira\_helper\_get\_session\_dl\_tdoa\_anchor\_location\_presence (C function), 106  
 fira\_helper\_get\_session\_dl\_tdoa\_anchor\_location\_type (C function), 106  
 fira\_helper\_get\_session\_dl\_tdoa\_block\_skipping (C function), 107  
 fira\_helper\_get\_session\_dl\_tdoa\_hop\_count (C function), 105  
 fira\_helper\_get\_session\_dl\_tdoa\_ranging\_method (C function), 104  
 fira\_helper\_get\_session\_dl\_tdoa\_responder\_tof (C function), 104  
 fira\_helper\_get\_session\_dl\_tdoa\_time\_reference\_anchor (C function), 104  
 fira\_helper\_get\_session\_dl\_tdoa\_tx\_timestamp\_len (C function), 105  
 fira\_helper\_get\_session\_dl\_tdoa\_tx\_timestamp\_type (C function), 105  
 fira\_helper\_get\_session\_enable\_diagnostics (C function), 143  
 fira\_helper\_get\_session\_far\_proximity\_config\_cm (C function), 147  
 fira\_helper\_get\_session\_in\_band\_termination\_attempt\_count (C function), 103  
 fira\_helper\_get\_session\_inter\_frame\_interval\_ms (C function), 145  
 fira\_helper\_get\_session\_key\_rotation (C function), 138

fira_helper_get_session_key_rotation_rate (C function), 138	fira_helper_get_session_key_rotation_rate (C function), 131
fira_helper_get_session_link_layer_mode (C function), 140	fira_helper_get_session_round_hopping (C function), 132
fira_helper_get_session_lower_aoa_bound_config_azimuth_2p (C function), 148	fira_helper_get_session_schedule_mode (C function), 133
fira_helper_get_session_lower_aoa_bound_config_elevation_2p (C function), 150	fira_helper_get_session_session_info_ntf_config (C function), 146
fira_helper_get_session_mac_address_mode (C function), 133	fira_helper_get_session_sfd_id (C function), 136
fira_helper_get_session_mac_fcs_type (C function), 141	fira_helper_get_session_short_address (C function), 130
fira_helper_get_session_mac_payload_encryption (C function), 139	fira_helper_get_session_slot_duration_rstu (C function), 131
fira_helper_get_session_max_number_of_measurements (C function), 134	fira_helper_get_session_static_sts_iv (C function), 137
fira_helper_get_session_max_rr_retry (C function), 134	fira_helper_get_session_sts_config (C function), 129
fira_helper_get_session_measurement_sequence (C function), 144	fira_helper_get_session_sts_length (C function), 144
fira_helper_get_session_min_frames_per_rr (C function), 145	fira_helper_get_session_sub_session_id (C function), 137
fira_helper_get_session_mtu_size (C function), 145	fira_helper_get_session_time0_ns (C function), 130
fira_helper_get_session_multi_node_mode (C function), 129	fira_helper_get_session_time_base (C function), 141
fira_helper_get_session_near_proximity_config_cm (C function), 147	fira_helper_get_session_upper_aoa_bound_config_azimuth_2p (C function), 149
fira_helper_get_session_number_of_sts_segments (C function), 142	fira_helper_get_session_upper_aoa_bound_config_elevation_2p (C function), 151
fira_helper_get_session_owr_aoa_measurement_ntf_period (C function), 146	fira_helper_get_session_ut_device_id (C function), 157
fira_helper_get_session_parameters (C function), 96	fira_helper_get_session_ut_device_id_len (C function), 157
fira_helper_get_session_phr_data_rate (C function), 142	fira_helper_get_session_ut_random_window (C function), 156
fira_helper_get_session_preamble_code_index (C function), 135	fira_helper_get_session_ut_report_config_count (C function), 158
fira_helper_get_session_preamble_duration (C function), 136	fira_helper_get_session_ut_report_config_event (C function), 158
fira_helper_get_session_prf_mode (C function), 142	fira_helper_get_session_ut_report_config_interval (C function), 157
fira_helper_get_session_priority (C function), 132	fira_helper_get_session_ut_tx_interval_ms (C function), 156
fira_helper_get_session_psdu_data_rate (C function), 136	fira_helper_get_session_ut_tx_timestamp_len (C function), 156
fira_helper_get_session_ranging_round_control (C function), 133	fira_helper_get_session_vendor_id (C function), 137
fira_helper_get_session_ranging_round_usage (C function), 128	fira_helper_get_session_vupper64 (C function), 138
fira_helper_get_session_report_psdus (C function), 108	fira_helper_init_session (C function), 94
fira_helper_get_session_report_rssi (C function), 139	fira_helper_notification_cb_t (C function), 92
fira_helper_get_session_result_report_config (C function), 139	fira_helper_open (C function), 92
fira_helper_get_session_rframe_config (C function), 135	fira_helper_session_get_count (C function), 96
fira_helper_get_session_round_duration_slots (C function), 131	fira_helper_session_get_data_size_in_ranging (C function), 158
	fira_helper_session_get_state (C function), 96
	fira_helper_set_device_status_cb (C function), 93
	fira_helper_set_hus_controlee_config (C function), 159
	fira_helper_set_hus_controller_config (C function), 159

