Developer

DWM3001CDK SDK Developer Guide

Qorvo

Release 0.1.1

Contents:

List of Figures

List of Tables

1 Revision History

Version	Date	Comment
DW3_QM33_SDK_0.1.0	2022-09-27	• Initial version

2 Overview

2.1 DWM3001CDK Package Content

The following table shows DWM3001CDK EVB package content once unzipped.

Table 2.1: DWM3001CDK package content

Folder name	Folder description
Binaries	Executable files to flash on the board
Doc	SDK user manual and release notes
Sources	DWM3001CDK SDK source code

Please refer to *Building SDK Firmware* section on how to build the firmware.

Please refer to *SDK Architecture* section for details on source code.

3 Board Connections

3.1 DWM3001CDK

Connect the two micro-USB connectors available on the board as in *DWM3001CDK connections*

- J9: used for flashing/debugging.
- J20: used for UART/USB communications with MCU.

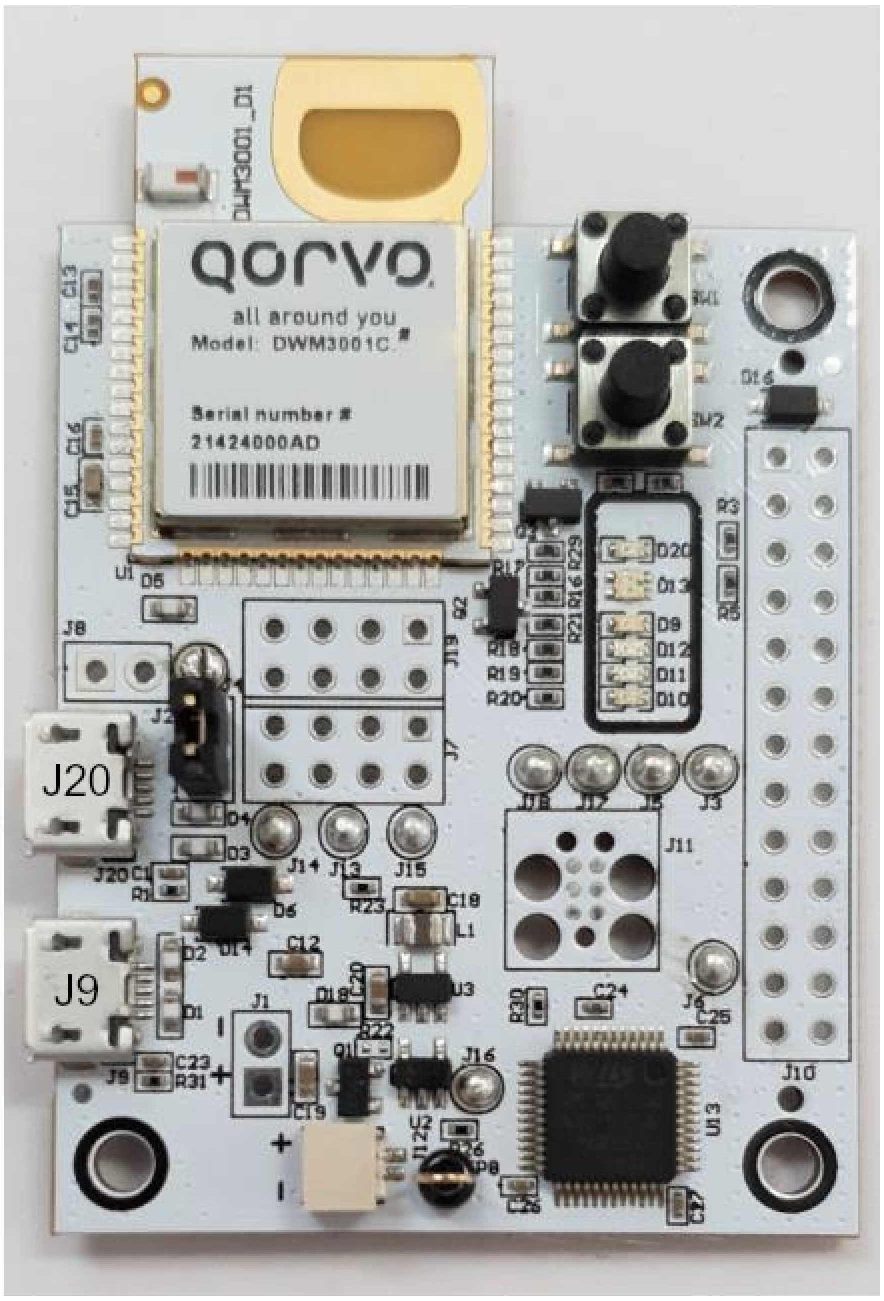


Fig. 3.1: DWM3001CDK connections

4 Flashing SDK Firmware

4.1 Required Tools

4.1.1 Hardware tools

For the boards equipped with Segger J-Link OB, the same features as J-Link BASE are available via USB.

For the other boards, the following hardware tools are needed to flash the binary into the target board over SWD interface:

- J-Link debug/flash probe[1]
- J-Link adapter CortexM[2]

4.1.2 Software tools

Boards can be flashed using Segger J-Flash Lite:

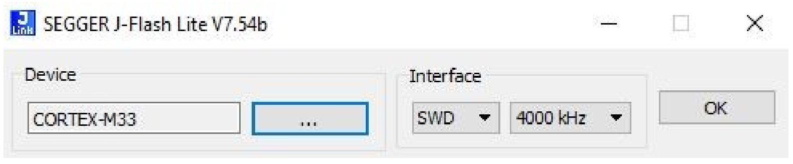
- Segger J-Flash Lite[3]

[1] https://www.segger.com/products/debug-probes/j-link/
[2] https://www.segger.com/products/debug-probes/j-link/accessories/adapters/9-pin-cortex-m-adapter/
[3] https://wiki.segger.com/J-Flash_Lite

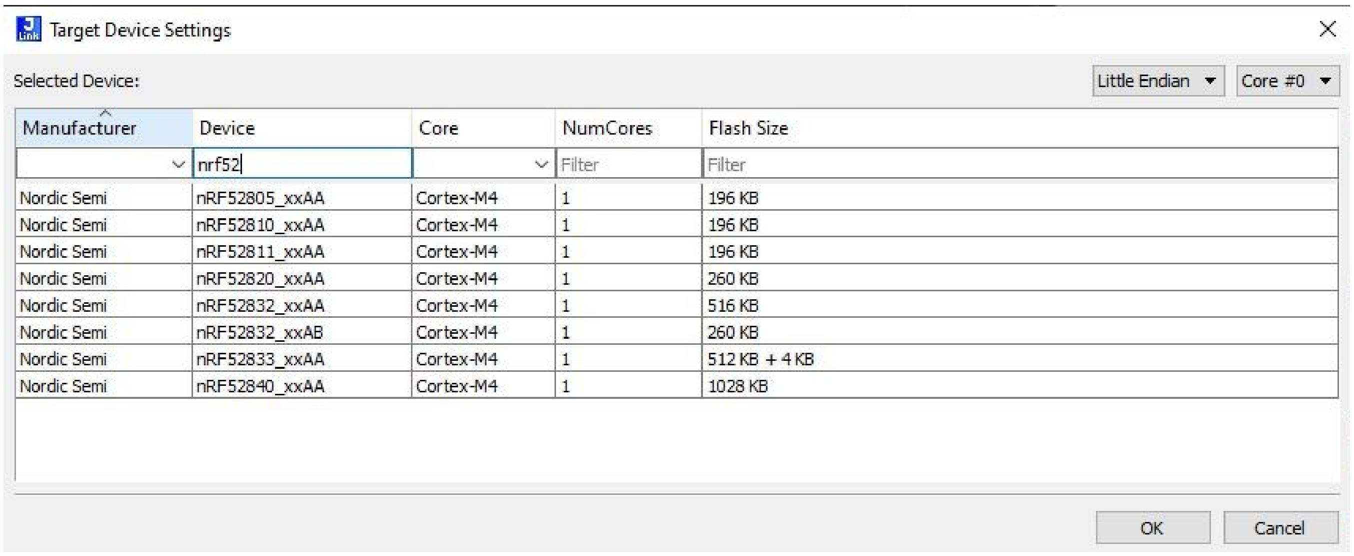
4.2 Flashing

4.2.1 Using J-Flash-Lite

1. Open J-Flash lite. Click on "…" button.

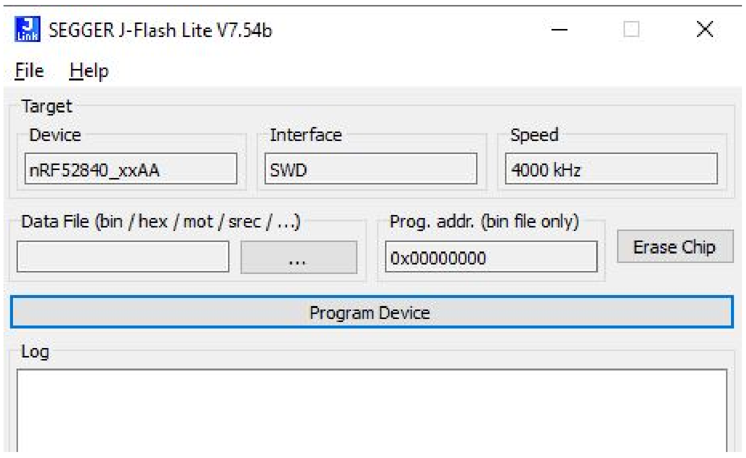


2. Select device and click on "OK"



Manufacturer	Device	Core	NumCores	Flash Size
	nrf52		Filter	Filter
Nordic Semi	nRF52805_xxAA	Cortex-M4	1	196 KB
Nordic Semi	nRF52810_xxAA	Cortex-M4	1	196 KB
Nordic Semi	nRF52811_xxAA	Cortex-M4	1	196 KB
Nordic Semi	nRF52820_xxAA	Cortex-M4	1	260 KB
Nordic Semi	nRF52832_xxAA	Cortex-M4	1	516 KB
Nordic Semi	nRF52832_xxAB	Cortex-M4	1	260 KB
Nordic Semi	nRF52833_xxAA	Cortex-M4	1	512 KB + 4 KB
Nordic Semi	nRF52840_xxAA	Cortex-M4	1	1028 KB