tion), 159  
 fira\_helper\_set\_scheduler (C function), 93  
 fira\_helper\_set\_session\_application\_data\_endpoint  
 (C function), 159  
 fira\_helper\_set\_session\_block\_duration\_ms (C  
 function), 112  
 fira\_helper\_set\_session\_block\_stride\_length (C  
 function), 112  
 fira\_helper\_set\_session\_cap\_size\_max (C function),  
 124  
 fira\_helper\_set\_session\_cap\_size\_min (C function),  
 123  
 fira\_helper\_set\_session\_channel\_number (C func-  
 tion), 115  
 fira\_helper\_set\_session\_data\_repetition\_count  
 (C function), 121  
 fira\_helper\_set\_session\_data\_transfer\_status\_ntf\_config  
 (C function), 122  
 fira\_helper\_set\_session\_destination\_short\_address  
 (C function), 110  
 fira\_helper\_set\_session\_device\_role (C function),  
 109  
 fira\_helper\_set\_session\_device\_type (C function),  
 99  
 fira\_helper\_set\_session\_diags\_frame\_reports\_fields  
 (C function), 125  
 fira\_helper\_set\_session\_dl\_tdoa\_active\_ranging\_rounds  
 (C function), 102  
 fira\_helper\_set\_session\_dl\_tdoa\_anchor\_cfo (C  
 function), 101  
 fira\_helper\_set\_session\_dl\_tdoa\_anchor\_location  
 (C function), 102  
 fira\_helper\_set\_session\_dl\_tdoa\_anchor\_location\_presence  
 (C function), 101  
 fira\_helper\_set\_session\_dl\_tdoa\_anchor\_location\_type  
 (C function), 102  
 fira\_helper\_set\_session\_dl\_tdoa\_block\_skipping  
 (C function), 103  
 fira\_helper\_set\_session\_dl\_tdoa\_hop\_count (C  
 function), 101  
 fira\_helper\_set\_session\_dl\_tdoa\_ranging\_method  
 (C function), 100  
 fira\_helper\_set\_session\_dl\_tdoa\_responder\_tof  
 (C function), 99  
 fira\_helper\_set\_session\_dl\_tdoa\_time\_reference\_and\_time  
 (C function), 99  
 fira\_helper\_set\_session\_dl\_tdoa\_tx\_timestamp\_len  
 (C function), 100  
 fira\_helper\_set\_session\_dl\_tdoa\_tx\_timestamp\_type  
 (C function), 100  
 fira\_helper\_set\_session\_enable\_diagnostics (C  
 function), 124  
 fira\_helper\_set\_session\_far\_proximity\_config\_cm  
 (C function), 128  
 fira\_helper\_set\_session\_in\_band\_termination\_attempt\_count  