3. Click on "…" button. Select Data file (.hex) and "Program Device".



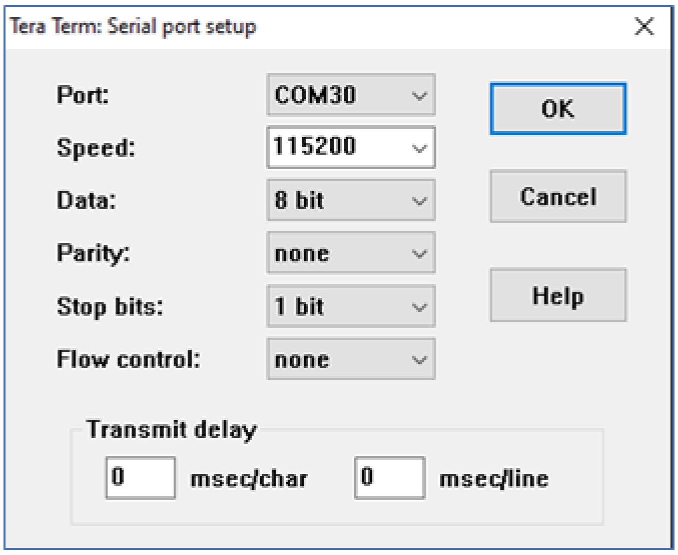
Device: nRF52840_xxAA
Interface: SWD
Speed: 4000 kHz
Data File (bin / hex / mot / srec / …)
Prog. addr. (bin file only): 0x00000000

5 Command Line Interface - CLI Build

5.1 Setup

Open a Serial Terminal (example using TeraTerm[4]) and set "Serial Port" as follow :

- Baudrate : 115200
- Number of data bits : 8
- Parity : None
- Number of stop bits : 1
- Flow control : None



Tera Term: Serial port setup

Port:	COM30
Speed:	115200
Data:	8 bit
Parity:	none
Stop bits:	1 bit
Flow control:	none

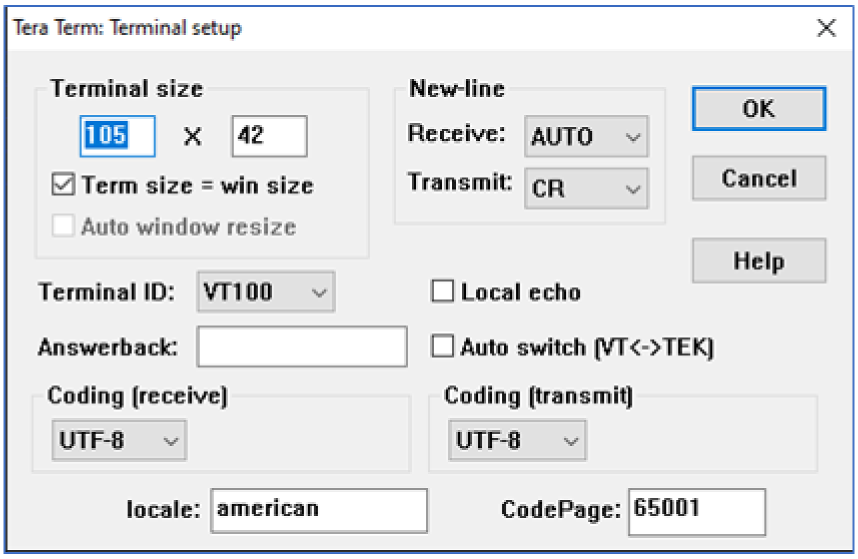
Transmit delay

0 msec/char 0 msec/line

Fig. 5.1: Com serial port settings.

[4] https://ttssh2.osdn.jp/index.html.en

Set “New-line” as follow in the terminal setup.



Tera Term: Terminal setup

Terminal size: 105 X 42
☑ Term size = win size
☐ Auto window resize

New-line
Receive: AUTO
Transmit: CR

Terminal ID: VT100
Answerback:
☐ Local echo
☐ Auto switch (VT<->TEK)

Coding (receive): UTF-8
Coding (transmit): UTF-8

locale: american
CodePage: 65001

Fig. 5.2: New-line serial port settings.

5.2 Anytime Commands

5.2.1 Overview

Anytime commands listed in table below can be executed anytime.

Table 5.1: Anytime Commands

Command	Description
HELP	Outputs a list of all known commands available in the current mode of operation or shows the help of the particular command <CMD>. Equivalent shortcut is “?”.
STAT	Reports the status. Gives a dump of software version info, configuration values and the current operation mode.
DECA$	Reports the software version info, configuration values and the current operation mode.
STOP	Stops running any of top-level applications and places the node to the STOP mode, where only core tasks are running.
SAVE	Stores runtime configuration to NVM.
THREAD	Reports information about running threads and memory consumption.

Application or settings commands will return “ok” if the command was properly executed.

5.2.2 HELP

This command displays the help information.

Command:

```
help
```

or

```
?
```

Response:

```
nRF52840DK - DW3_QM33_SDK - FreeRTOS
---- Anytime commands --------
HELP      ?         STOP      THREAD
STAT      SAVE      DECA$
---- Application selection ---
LISTENER2 TCFM      TCWM      RESPF
INITF
---- LISTENER Options --------
LSTAT
---- FiRa Options --------
PAVRG
---- IDLE time commands -----
ANTTXA    ANTRXA    XTALTRIM  PDOAOFF
UART
---- Service commands --------
RESTORE   DIAG      UWBCFG    STSKEYIV
TXPOWER   ANTENNA   DECAID    VERSION
ok
```

Note: This command may also take any command as a parameter. It allows to display its description. Examples will follow.

```
help initf
INITF:
INITF [RFRAME BPRF set] [Slot duration rstu] [Block duration ms] [Round durat
ion slots] [RR usage] [Session id] [vupper64 xx:xx:xx:xx:xx:xx:xx:xx] [Multi
node mode] [Round hopping] [Initiator Addr] [Responder 1 Addr] ... [Responder
n Addr]


help respf
RESPF:
RESPF [RFRAME BPRF set] [Slot duration rstu] [Block duration ms] [Round durat
ion slots] [RR usage] [Session id] [vupper64 xx:xx:xx:xx:xx:xx:xx:xx] [Multi
node mode] [Round hopping] [Initiator Addr] [Responder Addr]
```

5.2.3 STAT

This command reports the status. It gives a dump of software version information, supported applications and the current operation mode.

```
stat
MODE: STOP
LAST ERR CODE: 0
MAX MSG LEN: 0
JS010F{"Info":{
"Device":"nRF52840DK - DW3_QM33_SDK - FreeRTOS",
"Current App":"STOP",
"Version":"0.1.0-220922",
"Build":"Sep 22 2022 16:14:05",
"Apps":["LISTENER2","TCFM","TCWM","RESPF","INITF"],
"Driver":"DW3XXX Device Driver Version 06.00.14",
"UWB stack":"R11.9.2"}}
ok
```

5.2.4 DECA$

This command reports the software version information, configuration values and the current operation mode.

```
deca$
JS010F{"Info":{
"Device":"nRF52840DK - DW3_QM33_SDK - FreeRTOS",
"Current App":"STOP",
"Version":"0.1.0-220922",
"Build":"Sep 22 2022 16:14:05",
"Apps":["LISTENER2","TCFM","TCWM","RESPF","INITF"],
"Driver":"DW3XXX Device Driver Version 06.00.14",
"UWB stack":"R11.9.2"}}
ok
```

5.2.5 SAVE

This command saves the configuration and current operational mode to the NVM (Non-Volatile-Memory). Saved UW-BCFG parameters will stay after reboot or power off-on therefore making this ideal for battery powered configuration options. After a reboot (or power on) the device will start in the selected operational mode.

For INITF and RESPF commands the default operational parameters will be applied (i.e. Deferred DS-TWR with 2ms slot and 200ms interval). The default parameters could be changed manually in the common_fira.c file, function scan_fira_params()if desired.

```
save
ok
```

5.2.6 STOP

It stops running any of top-level applications and places the node to the STOP mode, where only core tasks are running.

```
stop
ok
```

5.2.7 THREAD

This command reports information about running threads and their stack usage.

For each Thread, there is a ratio display in Stack usage column

- The 1st value is the maximum stack depth reached during runtime
- The 2nd value is the maximum stack allocated to the thread

```
THREAD NAME             Stack usage
ctrlTask                396/2048
IDLE                    164/512
defaultTask             148/4304
flushtask               184/512
Tmr Svc                 156/512
Total HEAP              51200
Current HEAP used       9744
Max HEAP used           9936
ok
```

5.3 Service Commands

Service commands can only be executed in STOP mode when no appliations are running.

Table 5.2: Service Commands

Command	Description
RESTORE	Restores the default configuration
UWBCFG	Reports a list of UWB configuration parameters and value.
VERSION	Reports firmware version currently running in the target.
DIAG	Enables diagnostic mode.
TXPOWER	Reports TxPower or changes its value.
DECAID	Reports information about the chip.
ANTENNA	Reports or sets the list of used antennas.

5.3.1 RESTORE

It restores the default configuration for both UWB and the System. Use *SAVE* command after execution to save default config back to NVM.

```
restore
ok
save
ok
```

5.3.2 UWBCFG

Reports a list of UWB configuration parameters and value.

```
uwbcfg
JS00BE{"UWB PARAM":{
"CHAN":9,
"PLEN":64,
"PAC":8,
"TXCODE":9,
"RXCODE":9,
"SFDTYPE":3,
"DATARATE":6810,
"PHRMODE":0,
"PHRRATE":0,
"SFDTO":65,
"STSMODE":0,
"STSLEN":64,
"PDOAMODE":1}}
```

Note: To see UWB parameters, use “UWBCFG”. To set UWB config, use “UWBCFG <List of parameters>” with parameters listed in table 5.3.

```
uwbcfg 9 64 8 9 9 3 6810 0 0 65 0 64 1
JS00BE{"UWB PARAM":{
"CHAN":9,
"PLEN":64,
"PAC":8,
"TXCODE":9,
"RXCODE":9,
"SFDTYPE":3,
```

(continues on next page)

(continued from previous page)

```
"DATARATE":6810,
"PHRMODE":0,
"PHRRATE":0,
"SFDTO":65,
"STSMODE":0,
"STSLEN":64,
"PDOAMODE":1}}
ok
```

Table 5.3: Parameter list and default value

Order	Config Parameter	Default value	Description
1	**CHAN**	**9**	**Channel: 5, 9**
2	PLEN	64	Preamble length: 32, 64, 128, 256, 512, 1024, 2058, 4096
3	RX PAC	8	RX PAC size: 4, 8, 16, 32
4	**TX CODE**	**9**	**Tx Preamble code: 9, 10, 11, 12**
5	RX CODE	9	Rx Preamble code: 9, 10, 11, 12
6	SFD TYPE	3	0 = 4A 8-bit, 1 = DW 8-bit, 2 = DW 16-bit, 3 = 4Z 8-bit
7	DATARATE	6810	Datarate in Kbps: 850, 6810, 27240, 31200, 54480.
8	PHR MODE	0	0 = Standard, 1 = DW extended
9	PHR RATE	0	0 = 850kbps (Standard), 1 = 6.8Mbps
10	SFD TO	65	SFD Timeout = PLEN + 1 + SFD_length - RX_PAC
11	STS MODE	0	0 = OFF, 1 = SP1, 2 = SP2, 3 = SP3, 9 = SP1 + SDC
12	STS LEN	64	Possible values: 2048, 1024, 512, 256, 128, 64, 32
13	PDOA MODE	1	0 = OFF, 1 = ON using Preamble, 3 = ON using STS

For FiRa compliant ranging and AoA, only CHAN and TX_CODE are used during ranging. The remainder of the values will be overwritten to be FiRa compliant application.