 (C function), 152  
 fira\_helper\_set\_session\_inter\_frame\_interval\_ms  
 (C function), 126  
 fira\_helper\_set\_session\_key (C function), 151  
 fira\_helper\_set\_session\_key\_rotation (C function),  
 119  
 fira\_helper\_set\_session\_key\_rotation\_rate (C  
 function), 119  
 fira\_helper\_set\_session\_link\_layer\_mode (C func-  
 tion), 121  
 fira\_helper\_set\_session\_lower\_aoa\_bound\_config\_azimuth\_2  
 (C function), 147  
 fira\_helper\_set\_session\_lower\_aoa\_bound\_config\_elevation  
 (C function), 149  
 fira\_helper\_set\_session\_mac\_address\_mode (C func-  
 tion), 113  
 fira\_helper\_set\_session\_mac\_fcs\_type (C function),  
 122  
 fira\_helper\_set\_session\_mac\_payload\_encryption  
 (C function), 119  
 fira\_helper\_set\_session\_max\_number\_of\_measurements  
 (C function), 115  
 fira\_helper\_set\_session\_max\_rr\_retry (C function),  
 115  
 fira\_helper\_set\_session\_measurement\_sequence (C  
 function), 124  
 fira\_helper\_set\_session\_min\_frames\_per\_rr (C  
 function), 125  
 fira\_helper\_set\_session\_mtu\_size (C function), 126  
 fira\_helper\_set\_session\_multi\_node\_mode (C func-  
 tion), 109  
 fira\_helper\_set\_session\_near\_proximity\_config\_cm  
 (C function), 127  
 fira\_helper\_set\_session\_number\_of\_sts\_segments  
 (C function), 122  
 fira\_helper\_set\_session\_owr\_aoa\_measurement\_ntf\_period  
 (C function), 126  
 fira\_helper\_set\_session\_phr\_data\_rate (C func-  
 tion), 123  
 fira\_helper\_set\_session\_preamble\_code\_index (C  
 function), 116  
 fira\_helper\_set\_session\_preamble\_duration (C  
 function), 116  
 fira\_helper\_set\_session\_prf\_mode (C function), 123  
 fira\_helper\_set\_session\_priority (C function), 113  
 fira\_helper\_set\_session\_psdu\_data\_rate (C func-  
 tion), 117  
 fira\_helper\_set\_session\_ranging\_round\_control  
 (C function), 114  
 fira\_helper\_set\_session\_ranging\_round\_usage (C  
 function), 108  
 fira\_helper\_set\_session\_report\_psdus (C function),  
 103  
 fira\_helper\_set\_session\_report\_rssi (C function),  
 120