5.3.3 VERSION

Reports firmware version currently running in the target.

```
version
VERSION:0.1.0-220922
ok
```

5.3.4 DIAG

DIAG provides different way to display the log while ranging. By default it is set to 0.

With DIAG set to 1, it allows to display RSSI and NLOS (Non-LIne of Sight / FROM metrics).

Both RSSI and NLOS calculations make use of IPATOV, STS1, STS2, and CIR (Channel Impulse Response) parameters. In particular, three points of the first path amplitude magnitude, the number of symbols accumulated, and the CIR power value. Details on RSSI can be found in the DW3000 User Manual, sections 4.7.1 and 4.7.2 and for NLOS probability in the App note APS006 Part 3.

Note:

In DIAG 1 mode, FiRa slot cannot be less than 3ms due to delays in acquiring and processing RSSI and NLOS. Therefore, the following should be executed:

```
diag 1
ok
initf 4 3600

where "4" indicates BPRF #4 is used for RFRAME (SP3, SFD_4Z). and 3600 corresponds to 3ms slot␣
↪duration.
```

5.3.5 TXPOWER

TX power is a 32-bit register. Each byte is able to control the TX power across the 4 different fields (data, header, synchronization, and STS) in the TX packet.

We typically recommend keeping the same value for each of these fields.

TxPower can be adapted for a specific device through a spectrum analyzer. Default value is adapted to fit current FCC regulation on channel 9.

To change the TX power, TXPOWER command should be used.

```
help txpower
TXPOWER:
Tx Power settings.
Usage: To see Tx power "txpower". To set the Tx power "txpower 0x<POWER_HEX> 0x<PGDLY_
↪HEX> 0x<PGCOUNT_HEX>"
```

5.3.6 DECAID

This command reports information about the chip.

```
decaid
Decawave device ID = 0xdeca0314
Decawave lotID = 0x00000000, partID = 0x0aaaaaaa
ok
```

5.3.7 ANTENNA

This command reports or sets the list of used antennas.

To see antennas configuration, use "ANTENNA".

```
antenna
CURRENT ANTENNA_TYPE:
PORT1: JOLIE9
PORT2: JOLIE9
PORT3: NONE
PORT4: NONE
ok
```

To set antennas configuration, use "ANTENNA <PORT1> <PORT2> <PORT3> <PORT4>".

```
ANTENNA JOLIE5 JOLIE5 JOLIE5 JOLIE5
CURRENT ANTENNA_TYPE:
PORT1: JOLIE5
```

(continues on next page)

(continued from previous page)

```
PORT2: JOLIE5
PORT3: JOLIE5
PORT4: JOLIE5
ok
```

This command changes the parameters depending on the antenna (LUT, Distance between centers of PDoA antennas).

Note: It is possible to see the antenna possibilities using the command ANTENNA VALUES. By default the name corresponds to NAME OF THE ANTENNA + CHANNEL.

5.4 IDLE time Commands

5.4.1 Overview

IDLE time commands listed in table below can be executed only when the mode is STOP, corresponding to IDLE state.

Table 5.4: Service Commands

Command	Description
ANTTXA	Reports Tx Antenna delay or changes its value
ANTRXA	Reports Rx Antenna delay or changes its value
XTALTRIM	Reports Crystal trim or changes its value
PDOAOFF	Reports PDoA offset or changes its value
UART	Enables selected UART

5.4.2 ANTTXA

Reports Tx Antenna delay or changes its value.

To see Tx Antenna delay, use “ANTTXA”.

```
ANTTXA
ANT_TXA: 16405
ok
```

To set Tx Antenna delay, use “ANTTXA <delay>”.

```
ANTTXA 16000
ok
```

5.4.3 ANTRXA

Reports Rx Antenna delay or changes its value.

To see Rx Antenna delay, use “ANTRXA”.

```
ANTRXA
ANT_RXA: 16405
ok
```

To set Rx Antenna delay, use “ANTRXA <delay>”.

```
ANTRXA 16500
ok
```

5.4.4 XTALTRIM

Reports Crystal trim or changes its value.

This allows a fine control over the crystal oscillator to tune the operating frequencies quite precisely. Details on Crystal Oscillator Trim can be found in the User Manual.

To see Crystal Trim, use “XTALTRIM”.

```
XTALTRIM
JS001F{"XTAL":{
"TEMP TRIM":"0x2e"}}
ok
```

To set Crystal Trim, use “XTALTRIM 0x<value>”.

```
XTALTRIM 0x55
JS001F{"XTAL":{
"TEMP TRIM":"0x55"}}
ok
```

5.4.5 PDOAOFF

Reports PDoA offset or changes its value.

PDoA offset is used to remove the constant value on the PDoA and align PDoA = 0 to AoA = 0.

To see PDoA offset, use “PDOAOFF”.

```
PDOAOFF
JS0011{"PDOAOFF_deg":0}
ok
```

To set PDoA offset, use “PDOAOFF <angle>”.

```
PDOAOFF 4
JS0011{"PDOAOFF_deg":4}
ok
```

5.4.6 UART

Enables selected UART for the communication.

To select desired UART, use “UART <UART_number>”.

```
UART 0
ok
```

5.5 Application Commands

5.5.1 SDK Applications

Top-level applications cannot run concurrently since they use the same resources.

Table 5.5: List of applications

Command	Description
INITF	Sets the board as INITIATOR in a FiRa ranging session.
RESPF	Sets the board as RESPONDER in a FiRa ranging session.
TCFM	Sets the board into Continuous frame mode.
TCWM	Sets the board into Continuous wave mode.
LISTENER2	Puts the board into receive mode and reports any received packets.

5.5.2 FiRa application : INITF / RESPF

INITF sets the board as INITIATOR in a FiRa ranging session. See table below for configuration parameters.

```
help initf
INITF:
INITF [RFRAME BPRF set] [Slot duration rstu] [Block duration ms] [Round durat
ion slots] [RR usage] [Session id] [vupper64 xx:xx:xx:xx:xx:xx:xx:xx] [Multi
node mode] [Round hopping] [Initiator Addr] [Responder 1 Addr] ... [Responder
n Addr]
```

RESPF sets the board as RESPONDER in a FiRa ranging session. See table below for configuration parameters.

```
help respf
RESPF:
RESPF [RFRAME BPRF set] [Slot duration rstu] [Block duration ms] [Round durat
ion slots] [RR usage] [Session id] [vupper64 xx:xx:xx:xx:xx:xx:xx:xx] [Multi
node mode] [Round hopping] [Initiator Addr] [Responder Addr]
```

Table 5.6: INITF/RESPF parameter list and default value

Order	Config Parameter	Default value	Description
1	RFRAME BPRF	4	RFRAME BRFF set as per FiRa spec: 4 - SP3 SFD4Z , 6 - SP3 SFD4A
2	RSTU slot duration	2400	Duration of the slot in RSTU time units. 1ms = 1200 RSTU
3	Block duration ms	200	Duration of the FiRa ranging block in ms
4	Round duration slots	25	Duration of the FiRa ranging round inside the block
5	Ranging Round usage	2	0- Not used, 1- SS-TWR, 2- DS-TWR
6	Session ID	42	Session ID
7	vupper64	01:02:03:04:05:06:07:08	Eight hexadecimal numbers, represented static part of the STS in FiRa standard, Hex values separated by ":"
8	Multi node mode	0	0 for unicast, 1 for multi-node configuration
9	Round hopping	0	0 for no round hopping, 1 for round hopping
10	Initiator address	0	Address of FiRa Initiator, Decimal value 0-65535
11	Responder address	1	Address of responder or set of addresses for multiple responders, Decimal value 0-65535

Table 5.7: BPRF mode operating parameter sets

BPRF Set#	SYNC PSR	SFD# (802.15. 2020)	SFD Lengh	STS nr of Segments	STS Segment Length	PHR + data	Data Rate (Mbps)	Description
1	64	0	8	0	n/a	Yes	6.8	SP0 (Mandatory)
2	64	2	8	0	n/a	Yes	6.8	SP0 (Mandatory)
3	64	2	8	1	64	Yes	6.8	SP1 (Mandatory)
4	64	2	8	1	64	No	n/a	SP3 (Mandatory)
5	64	0	8	1	64	Yes	6.8	SP1 (Optional)
6	64	0	8	1	64	No	n/a	SP3 (Optional)

Note: PDoA Mode parameter, which can be found in UWBCFG config, must be set to '1' to use any updated PDoA to AoA Lookup tables (LUTs). PDoA LUTs are generated to convert from Phase difference of Arrival at the two IC RF inputs to an actual AoA. PdoA to AoA will change for a given board, antenna and channel combination and will be the same for any board or antenna (no part-to-part variation). When a custom board and antenna are used, a LUT for that combination must be generated. (See Application Note APS330 AoA Design Guide)

Listing 5.1: Example of default INITF command

```
initf 4 2400 200 25 2 42 01:02:03:04:05:06:07:08 0 0 0 1
```

Listing 5.2: Example of default RESPF command

```
respf 4 2400 200 25 2 42 01:02:03:04:05:06:07:08 0 0 0 1
```