`fira_helper_set_session_result_report_config` (C function), 120  
`fira_helper_set_session_rframe_config` (C function), 116  
`fira_helper_set_session_round_duration_slots` (C function), 111  
`fira_helper_set_session_round_hopping` (C function), 113  
`fira_helper_set_session_schedule_mode` (C function), 114  
`fira_helper_set_session_session_info_ntf_config` (C function), 127  
`fira_helper_set_session_sfd_id` (C function), 117  
`fira_helper_set_session_short_address` (C function), 110  
`fira_helper_set_session_slot_duration_rstu` (C function), 111  
`fira_helper_set_session_static_sts_iv` (C function), 118  
`fira_helper_set_session_sts_config` (C function), 109  
`fira_helper_set_session_sts_length` (C function), 125  
`fira_helper_set_session_sub_session_id` (C function), 117  
`fira_helper_set_session_time0_ns` (C function), 111  
`fira_helper_set_session_time_base` (C function), 112  
`fira_helper_set_session_upper_aoa_bound_config_azimuth_ranging` (C function), 148  
`fira_helper_set_session_upper_aoa_bound_config_elevation_ranging` (C function), 150  
`fira_helper_set_session_ut_device_id` (C function), 154  
`fira_helper_set_session_ut_device_id_len` (C function), 154  
`fira_helper_set_session_ut_random_window` (C function), 153  
`fira_helper_set_session_ut_report_config_count` (C function), 155  
`fira_helper_set_session_ut_report_config_event` (C function), 155  
`fira_helper_set_session_ut_report_config_interval` (C function), 155  
`fira_helper_set_session_ut_tx_interval_ms` (C function), 153  
`fira_helper_set_session_ut_tx_timestamp_len` (C function), 154  
`fira_helper_set_session_vendor_id` (C function), 118  
`fira_helper_set_session_vupper64` (C function), 118  
`fira_helper_set_sub_session_key` (C function), 151  
`fira_helper_start_session` (C function), 94  
`fira_helper_stop_session` (C function), 95  
`fira_helper_update_dt_anchor_ranging_rounds` (C function), 152  
`fira_hus_controlee_config_cmd` (C struct), 90  
`fira_hus_controlee_phase_config` (C struct), 90  
`fira_hus_controller_config_cmd` (C struct), 89  
`fira_hus_controller_phase_config` (C struct), 88  
`fira_link_layer_mode` (C enum), 176  
`fira_mac_fcs_type` (C enum), 169  
`fira_measurement_report_originator` (C enum), 163  
`fira_measurement_report_type` (C enum), 163  
`fira_measurement_sequence` (C struct), 173  
`fira_measurement_sequence_step` (C struct), 172  
`fira_measurement_type` (C enum), 171  
`fira_message_id` (C enum), 176  
`fira_multi_node_mode` (C enum), 162  
`fira_multicast_update_status` (C enum), 182  
`fira_owr_aoa_measurement_ntf_period` (C enum), 183  
`fira_owr_aoa_measurements` (C struct), 81  
`fira_owr_aoa_ranging_results` (C struct), 82  
`fira_owr_dtm_timestamp_len` (C enum), 179  
`fira_owr_dtm_timestamp_type` (C enum), 179  
`fira_owr_message_type` (C enum), 164  
`fira_owr_utm_device_id_len` (C enum), 181  
`fira_owr_utm_timestamp_len` (C enum), 180  
`fira_phr_data_rate` (C enum), 169  
`fira_preamble_duration` (C enum), 167  
`fira_prf_mode` (C enum), 166  
`fira_psdu_data_rate` (C enum), 168  
`fira_ranging_diagnostics_frame_report_flags` (C enum), 173  
`fira_ranging_diagnostics_frame_reports_status_flags` (C enum), 174  
`fira_ranging_info` (C struct), 79  
`fira_ranging_round_control_flags` (C enum), 180  
`fira_ranging_round_usage` (C enum), 161  
`fira_result_report_config_flags` (C enum), 177  
`fira_rframe_config` (C enum), 165  
`fira_schedule_mode` (C enum), 165  
`fira_session_info_ntf_config` (C enum), 175  
`fira_session_multicast_list_ntf_content` (C struct), 86  
`fira_session_time_base` (C struct), 171  
`fira_sfd_id` (C enum), 167  
`fira_sts_length` (C enum), 174  
`fira_sts_segments` (C enum), 168  
`fira_twr_measurements` (C struct), 78  
`fira_twr_ranging_results` (C struct), 80  
`fira_ul_tdoa_ranging_results` (C struct), 82

## M

`measurement_sequence` (C struct), 65

## P

`pctt_device_role` (C enum), 195  
`pctt_helper_close` (C function), 191  
`pctt_helper_notification_cb_t` (C function), 190  
`pctt_helper_open` (C function), 190  
`pctt_helper_pll_lock` (C function), 194



[pctt\\_helper\\_session\\_deinit \(C function\), 193](#)  
[pctt\\_helper\\_session\\_get\\_params \(C function\), 193](#)  
[pctt\\_helper\\_session\\_get\\_state \(C function\), 193](#)  
[pctt\\_helper\\_session\\_init \(C function\), 191](#)  
[pctt\\_helper\\_session\\_set\\_params \(C function\), 194](#)  
[pctt\\_helper\\_session\\_start \(C function\), 192](#)  
[pctt\\_helper\\_set\\_scheduler \(C function\), 191](#)  
[pctt\\_helper\\_set\\_test\\_payload \(C function\), 192](#)  
[pctt\\_helper\\_tx\\_cw \(C function\), 194](#)  
[pctt\\_number\\_of\\_sts\\_segments \(C enum\), 198](#)  
[pctt\\_parameters \(C struct\), 184](#)  
[pctt\\_phr\\_data\\_rate \(C enum\), 199](#)  
[pctt\\_preamble\\_duration \(C enum\), 196](#)  
[pctt\\_prf\\_mode \(C enum\), 196](#)  
[pctt\\_psdu\\_data\\_rate \(C enum\), 198](#)  
[pctt\\_result\\_data \(C struct\), 187](#)  
[pctt\\_rframe\\_config \(C enum\), 195](#)  
[pctt\\_session\\_parameters \(C struct\), 185](#)  
[pctt\\_session\\_state \(C enum\), 201](#)  
[pctt\\_sfd\\_id \(C enum\), 197](#)  
[pctt\\_status\\_ranging \(C enum\), 199](#)  
[pctt\\_sts\\_length \(C enum\), 201](#)  
[pctt\\_test\\_payload \(C struct\), 185](#)  
[power\\_state\\_stats \(C struct\), 31](#)

## Q

[QDEPRECATED \(C macro\), 3](#)

## S

[session\\_parameters \(C struct\), 65](#)

## U

[update\\_dt\\_anchor\\_ranging\\_rounds\\_cmd \(C struct\), 75](#)  
[update\\_dt\\_anchor\\_ranging\\_rounds\\_rsp \(C struct\), 75](#)  
[uwbmac\\_buf\\_alloc \(C function\), 37](#)  
[uwbmac\\_buf\\_alloc\\_quota \(C function\), 37](#)  
[UWBMAC\\_BUF\\_CB\\_SIZE \(C macro\), 37](#)  
[uwbmac\\_buf\\_free \(C function\), 37](#)  
[uwbmac\\_buf\\_free\\_msg\\_priv \(C function\), 45](#)  
[uwbmac\\_buf\\_get\\_data \(C function\), 43](#)  
[uwbmac\\_buf\\_get\\_frag\\_len \(C function\), 44](#)  
[uwbmac\\_buf\\_get\\_len \(C function\), 44](#)  
[uwbmac\\_buf\\_get\\_next\\_frag \(C function\), 43](#)  
[uwbmac\\_buf\\_get\\_size \(C function\), 44](#)  
[uwbmac\\_buf\\_headroom \(C function\), 38](#)  
[uwbmac\\_buf\\_pull \(C function\), 40](#)  
[uwbmac\\_buf\\_push \(C function\), 40](#)  
[uwbmac\\_buf\\_put \(C function\), 39](#)  
[uwbmac\\_buf\\_put\\_data \(C function\), 39](#)  
[uwbmac\\_buf\\_put\\_u8 \(C function\), 40](#)  
[uwbmac\\_buf\\_queue\\_empty \(C function\), 41](#)  
[uwbmac\\_buf\\_queue\\_init \(C function\), 41](#)  
[uwbmac\\_buf\\_queue\\_is\\_last \(C function\), 42](#)  
[uwbmac\\_buf\\_queue\\_next \(C function\), 42](#)  
[uwbmac\\_buf\\_queue\\_peek \(C function\), 42](#)