Listing 5.3: Example log from INITF execution

```
{"Block":0, "results":[{"Addr":"0x0001","Status":"Ok","D_cm":9,"LPDoA_deg":113.38,"LAoA_deg":46.
↪35,"LFoM":0,"RAoA_deg":-59.33,"CFO_ppm":-0.58}]}
{"Block":1, "results":[{"Addr":"0x0001","Status":"Ok","D_cm":16,"LPDoA_deg":123.96,"LAoA_deg
```

(continues on next page)

(continued from previous page)

```
↪":50.77,"LFoM":0,"RAoA_deg":37.51,"CFO_ppm":-0.47}]}
{"Block":2, "results":[{"Addr":"0x0001","Status":"Ok","D_cm":15,"LPDoA_deg":124.12,"LAoA_deg
↪":51.28,"LFoM":0,"RAoA_deg":-57.07,"CFO_ppm":-0.53}]}
{"Block":3, "results":[{"Addr":"0x0001","Status":"Ok","D_cm":16,"LPDoA_deg":121.94,"LAoA_deg
↪":49.80,"LFoM":0,"RAoA_deg":37.51,"CFO_ppm":-0.52}]}
{"Block":4, "results":[{"Addr":"0x0001","Status":"Ok","D_cm":13,"LPDoA_deg":123.28,"LAoA_deg
↪":50.77,"LFoM":0,"RAoA_deg":-67.64,"CFO_ppm":-0.31}]}
{"Block":5, "results":[{"Addr":"0x0001","Status":"Ok","D_cm":12,"LPDoA_deg":113.66,"LAoA_deg
↪":46.35,"LFoM":0,"RAoA_deg":-61.63,"CFO_ppm":-0.16}]}
{"Block":6, "results":[{"Addr":"0x0001","Status":"Ok","D_cm":16,"LPDoA_deg":120.49,"LAoA_deg
↪":49.29,"LFoM":0,"RAoA_deg":-60.51,"CFO_ppm":-0.43}]}
```

Table 5.8: INITF/RESPF log legend

Block	Ranging round number.
Addr	Short_addr: Address of the participating device.
Status	Status: “Ok” or “Err”
D_cm	Distance in cm (0.01m).
LPDoA_deg	Local raw PDoA measurements in deg, [-180..+180]. Required for PDoAoff adjustment.
LAoA_deg	AoA azimuth in deg, [-90..+90].
LFoM	aoa_fom: Estimation of local AoA reliability.
RAoA_deg	Remote_aoa_azimuth_2pi: “remote” over-the-air reporting of AoA from the Responder back to the Initiator. Reported as 0 if remote device is Non-AoA.
CFO_ppm	Crystal frequency offset in ppm units. This value is automatically adjusted by SW to be in the range of +/- [5..10]ppm. This value does not affect performance of TWR.

Note: When RESPF or INITF values are confirmed as above (see *UWBCFG*), the parameter “RFRAME” is referring to the STS Packet Structure (above is SP3) and “SFD ID” is the SFD# per Table 15-7c in the IEEE 802.15.4z-2020 standard. The combination above corresponds to the default BPRF Set# of 4 as specified by “RFRAME BPRF set”.

Set AoA / PDoA Enhancement Parameters

There are two parameters usable to improve AoA/PDoA performance as follows. These settings will need to be set at the device where the Local AoA will be collected. This is where the AoA antenna is used and will generally (but not always) be the initiator.

- **PDOAOFF** command sets the PDoA Offset value when the antenna is measuring at 0 deg. I.e. at the start of operation, (Regardless of PAVRG parameter settings). This compensates for part-to-part variation of the chip and small deltas in the antenna. To use this function, set the Tag perpendicular to the Node, i.e. at 0 degree at a distance greater than 1.5m.

Take care to ensure that the face of the AoA antenna is exactly perpendicular to the 0-deg position of the non-AoA antenna. One way to do this is to use a laser measuring device with a flat surface on the back. Place the flat surface against the face of the AoA antenna and align the laser with the opposite board. More characterization setup details are in the next section.

Start TWR by executing INITF/RESPF on relevant sides. It is recommended to allow for at least 500 samples to be outputted so that a good average can be calculated. The mean value should then be manually calculated using the “PDoA_deg” output.

```
{"Block":983,"Addr":"0x0000","Status":"Ok","D_cm":154,"PDoA_deg":144.39,"LAoA_deg":66.43, "LFoM
↪":235,"RAoA_deg":0,"CFO_ppm":5.49}
```

For example, if the mean of that value over 100-200 ranging blocks is “120”, execute “STOP”, “PDOAOFF 120” and then “SAVE” on the terminal of the device with the PDoA antenna. That command would instruct the device to apply an internal offset of 120 degree to PDoA measures. During next ranging sessions, the corresponded PDoA will be offset and AoA should be close to 0 deg.

```
pdoaoff 120
JS0013{"PDOAOFF_deg":120}
ok
save
ok
```

After applying PDoA offset, it will be adjusted in the Application:

```
{"Block":12,"Addr":"0x0000","Status":"Ok","D_cm":154,"PDoA_deg":9,"LAoA_deg":0.4,"LFoM":235,
↪"RAoA_deg":0,"CFO_ppm":5.39}
```

This should be confirmed before continuing testing.

Note: The PDoA offset value can vary across units, channel, antennas… etc. In production, once a configuration is fixed, i.e. hardware/uwbcfg/antenna, then PDoA offset needs to be measured only once in the factory.

- **PAVRG** command sets the number of measured points to apply PDoA averaging and filtering on, resulting in better AoA output results. The recommended value is **10**. Default value is 10.

```
pavrg 10
JS000E{"AVERAGE":10}
ok
save
ok
```

5.5.3 LISTENER2

PHY-level Listener application. This command will start a sniffer application based on the **UWBCFG** configuration.

Usage:

```
listener2

Listen for the UWB packets using the UWB configuration.
```

LSTAT command displays the statistics inside the LISTENER2 application. It needs to be fired while LISTENER2 application is running and it shows the following information:

Table 5.9: LSTAT details

Info	Description
CRCG	12-bit number of good CRC received frame events.
CRCB	12-bit number of bad CRC (CRC error) received frame events.
ARFE	8-bit number of address filter error events.
PHE	12-bit number of received header error events.
RSL	12-bit number of received frame sync loss event events.
SFDTO	12-bit number of SFD timeout events.
PTO	12-bit number of Preamble timeout events.
FTO	8-bit number of RX frame wait timeout events.
STSE	Counts the number of STS detections with bad STS quality.
STSE	Counts the number of STS detections with good STS quality.
SFDD	Counts the number of SFD detections (RXFR has to be set also).

Example:

Responder side: start listener by using default UWBCFG setting

```
listener2
ok
Found AOA DW3000 chip. PDoA is available.
Listener Top Application: Started
```

Initiator side: start INITF by using default setting

```
initf
JS010C{"F PARAMS":{
"SLOT, rstu":2400,
"Ranging Period, ms":200,
"Ranging round, slots":30,
"Session_ID":42,
"RFRAME":3,
"SFD ID":2,
"Multi node mode":0,
"Round hopping":0,
"Vupper64":01:02:03:04:05:06:07:08,
"Initiator Addr":0x0000,
"Responder[0] Addr":0x0001,
}
ok
```

Listener logs

Type “LISTENER2” to capture every possible logs while prioritizing speed, and limiting the amount of output data to 6 Bytes Type “LISTENER2 1” to dump all the data captured. This cannot run faster than every 1ms. Please note, if the buffer is full, some data packets may be skipped. Type “STOP” to halt.

Listener application output of SP0 packets (uwbcfg STS MODE parameter set to 0):

```
listener2
ok
Found AOA DW3000 chip. PDoA is available.
Listener Top Application: Started
JS00EF{"LSTN":[49,2B,01,00,26,13,00,FF,18,5A,08,08,08,08,08,08,08,08,2A,00,00,00,8C,2D,2F,0D,00,
↪3F,49,11,1E,68,56,31,46,A9,B1,06,BF,AF,4B,7C,BE,55,65,E8,A9,77,7A,3F,53,CF,25,6D,A4,48,F7,8B,
↪A8,56,0D,76,A0,DA,27,5F,58],"TS":"0xCE99FA8D","O":253}
JS00EF{"LSTN":[49,2B,01,00,26,13,00,FF,18,5A,08,08,08,08,08,08,08,08,2A,00,00,00,F0,2D,2F,0D,00,
↪3F,49,11,1E,68,56,31,46,A9,B1,06,BF,AF,4B,7C,BE,55,65,E8,A9,77,7A,3F,53,CF,25,6D,A4,48,F7,E8,
↪9B,C7,36,53,94,5D,24,F8,FD],"TS":"0xC8516B8E","O":277}
JS00EF{"LSTN":[49,2B,01,00,26,13,00,FF,18,5A,08,08,08,08,08,08,08,08,2A,00,00,00,54,2E,2F,0D,00,
↪3F,49,11,1E,68,56,31,46,A9,B1,06,BF,AF,4B,7C,BE,55,65,E8,A9,77,7A,3F,53,CF,25,6D,A4,48,F7,55,
↪9B,FF,C7,AE,49,15,40,9A,AB],"TS":"0xC208DC88","O":306}

stop
ok
```

5.5.4 UWB testing : TCFM / TCWM

5.5.4.1 TCFM

This command runs the Test Continuous Frame Mode.

Usage:

```
TCFM <NUM> <PAUSE> <LEN>

All parameters of this command are optional:

<NUM> : number of packets to transmit. Default value = 0: An infinite number of packets.
<PAUSE> : pause in between packets in ms. Default value = 1 ms.
<LEN> : length of the transmit payload, bytes. Default value = 20 bytes.
```

Note: The command with <PAUSE> set to 1 (i.e. 1ms) is widely used for the certification purpose when require to measure the Transmit power level of the tested unit.

Example:

```
TCFM
```

This would start the Test Continuous Frame Mode with an infinite number of packets, 1 ms pause in between packets and with 20 bytes transmit payload.

```
TCFM 1000
```

This would start the Test Continuous Frame Mode with 1000 packets, 1 ms pause in between packets and with 20 bytes transmit payload.

5.5.4.2 TCWM

This command would run the Test Continuous Wave Mode.