[uwbmac\\_buf\\_queue\\_pop \(C function\), 43](#)  
[uwbmac\\_buf\\_queue\\_purge \(C function\), 43](#)  
[uwbmac\\_buf\\_queue\\_push \(C function\), 41](#)  
[uwbmac\\_buf\\_queue\\_put \(C function\), 41](#)  
[uwbmac\\_buf\\_reserve \(C function\), 38](#)  
[uwbmac\\_buf\\_set\\_queue\\_mapping \(C function\), 45](#)  
[uwbmac\\_buf\\_tailroom \(C function\), 38](#)  
[uwbmac\\_buf\\_trim \(C function\), 38](#)  
[uwbmac\\_calibration\\_transaction\\_end \(C function\), 15](#)  
[uwbmac\\_calibration\\_transaction\\_start \(C function\), 15](#)  
[uwbmac\\_call\\_region \(C function\), 24](#)  
[uwbmac\\_call\\_region\\_cb \(C function\), 5](#)  
[uwbmac\\_call\\_region\\_free \(C function\), 24](#)  
[uwbmac\\_call\\_scheduler \(C function\), 23](#)  
[uwbmac\\_call\\_testmode \(C function\), 27](#)  
[uwbmac\\_channel\\_create \(C function\), 8](#)  
[uwbmac\\_channel\\_receive \(C function\), 8](#)  
[uwbmac\\_channel\\_release \(C function\), 8](#)  
[uwbmac\\_channel\\_set\\_timeout \(C function\), 8](#)  
[uwbmac\\_close\\_scheduler \(C function\), 21](#)  
[uwbmac\\_data\\_ops \(C struct\), 5](#)  
[uwbmac\\_device\\_info \(C struct\), 33](#)  
[uwbmac\\_device\\_state \(C enum\), 4](#)  
[uwbmac\\_device\\_state\\_cb \(C function\), 4](#)  
[uwbmac\\_device\\_state\\_report \(C function\), 63](#)  
[uwbmac\\_event\\_report \(C function\), 64](#)  
[uwbmac\\_exit \(C function\), 10](#)  
[uwbmac\\_get\\_calibration \(C function\), 16](#)  
[uwbmac\\_get\\_calibration\\_key\\_name \(C function\), 17](#)  
[uwbmac\\_get\\_channel \(C function\), 14](#)  
[uwbmac\\_get\\_channel\\_preamble\\_code \(C function\), 14](#)  
[uwbmac\\_get\\_device\\_count \(C function\), 6](#)  
[uwbmac\\_get\\_device\\_info \(C function\), 36](#)  
[uwbmac\\_get\\_low\\_power\\_mode \(C function\), 34](#)  
[uwbmac\\_get\\_pm\\_min\\_inactivity\\_s4 \(C function\), 34](#)  
[uwbmac\\_get\\_region\\_parameters \(C function\), 23](#)  
[uwbmac\\_get\\_scheduler \(C function\), 21](#)  
[uwbmac\\_get\\_scheduler\\_parameters \(C function\), 22](#)  
[uwbmac\\_get\\_spi\\_pids \(C function\), 26](#)  
[uwbmac\\_get\\_supported\\_channels \(C function\), 7](#)  
[uwbmac\\_get\\_time\\_ns \(C function\), 25](#)  
[uwbmac\\_get\\_trace\\_modules \(C function\), 31](#)  
[uwbmac\\_get\\_uwb\\_device\\_stats \(C function\), 36](#)  
[uwbmac\\_get\\_version \(C function\), 25](#)  
[uwbmac\\_init \(C function\), 10](#)  
[uwbmac\\_init\\_device \(C function\), 7](#)  
[uwbmac\\_is\\_started \(C function\), 11](#)  
[uwbmac\\_is\\_trace\\_module\\_enabled \(C function\), 31](#)  
[uwbmac\\_list\\_calibration\\_context \(C struct\), 17](#)  
[uwbmac\\_list\\_calibrations \(C function\), 18](#)  
[UWBMAC\\_MAX\\_CHANNEL\\_COUNT \(C macro\), 4](#)  
[uwbmac\\_msg \(C struct\), 45](#)  
[uwbmac\\_msg\\_copy \(C function\), 46](#)  
[uwbmac\\_msg\\_free\\_priv \(C function\), 46](#)