Usage:

```
TCWM

Test Continuous Wave mode generates the base frequency of the corresponded channel on the RF␣
↪port.
```

Note: This command can be used to check the appropriate reaction of the unit to the XTALTRIM command.

Example:

```
TCWM
```

Start continuous carrier wave transmission, typically used for crystal trim calibration and verification.

6 Universal Command Interface - UCI Build

6.1 Overview

In UCI build, UCI will be the only available applications and active by default.

For details about UCI protocol, see details here:

UCI[5] generic specification can be found in FiRa[6] website (membership is needed in order to download the document).

6.2 Interfaces

The SDK currently supports different interfaces for UCI commands.

6.2.1 Physical transport

- UART: all targets (115200,8N1, no flow control) or USB (CDC ACM class) on boards with USB peripheral.

6.2.2 Active interface selection

At boot time, all interfaces available on a board will be enabled in reception mode and are ready to handle a UCI command. Once a command is received on a transport peripheral, it will become the active interface and will be dedicated to UCI transactions.

Please follow *Board Connections* on how to connect those interfaces to your board.

6.3 UCI over UART python scripts

6.3.1 Overview

A Python script is provided to show how to do UCI FiRa ranging over UART or USB VCOM.

uci_uart_fira_test.py. Main entry point. It shows how to do FiRa range over USB/UART. It needs:

- **core.py**. UCI client implementation
- **v1_0.py**. FiRa session implementation

Note: Python 3.10 64bit should be used to run the example scripts.

Python requirements should also be installed before executing the scripts.

[5] https://groups.firaconsortium.org/wg/members/document/1949
[6] https://www.firaconsortium.org/

```
python -m pip install -r requirements.txt
```

6.3.2 Example

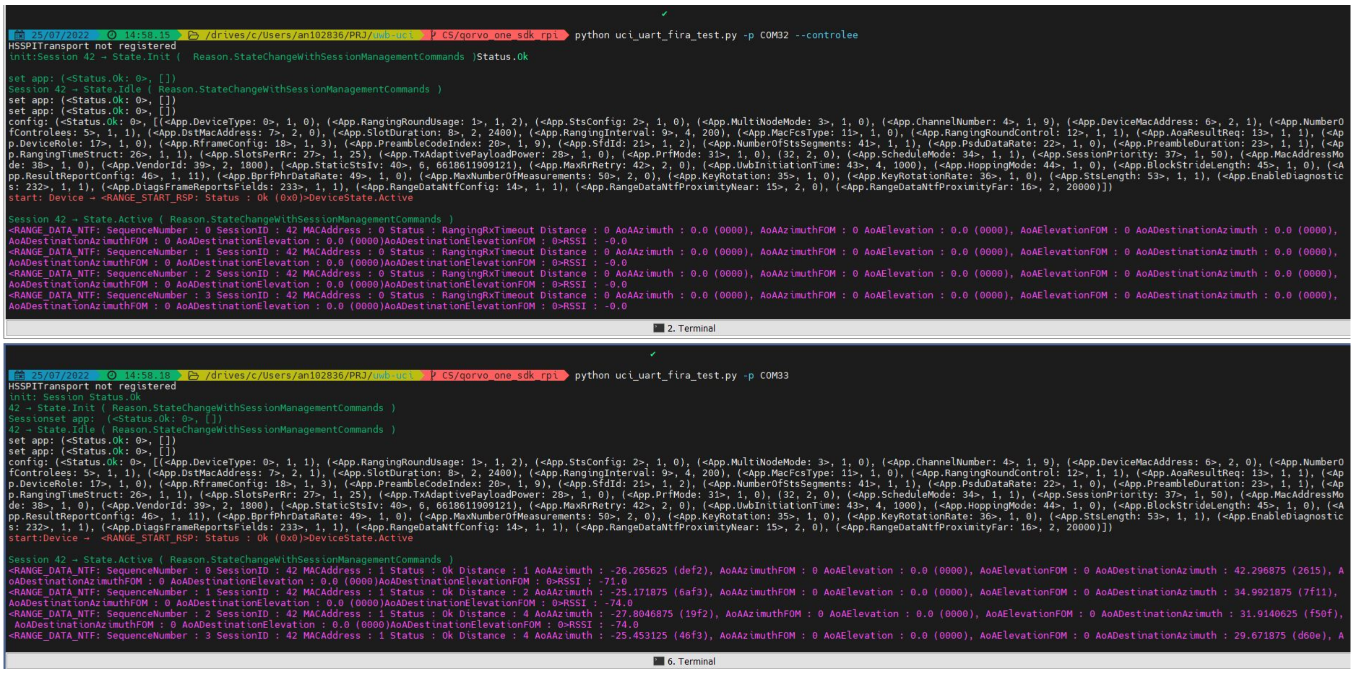
1. A python script (**uci_uart_fira_test.py**) is provided as an example on how to send/receive UCI commands using UART or USB VCOM. FiRa session parameters can be easily modified in the script.

   1. Open a first command shell in the python script location.
   2. Initialize a TWR FiRa session as controller on the first board (in this example, in COM18).

      ```
      python uci_uart_fira_test.py -p COM18
      ```

   3. Open a second command shell in the python script location.
   4. Initialize a TWR FiRa session as controlee on the second board (in this example, in COM19).

      ```
      python uci_uart_fira_test.py -p COM19 --controlee
      ```



```
 25/07/2022   14:58.15   /drives/c/Users/an102836/PRJ/uwb-uci   CS/qorvo_one_sdk_rpi  python uci_uart_fira_test.py -p COM32 --controlee
HSSPITransport not registered
init:Session 42 → State.Init (  Reason.StateChangeWithSessionManagementCommands )Status.Ok

set app: (<Status.Ok: 0>, [])
Session 42 → State.Idle ( Reason.StateChangeWithSessionManagementCommands )
set app: (<Status.Ok: 0>, [])
set app: (<Status.Ok: 0>, [])
config: (<Status.Ok: 0>, [(<App.DeviceType: 0>, 1, 0), (<App.RangingRoundUsage: 1>, 1, 2), (<App.StsConfig: 2>, 1, 0), (<App.MultiNodeMode: 3>, 1, 0), (<App.ChannelNumber: 4>, 1, 9), (<App.DeviceMacAddress: 6>, 2, 1), (<App.NumberO
fControlees: 5>, 1, 1), (<App.DstMacAddress: 7>, 2, 0), (<App.SlotDuration: 8>, 2, 2400), (<App.RangingInterval: 9>, 4, 200), (<App.MacFcsType: 11>, 1, 0), (<App.RangingRoundControl: 12>, 1, 1), (<App.AoaResultReq: 13>, 1, 1), (<Ap
p.DeviceRole: 17>, 1, 0), (<App.RframeConfig: 18>, 1, 3), (<App.PreambleCodeIndex: 20>, 1, 9), (<App.SfdId: 21>, 1, 2), (<App.NumberOfStsSegments: 41>, 1, 1), (<App.PsduDataRate: 22>, 1, 0), (<App.PreambleDuration: 23>, 1, 1), (<Ap
p.RangingTimeStruct: 26>, 1, 1), (<App.SlotsPerRr: 27>, 1, 25), (<App.TxAdaptivePayloadPower: 28>, 1, 0), (<App.PrfMode: 31>, 1, 0), (32, 2, 0), (<App.ScheduleMode: 34>, 1, 1), (<App.SessionPriority: 37>, 1, 50), (<App.MacAddressMo
de: 38>, 1, 0), (<App.VendorId: 39>, 2, 1800), (<App.StaticStsIv: 40>, 6, 6618611909121), (<App.MaxRrRetry: 42>, 2, 0), (<App.UwbInitiationTime: 43>, 4, 1000), (<App.HoppingMode: 44>, 1, 0), (<App.BlockStrideLength: 45>, 1, 0), (<A
pp.ResultReportConfig: 46>, 1, 11), (<App.BprfPhrDataRate: 49>, 1, 0), (<App.MaxNumberOfMeasurements: 50>, 2, 0), (<App.KeyRotation: 35>, 1, 0), (<App.KeyRotationRate: 36>, 1, 0), (<App.StsLength: 53>, 1, 1), (<App.EnableDiagnostic
s: 232>, 1, 1), (<App.DiagsFrameReportsFields: 233>, 1, 1), (<App.RangeDataNtfConfig: 14>, 1, 1), (<App.RangeDataNtfProximityNear: 15>, 2, 0), (<App.RangeDataNtfProximityFar: 16>, 2, 20000)])
start: Device → <RANGE_START_RSP: Status : Ok (0x0)>DeviceState.Active

Session 42 → State.Active ( Reason.StateChangeWithSessionManagementCommands )
<RANGE_DATA_NTF: SequenceNumber : 0 SessionID : 42 MACAddress : 0 Status : RangingRxTimeout Distance : 0 AoAAzimuth : 0.0 (0000), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 0.0 (0000),
AoADestinationAzimuthFOM : 0 AoADestinationElevation : 0.0 (0000)AoADestinationElevationFOM : 0>RSSI : -0.0
<RANGE_DATA_NTF: SequenceNumber : 1 SessionID : 42 MACAddress : 0 Status : RangingRxTimeout Distance : 0 AoAAzimuth : 0.0 (0000), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 0.0 (0000),
AoADestinationAzimuthFOM : 0 AoADestinationElevation : 0.0 (0000)AoADestinationElevationFOM : 0>RSSI : -0.0
<RANGE_DATA_NTF: SequenceNumber : 2 SessionID : 42 MACAddress : 0 Status : RangingRxTimeout Distance : 0 AoAAzimuth : 0.0 (0000), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 0.0 (0000),
AoADestinationAzimuthFOM : 0 AoADestinationElevation : 0.0 (0000)AoADestinationElevationFOM : 0>RSSI : -0.0
<RANGE_DATA_NTF: SequenceNumber : 3 SessionID : 42 MACAddress : 0 Status : RangingRxTimeout Distance : 0 AoAAzimuth : 0.0 (0000), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 0.0 (0000),
AoADestinationAzimuthFOM : 0 AoADestinationElevation : 0.0 (0000)AoADestinationElevationFOM : 0>RSSI : -0.0
2. Terminal
```

```
 25/07/2022   14:58.18   /drives/c/Users/an102836/PRJ/uwb-uci   CS/qorvo_one_sdk_rpi  python uci_uart_fira_test.py -p COM33
HSSPITransport not registered
init: Session Status.Ok
42 → State.Init ( Reason.StateChangeWithSessionManagementCommands )
Sessionset app:  (<Status.Ok: 0>, [])
42 → State.Idle ( Reason.StateChangeWithSessionManagementCommands )
set app: (<Status.Ok: 0>, [])
set app: (<Status.Ok: 0>, [])
config: (<Status.Ok: 0>, [(<App.DeviceType: 0>, 1, 1), (<App.RangingRoundUsage: 1>, 1, 2), (<App.StsConfig: 2>, 1, 0), (<App.MultiNodeMode: 3>, 1, 0), (<App.ChannelNumber: 4>, 1, 9), (<App.DeviceMacAddress: 6>, 2, 0), (<App.NumberO
fControlees: 5>, 1, 1), (<App.DstMacAddress: 7>, 2, 1), (<App.SlotDuration: 8>, 2, 2400), (<App.RangingInterval: 9>, 4, 200), (<App.MacFcsType: 11>, 1, 0), (<App.RangingRoundControl: 12>, 1, 1), (<App.AoaResultReq: 13>, 1, 1), (<Ap
p.DeviceRole: 17>, 1, 0), (<App.RframeConfig: 18>, 1, 3), (<App.PreambleCodeIndex: 20>, 1, 9), (<App.SfdId: 21>, 1, 2), (<App.NumberOfStsSegments: 41>, 1, 1), (<App.PsduDataRate: 22>, 1, 0), (<App.PreambleDuration: 23>, 1, 1), (<Ap
p.RangingTimeStruct: 26>, 1, 1), (<App.SlotsPerRr: 27>, 1, 25), (<App.TxAdaptivePayloadPower: 28>, 1, 0), (<App.PrfMode: 31>, 1, 0), (32, 2, 0), (<App.ScheduleMode: 34>, 1, 1), (<App.SessionPriority: 37>, 1, 50), (<App.MacAddressMo
de: 38>, 1, 0), (<App.VendorId: 39>, 2, 1800), (<App.StaticStsIv: 40>, 6, 6618611909121), (<App.MaxRrRetry: 42>, 2, 0), (<App.UwbInitiationTime: 43>, 4, 1000), (<App.HoppingMode: 44>, 1, 0), (<App.BlockStrideLength: 45>, 1, 0), (<A
pp.ResultReportConfig: 46>, 1, 11), (<App.BprfPhrDataRate: 49>, 1, 0), (<App.MaxNumberOfMeasurements: 50>, 2, 0), (<App.KeyRotation: 35>, 1, 0), (<App.KeyRotationRate: 36>, 1, 0), (<App.StsLength: 53>, 1, 1), (<App.EnableDiagnostic
s: 232>, 1, 1), (<App.DiagsFrameReportsFields: 233>, 1, 1), (<App.RangeDataNtfConfig: 14>, 1, 1), (<App.RangeDataNtfProximityNear: 15>, 2, 0), (<App.RangeDataNtfProximityFar: 16>, 2, 20000)])
start:Device →  <RANGE_START_RSP: Status : Ok (0x0)>DeviceState.Active

Session 42 → State.Active ( Reason.StateChangeWithSessionManagementCommands )
<RANGE_DATA_NTF: SequenceNumber : 0 SessionID : 42 MACAddress : 1 Status : Ok Distance : 1 AoAAzimuth : -26.265625 (def2), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 42.296875 (2615), A
oADestinationAzimuthFOM : 0 AoADestinationElevation : 0.0 (0000)AoADestinationElevationFOM : 0>RSSI : -71.0
<RANGE_DATA_NTF: SequenceNumber : 1 SessionID : 42 MACAddress : 1 Status : Ok Distance : 2 AoAAzimuth : -25.171875 (6af3), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 34.9921875 (7f11),
AoADestinationAzimuthFOM : 0 AoADestinationElevation : 0.0 (0000)AoADestinationElevationFOM : 0>RSSI : -74.0
<RANGE_DATA_NTF: SequenceNumber : 2 SessionID : 42 MACAddress : 1 Status : Ok Distance : 4 AoAAzimuth : -27.8046875 (19f2), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 31.9140625 (f50f),
 AoADestinationAzimuthFOM : 0 AoADestinationElevation : 0.0 (0000)AoADestinationElevationFOM : 0>RSSI : -74.0
<RANGE_DATA_NTF: SequenceNumber : 3 SessionID : 42 MACAddress : 1 Status : Ok Distance : 4 AoAAzimuth : -25.453125 (46f3), AoAAzimuthFOM : 0 AoAElevation : 0.0 (0000), AoAElevationFOM : 0 AoADestinationAzimuth : 29.671875 (d60e), A
6. Terminal
```

Fig. 6.1: Example of UCI over UART

6.4 UCI Core

6.4.1 Overview

The UCI core includes the following required core functionality:

- Packet formats to transmit Commands, Responses and Notification messages over UCI.
- Definitions of the Commands, Responses and Notifications are used for different operations between the Host and the UWBS device.
- A flow control mechanism for Command / Response exchange of messages.

- Segmentation and reassembly of all UCI messages.
- Initialization and configuration of the UWBS device.

For further detials, please refer to UCI[7] generic specification which can be found in FiRa[8] website (membership is needed in order to download the document).

6.4.2 UCI Control Messages

The Host uses UCI Control Messages to control and to configure the UWBS device. Control Message consists of Commands, Responses and Notifications. The Commands are only allowed to be sent in the direction from Host to UWBS, and Responses and Notifications are only allowed to be sent in the other direction. Control Messages are transmitted in UCI Control Packets; UCI supports segmentation of Control Messages into multiple Packets.

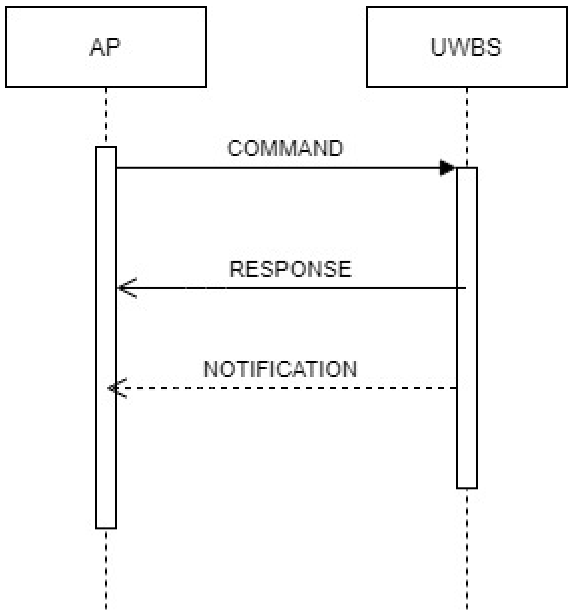


Fig. 6.2: Control Messages exchange.

A Command is sent by the Host to instruct the UWBS to perform a specific action. For each Command received, the UWBS answers with a Response to acknowledge the receipt of the Command. The response may also indicate the changes that the Command caused at the UWBS.

Notifications are only sent from the UWBS to the Host. A notification is sent to deliver additional information related to a Command. A notification is also sent independently of any Command or Response, unless specified otherwise.

The payload of the Control Messages is sent over the UCI Transport as a payload of Control Packets.

A Control Packet contains either a complete or a segment of a Control Message payload.

Both the Host and UWBS are capable of supporting Control Messages with a payload of MAX_PAYLOAD_SIZE octets, which is the maximum size of any Control Message payload.

As a result, Control Messages can be segmented into multiple Control Packets when sent over the UCI.

[7] https://groups.firaconsortium.org/wg/members/document/1949
[8] https://www.firaconsortium.org/

6.4.2.1 Flow Control for Control Messages

The Host and UWBS can send a complete Control Message over the UCI in as many packets as needed. There is no packet-based flow control for Control Messages in UCI.

The following flow control rules apply to Control Messages:

- After sending a Command, the Host does not send any command until it receives a Response for that command, or until it has taken steps to restore the capability to exchange Messages with the UWBS if it determines that too much time has elapsed waiting for a Response.
- After sending a Command, the Host is able to receive a Response.
- After sending a Response, the UWBS is ready to receive the next Command from the Host.
- The Host is able to receive Notifications from the UWBS at any time.

6.4.2.2 Exception Handling for Control Messages

The rules in this section define the exception processing to be performed by a receiver for an erroneous Control Message. Any command received by the Host is ignored.

Control Messages where the type of Control Message can be determined but containing errors are handled as follows:

- If the Control Message is a Command, the UWBS ignores the content of the Command and sends a Response with the same GID and OID field values as in the Command and with a Status value STATUS_SYNTAX_ERROR. The Response does not contain any additional fields.
- If the Control Message is a Response, the Host ignores the content of the Response and is free to send another Command.
- If the Control Message is a Notification, the Host ignores the Notification.

The UWBS responds to an unknown Command (unknown GID or OID) with a Response having the same GID and OID field values as the Command, followed by a Status field with the value of STATUS_UNKNOWN_GID/ STATUS_UNKNOWN_OID respectively and no additional fields.

The Host ignores any unknown Response or Notification (unknown GID or OID).

Additional octets at the end of a Control Message are ignored and not considered an error.

6.4.3 Packets Format

6.4.3.1 Common Packet Header

All packets have a common header, consisting of a Message Type (MT) field and a Packet Boundary Flag (PBF) field.

Table 6.1: UCI core packet format

octet	Bits	Field	Description
0	3	MT	Message type
	1	PBF	Packet Boundary Flag
	4		Information
1 to N	8 * N		Information

Message Type (MT) The MT field indicates the contents of the Packet and they are a 3-bit field containing one of the values listed in Table *MT Values*. The content of the Information field is dependent on the value of the MT field. The receiver of an MT designated as RFU silently discards the packet.

Table 6.2: MT Values

MT	Description
0b000	RFU
0b001	Control Packet - Command message as Information
0b010	Control Packet - Response message as Information
0b011	Control Packet - Notification message as Information
0b100 - 0b111	RFU

Packet Boundary Flag (PBF)

The Packet Boundary Flag (PBF) is used for segmentation and reassembly, and it is a 1-bit field containing one of the values listed in Table *PBF Values*.