[uwbmac\\_msg\\_init \(C function\), 46](#)  
[uwbmac\\_msg\\_length \(C function\), 47](#)  
[uwbmac\\_msg\\_payload \(C function\), 47](#)  
[uwbmac\\_msg\\_read\\_data \(C function\), 57](#)  
[uwbmac\\_msg\\_read\\_nested \(C function\), 56](#)  
[uwbmac\\_msg\\_read\\_tag \(C function\), 56](#)  
[uwbmac\\_msg\\_size \(C function\), 47](#)  
[uwbmac\\_parser\\_add \(C function\), 52](#)  
[uwbmac\\_parser\\_add\\_binary \(C function\), 55](#)  
[uwbmac\\_parser\\_add\\_bool \(C function\), 53](#)  
[uwbmac\\_parser\\_add\\_flag \(C function\), 52](#)  
[uwbmac\\_parser\\_add\\_nested \(C function\), 56](#)  
[uwbmac\\_parser\\_add\\_none \(C function\), 52](#)  
[uwbmac\\_parser\\_add\\_s16 \(C function\), 53](#)  
[uwbmac\\_parser\\_add\\_s32 \(C function\), 53](#)  
[uwbmac\\_parser\\_add\\_s64 \(C function\), 54](#)  
[uwbmac\\_parser\\_add\\_s8 \(C function\), 53](#)  
[uwbmac\\_parser\\_add\\_string \(C function\), 55](#)  
[uwbmac\\_parser\\_add\\_u16 \(C function\), 54](#)  
[uwbmac\\_parser\\_add\\_u32 \(C function\), 54](#)  
[uwbmac\\_parser\\_add\\_u64 \(C function\), 55](#)  
[uwbmac\\_parser\\_add\\_u8 \(C function\), 54](#)  
[uwbmac\\_parser\\_element \(C struct\), 49](#)  
[uwbmac\\_parser\\_init\\_msg \(C function\), 49](#)  
[uwbmac\\_parser\\_init\\_nested\\_loop \(C function\), 50](#)  
[uwbmac\\_parser\\_is\\_present \(C function\), 51](#)  
[uwbmac\\_parser\\_next\\_nested\\_loop\\_element \(C function\), 51](#)  
[uwbmac\\_parser\\_read \(C function\), 50](#)  
[uwbmac\\_parser\\_read\\_array \(C function\), 50](#)  
[uwbmac\\_payload\\_type \(C enum\), 47](#)  
[uwbmac\\_pids\\_info \(C struct\), 25](#)  
[uwbmac\\_poll\\_events \(C function\), 11](#)  
[uwbmac\\_power\\_stats \(C struct\), 32](#)  
[uwbmac\\_query\\_gpio\\_timestamp \(C function\), 35](#)  
[uwbmac\\_region\\_call\\_reply \(C function\), 64](#)  
[uwbmac\\_register\\_data\\_ops \(C function\), 9](#)  
[uwbmac\\_register\\_device\\_state\\_callback \(C function\), 7](#)  
[uwbmac\\_register\\_report\\_callback \(C function\), 9](#)  
[uwbmac\\_register\\_testmode\\_callback \(C function\), 27](#)  
[uwbmac\\_reinit\\_crypto \(C function\), 36](#)  
[uwbmac\\_reset\\_calibration \(C function\), 18](#)  
[uwbmac\\_rx\\_queue\\_stop \(C function\), 13](#)  
[uwbmac\\_rx\\_queue\\_wake \(C function\), 13](#)  
[uwbmac\\_se\\_derive\\_key \(C function\), 35](#)  
[uwbmac\\_se\\_set\\_key \(C function\), 35](#)  