Table 6.3: PBF Values

PBF	Description
0b0	The packet contains a complete message, or the Packet contains the last segment of the segmented message
0b1	The Packet contains a segment of a Message that is not the last segment

The following rules apply to the PBF flag in Packets:

- If the packet contains a complete Message, the PBF is set to 0b0.
- If the packet contains the last segment of a segmented Message, the PBF is set to 0b0.
- If the packet does not contain the last segment of a segmented Message, the PBF is set to 0b1.
- The segmentation and reassembly functionality is supported by both the Host and the UWBS.

The following rules apply to segmenting Control Messages:

- For each segment of a Control Message, the header of the Control Packet contains the same MT, GID and OID values.
- From Host to UWBS: the segmentation and reassembly feature is used when sending a Command Message from the Host to the UWBS that generates a Control Packet with a payload larger than MAX_PAYLOAD_SIZE. Each segment of a Command Message, except for the last one, contains a payload with the length of MAX_PAYLOAD_SIZE octets.
- From UWBS to Host: the segmentation and reassembly feature is used when sending a Response/Notification Message from the UWBS to the Host that generates a Control Packet with a payload larger than MAX_PAYLOAD_SIZE. Each segment of a Control Message, except for the last one, contains a payload with the length of MAX_PAYLOAD_SIZE octets.

6.4.3.2 Format of Control Packets

Table 6.4: UCI core control packet format

Octet	Bits	Field	Description
0	3	MT	Message type
	1	PBF	Packet Boundary Flag
	4	GID	Group Identifier
1	2	RFU	Reserved Future Use
	6	OID	Opcode Identifier
2	8	RFU	Reserved Future Use
3	8	L	Payload Length
4 to 4+L	8*L	L	Payload Length

Each Control Packet has 4-octets Packet Header and may have one or more payload octets for carrying a Control Message payload or a segment of a Control Message payload. In the case of an 'empty' Control Message, only the Packet Header is sent.

Message Type (MT) Refer to *MT Values* table for details of the MT field.

Packet Boundary Flag (PBF) Refer to *PBF Values* table for details of the PBF field.

Group Identifier (GID) The Group Identifier (GID) indicates the categorization of the message and shall be a 4-bit field containing one of the values listed in Table *Messages Identifiers*.

Opcode Identifier (OID) The Opcode Identifier (OID) indicates the identification of the Control Message and shall be a 6-bit field that is a unique identification of a set of Command, Response Messages within the group. OID values are defined along with the definition of the respective Control Messages described in Table *Messages Identifiers*.

Payload Length (L) The Payload Length indicates the number of octets present in the payload. The Payload Length field is an 8-bit field containing a value from 0 to MAX_PAYLOAD_SIZE.

6.4.3.3 UCI FiRa ranging session message flow

The following diagram shows a typical UCI FiRa ranging message flow.

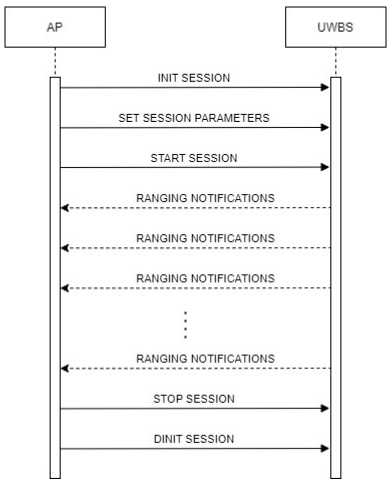


Fig. 6.3: Ranging session message flow

6.5 UCI Messages

6.5.1 Messages Identifiers

Table 6.5: GID and OID definitions

GID	OID	Message name	Description
UCI core 0x0	0x00	CORE_DEVICE_RESET	Reset device
	0x01	CORE_DEVICE_STATUS	Device status notification
	0x02	CORE_GET_DEVICE_INFO	Get device information
	0x03	CORE_GET_CAPS_INFO	Get device capabilities
	0x04	CORE_SET_CONFIG	Set device configuration
	0x05	CORE_GET_CONFIG	Get device configuration
	0x06	Reserved for future usage	
	0x07	CORE_GENERIC_ERROR	Generic error notification
	...	Reserved for future usage	
UWB session config group 0x1	0x00	SESSION_INIT	Request to create a UWB session
	0x01	SESSION_DEINIT	Request to destroy a UWB session
	0x02	SESSION_STATUS	Notifies the current UWB session state
	0x03	SESSION_SET_APP_CONFIG	Configuration request for UWB session with specified configurations
	0x04	SESSION_GET_APP_CONFIG	To get the current configuration for specified UWB session
	0x05	SESSION_GET_COUNT	To get the number of UWB session created in UWBS device
	0x06	SESSION_GET_STATE	To get the current state of the UWB session
	0x07	SESSION_UPDATE_CONTROLLER_	To update the multicast list
UWB session config group 0x2	0x00	RANGE_START	To activate the UWB ranging session.
	0x01	RANGE_STOP	To deactivate the ranging session
	0x02	RFU	RFU

6.5.2 UCI Core Group

6.5.2.1 Device Reset

This command resets the device.

It follows the behavior defined in FiRa UCI specification.

Table 6.6: CORE_DEVICE_RESET_CMD

Size (octet)	Field
1	Reset configuration, must be 1

Table 6.7: CORE_DEVICE_RESET_RSP

Size (octet)	Field
1	Status

6.5.2.2 Device Status Notification

This notification is sent when the device state is changed.

It follows the behavior defined in FiRa UCI specification.

Table 6.8: `CORE_DEVICE_STATUS_NTF`

Size (octet)	Field
1	Device state

Table 6.9: Device state values

State value	Device state
0x00	Reserved for future usage
0x01	`DEVICE_STATE_READY`: device is initialized and ready
0x02	`DEVICE_STATE_ACTIVE`: device is started
...	Reserved for future usage
0xff	`DEVICE_STATE_ERROR`: unrecoverable error state

Device is started when the first FiRa session is started, and stopped when all FiRa sessions are stopped.

6.5.2.3 Getting Device Information

This command follows the behavior defined in FiRa UCI specification.

The `CORE_GET_DEVICE_INFO_CMD` command has no data.

Table 6.10: `CORE_GET_DEVICE_INFO_RSP`

Size (octet)	Field
1	Status
2	FiRa UCI generic version
2	FiRa MAC version
2	FiRa PHY version
2	FiRa UCI test version
1	Vendor specific information length
n	Vendor specific information

When FiRa is not included in the device, FiRa MAC, PHY and UCI test versions are 0. FiRa UCI generic version is the version on which UCI transport is based.

Table 6.11: `CORE_GET_DEVICE_INFO_RSP` vendor specific information

Size (octet)	Field
4	Device ID
4	Part ID
3	Firmware Version
1	AoA capability (0:Error, 1:AoA, 2:non-AoA)

6.5.2.4 Getting Device Capabilities

This command follows the behavior defined in FiRa UCI specification.

The `CORE_GET_CAPS_INFO_CMD` command has no data.

Table 6.12: `CORE_GET_CAPS_INFO_RSP`

Size (octet)	Field
1	Status
1	Number of capability parameters
n	Capability parameters as TLV form • type: 1 octet • length: 1 octet • value: <length> octets

Table 6.13: `CORE_GET_CAPS_INFO_RSP` capabilities

Name	Type	Length	Description
`CHANNELS`	0xa0	4	Bit mask of supported channels
`PREAMBLE_CODES`	0xa1	8	Bit mask of supported preamble codes

6.5.2.5 Setting Device Configuration

This command follows the behavior defined in FiRa UCI specification, additional vendor parameters are defined.

Table 6.14: `CORE_SET_CONFIG_CMD`

Size (octet)	Field
1	Number of parameters
n	Parameters as TLV form • type: 1 octet • length: 1 octet, reset to default if 0 • value: <length> octets

Table 6.15: `CORE_SET_CONFIG_RSP`

Size (octet)	Field
1	Status
1	Number of parameters status
n	List of parameter status • type: 1 octet • status: 1 octet

Parameters are defined in *Device Configuration Parameters*.

6.5.2.6 Retrieving Device Configuration

This command follows the behavior defined in FiRa UCI specification, additional vendor parameters are defined.

Table 6.16: `CORE_GET_CONFIG_CMD`

Size (octet)	Field
1	Number of parameters
n	List of parameter types • type: 1 octet

Table 6.17: `CORE_GET_CONFIG_RSP`

Size (octet)	Field
1	Status
1	Number of parameters
n	Parameters as TLV form • type: 1 octet • length: 1 octet, 0 if omitted • value: <length> octets

Parameters are defined in *Device Configuration Parameters*.

6.5.2.7 Device Configuration Parameters

This is used with `CORE_SET_CONFIG` and `CORE_GET_CONFIG`.

Table 6.18: Device configuration parameters

Name	Type	Length	Default	Description
`DEVICE_STATE`	0x00	1	NA	State reported by `CORE_DEVICE_STATUS`, read only
`LOW_POWER_MODE`	0x01	1	1	Enable (1) or disable (0) low power mode, see FiRa UCI specification
...				Reserved for future usage
`CHANNEL`	0xa0	1	DS	The device default channel
`PREAMBLE_CODE`	0xa1	1	DS	The device default preamble code
`PAN_ID`	0xa2	2	0xffff	The identifier of the PAN on which the device is operating, if 0xffff, device is not associated
`SHORT_ADDR`	0xa3	2	0xffff	The address the device uses to communicate in the PAN
`EXTENDED_ADDR`	0xa4	8	DS	The extended address assigned to the device
`PAN_COORD`	0xa5	1	0	Device is a PAN coordinator if 1
`PROMISCUOUS`	0xa6	1	0	Enable (1) or disable (0) promiscuous mode
`FRAME_RETRIES`	0xa7	1	3	The maximum number of retries after a transmission failure
...				Reserved for future usage

NA = Not applicable. DS = Device specific.

Device configuration can be overridden by a region during its operation.

6.5.2.8 Generic Error

This notification follows the behavior defined in FiRa UCI specification.

Table 6.19: `CORE_GENERIC_ERROR_NTF`

Size (octet)	Field
1	Status

7 SDK Architecture

7.1 Overview

The SDK platform supports FreeRTOS and Zephyr and ARM Cortex M4. The sections below discuss the architecture, structure and workflow of the software. This should enable the developer to understand the philosophy and be able to add functionality or port the project to other platforms if needed.