[uwbmac\\_set\\_calibration \(C function\), 16](#)  
[uwbmac\\_set\\_channel \(C function\), 13](#)  
[uwbmac\\_set\\_channel\\_preamble\\_code \(C function\), 14](#)  
[uwbmac\\_set\\_extended\\_addr \(C function\), 19](#)  
[uwbmac\\_set\\_frame\\_retries \(C function\), 12](#)  
[uwbmac\\_set\\_low\\_power\\_mode \(C function\), 33](#)  
[uwbmac\\_set\\_pan\\_id \(C function\), 18](#)  
[uwbmac\\_set\\_pm\\_min\\_inactivity\\_s4 \(C function\), 34](#)  
[uwbmac\\_set\\_promiscuous\\_mode \(C function\), 20](#)  
[uwbmac\\_set\\_region\\_parameters \(C function\), 22](#)  
[uwbmac\\_set\\_regions \(C function\), 22](#)  
[uwbmac\\_set\\_scanning\\_mode \(C function\), 26](#)  
[uwbmac\\_set\\_scheduler \(C function\), 20](#)  
[uwbmac\\_set\\_scheduler\\_parameters \(C function\), 21](#)  
[uwbmac\\_set\\_short\\_addr \(C function\), 19](#)  
[uwbmac\\_start \(C function\), 10](#)  
[uwbmac\\_stop \(C function\), 11](#)  
[uwbmac\\_testmode\\_cb\\_t \(C function\), 26](#)  
[uwbmac\\_testmode\\_reply \(C function\), 64](#)  
[uwbmac\\_trace\\_info \(C struct\), 29](#)  
[uwbmac\\_trace\\_init \(C function\), 28](#)  
[uwbmac\\_trace\\_module\\_enable \(C function\), 30](#)  
[uwbmac\\_trace\\_module\\_enable\\_by\\_id \(C function\), 30](#)  
[uwbmac\\_trace\\_module\\_ids \(C enum\), 28](#)  
[uwbmac\\_tx \(C function\), 12](#)  
[uwbmac\\_tx\\_drop \(C function\), 13](#)  
[uwbmac\\_uwb\\_device\\_stats \(C struct\), 32](#)  
[uwbmac\\_writer\\_add \(C function\), 58](#)  
[uwbmac\\_writer\\_add\\_binary \(C function\), 62](#)  
[uwbmac\\_writer\\_add\\_bool \(C function\), 59](#)  
[uwbmac\\_writer\\_add\\_flag \(C function\), 58](#)  
[uwbmac\\_writer\\_add\\_s16 \(C function\), 59](#)  
[uwbmac\\_writer\\_add\\_s32 \(C function\), 60](#)  
[uwbmac\\_writer\\_add\\_s64 \(C function\), 60](#)  
[uwbmac\\_writer\\_add\\_s8 \(C function\), 59](#)  
[uwbmac\\_writer\\_add\\_singleton\\_map \(C function\), 63](#)  
[uwbmac\\_writer\\_add\\_string \(C function\), 62](#)  
[uwbmac\\_writer\\_add\\_u16 \(C function\), 61](#)  
[uwbmac\\_writer\\_add\\_u32 \(C function\), 61](#)  
[uwbmac\\_writer\\_add\\_u64 \(C function\), 61](#)  
[uwbmac\\_writer\\_add\\_u8 \(C function\), 60](#)  
[uwbmac\\_writer\\_end\\_nested \(C function\), 63](#)  
[uwbmac\\_writer\\_init\\_msg \(C function\), 57](#)  
[uwbmac\\_writer\\_start\\_nested \(C function\), 62](#)  
[uwbmac\\_writer\\_success \(C function\), 58](#)