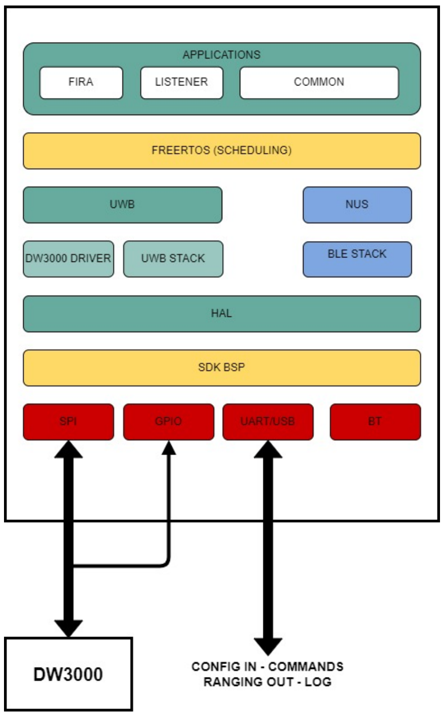


Fig. 7.1: Overview of FreeRTOS based architecture.

The figure above shows the layered structure: “Top-level applications”, “FreeRTOS”, “UWB Stack”, “HAL” and “SDK BSP”.

7.1.1 Applications layer

It contains top level applications of the SDK software:

- INITF / RESPF
- TCFM / TCWM
- LISTENER2
- UCI

They are described in detail in Applications folder section. Top-level applications cannot run concurrently since they use the same resources.

7.1.2 UWB layer

This layer and its sublayers (drivers and MAC) are responsible for linking the top-level applications with the radio physical layer.

7.1.3 HAL layer

HAL is in charge of abstracting the target hardware where the SDK is running.

7.1.4 SDK BSP layer

SDK BSP contains the Board Support Package for the different targets supported in the SDK.

7.1.5 OSAL layer

OSAL is in charge of abstracting the OS used.

7.2 Folder structure

The table below shows how the source code is distributed among the different folders.

Table 7.1: SDK folder structure

Folder name	Folder description
Libs	Libraries used for MAC layer and chip driver layer.
Projects	It contains the different projects grouped by OS and target.
SDK_BSP	Board support package grouped by manufacturer.
Src	Source code folders

7.3 Folders content

7.3.1 Libs Folder

7.3.1.1 UWB driver library

Chip driver is provided as a library in **dwt_uwb_driver** folder:

- Inc: driver APIs header files.
- lib: driver APIs library files.

7.3.1.2 UWB stack library

UWB MAC layer is provided as library in **uwbstack** folder:

- cmsis_v1
- cmsis_v2
- uwbstack_lib: MAC library header files and library files.

7.3.2 Projects folder

Projects folder contains the SDK project files, splitted according to the OS used:

Table 7.2: List of targets/OS supported in the SDK

Target	FreeRTOS	Zephyr
DWM3001CDK	Yes	No

7.3.3 SDK BSP Folder

SDK BSP folder includes the Board Support Package (drivers) needed for the different targets supported.

7.3.4 AppConfig Folder

This folder contains what is needed to store non volatile configuration into Non Volatile Memory (NVM).

The way to save data to NVM is both Board dependent (what flash memory is used) and OS dependent (storage machanism is/is not handled by OS).

For this reason, we implemented this as stand alone folder to offer the required flexibility.

A user can then adapt the code to match his requirements

The keyfile is located inside *AppConfig/{MyBoard}/{MyOS}/config.c*

Please also see *NVM configuration* section on NVM configuration for further details.

7.3.4.1 API

The API header file is located inside *AppConfig/Inc/appConfig.h*

It contains 3 functions :

```
void load_bssConfig(void)
```

This function will load config from FLASH to RAM

```
void restore_bssConfig(void)
```

This function will restore default config to RAM and save it back to FLASH

```
error_e save_bssConfig(void)
```

This function will save the RAM config to FLASH

7.3.4.2 Linker symbols used

```
extern uint8_t __config_entry_start[];
extern uint8_t __config_entry_end[];
extern uint8_t __fconfig_start[];
extern uint8_t __rconfig_start[];
extern uint8_t __rconfig_end[];
extern uint8_t __rconfig_crc_end[];
```

Please also see *Linker sections*.

7.3.5 Apps Folder

7.3.5.1 common

This folder contains everything needed to manage available applications, CLI commands and the *Always running Threads*

7.3.5.2 fira : INITF / RESPF

INITF (Initiator FiRa) is an application that sets the unit as initiator in a FiRa ranging session.

RESPF (Responder FiRa) is an application that sets the unit as responder in a FiRa ranging session.

7.3.5.3 listener2 : LISTENER2

LISTENER2 is an application that sets the unit into receive mode and it reports any received data.

7.3.5.4 tcfm_tcwm : TCFM / TCWM

TCFM (Test Continuous Frame Mode) is an application in which frames are continuously transmitted.

TCWM (Test Continuous Wave Mode) is an application in which a UWB wave is continuously transmitted.

7.3.5.5 uci : UCI

UCI is the default application running when using the UCI build.

7.3.6 Boards Folder

Boards folder contains a sub-folder for each target supported in the SDK. Two files are used to customize a particular target:

<target_name>.c - code used during startup for target's initialization (board, peripherals, interfaces)

custom_board.h - header file describing target's pinout

rf_tuning_config.c - code to change the pulse shape between standard and alternate pulse shape and default values (Antenna delay, Offsets, Antenna, Tx Power, Crystal Trim, averaging, PG Delay)

7.3.7 HAL Folder

HAL (Hardware Abstraction Layer) folder provides all the services needed by the application layer so the access to hardware resources is agnostic.

It provides the following services:

- BT UART
- BUTTON
- ERROR
- GPIO
- LED
- RTC
- SLEEP
- SPI
- TIMER
- UART
- USB
- UWB

- WATCHDOG

7.3.8 OSAL Folder

OSAL folder contains the source code to abstract the OS used in a project:

- Dynamic memory (calloc, malloc, free)
- Critical section access (enter/leave)
- OS task / signal management

7.3.9 UWB Folder

UWB folder contains the source code to :

- Interact with the MAC
- Manage chip states
- Control some chip registers (Crystal trimming)
- Determine RF parameters (Lambda, distance between antennas for AoA, LUT)
- Convert PDoA in AoA

7.4 Linker sections

The SDK relies on linker sections to automatically register applications, drivers, NVM configurations, etc… This section defines how to modify your linker section to handle this. The linker description file (ld file) is located in your project folder. See section *Projects folder* for details.

7.4.1 Flash sections

7.4.1.1 ISR Vectors section

This section contains the traditional ISR vector table. It starts with a vector array defined with symbol .isr_vector. It is usually located at the Flash entry address.

7.4.1.2 NVM section

This section is a placeholder for storing configuration which will not be lost between 2 power cycles.

It starts with symbol `__fconfig_start` and ends with symbol `__fconfig_end`.

It is mandatory to ensure that the `__fconfig_start` is aligned at the start of a flash page and that a full page is reserved for this purpose.

Only one variable will be explicitely placed in this section, defined like this:

```
__attribute__((section(".fconfig"))) const uint32_t DummyConfig[1]  = {0xDEADBEEF};
```

Its role is to guaranty that the checksum will be incorrect at 1st powerup after download to force a factory default to take place.

See section *NVM configuration* for details about NVM save and restore machanism.

7.4.1.3 UWB Drivers section

This section is only valid for DW3 QM33 based projects.

It is a placeholder for storing the list of possible DW3 QM33 drivers.

It starts with symbol `__dw_drivers_start` and ends with symbol `__dw_drivers_end`.

Drivers from the lib dwt_uwb_driver are defining sections `.dw_drivers`.

During the driver probing phase, the function will browse this section through all the register drivers.

It will determine which one is appropriate for the version of DW3 QM33 connected to the main CPU.

7.4.1.4 Applications and Commands section

This series of section will determine which applications are available for the current build.

They are placed in a way that the FW browses them and displays them in the correct order when executing HELP command.

See section *HELP* for details.

7.4.1.5 Applications section

This series of sections will determine which applications are available for the current build.

7.4.1.6 Applications NVM Configuration section

This series of sections will determine which applications are available for the current build.

The 1st sub-section will handle all the commands which are allowed to execute at anytime.

It starts with symbol `__known_commands_start`.

It contains default function defined with symbol `.known_commands_anytime`.

The 2nd sub-section will handle all the commands which are considered as Applications.

It starts with symbol `__known_commands_app_start`.

It contains default function defined with symbol `.known_commands_app`.

The 3rd sub-section will handle all the sub-commands of individual applications.

It starts with symbol `__known_commands_app_start`.

It contains default function defined with symbol `.known_app_subcommands`.

The 4th sub-section will handle all the commands that can only bu ran if we are in idle state (meaning, no application running).

It starts with symbol `__known_commands_ilde_start`.

It contains default function defined with symbol `.known_commands_ilde`.

The 5th sub-section will handle all the commands considered service commands.

It starts with symbol `__known_commands_service_start`.

It contains default function defined with symbol `.known_commands_service`.

Lastly, the symbol `__known_commands_end` will mark the end of this section.

7.4.1.7 Restoring default Configuration section

This section is a placeholder for storing configuration default functions.

It starts with symbol `__config_entry_start` and ends with symbol `__config_entry_end`.

It contains default function defined with symbol `.config_entry`.

When the user will call RESTORE command (see *RESTORE*) or if the Configuration CRC is invalid, then a factory default will be called.

To do so, each functions listed in this section will be executed.

For example, for FiRa application, the function below will be executed.

```
static void restore_fira_default_config(void)
{
    memcpy(&fira_config_ram, &fira_config_flash_default, sizeof(fira_config_ram));
}

__attribute__((section(".config_entry"))) const void (*p_restore_fira_default_config)(void) =␣
↪(const void *)&restore_fira_default_config;
```

7.4.2 RAM sections

7.4.2.1 Applications NVM Configuration section

This section is a placeholder for every parameter an Application wants to store into NVM.

It starts with symbol `__rconfig_start` and ends with symbol `__rconfig_end`.

It is immediately followed by a CRC.

The end symbol for RAM Config + CRC block is `__rconfig_crc_end`

Variables and structures are defined using sections `.rconfig`.

For example, the FiRa Application will store its NVM parameter like this:

```
static fira_param_t fira_config_ram __attribute__((section(".rconfig"))) = {0};
```

See section *NVM configuration* for details about NVM save and restore machanism.

7.5 NVM configuration

7.5.1 Overview

- The way the configuration is handled inside the SDK is done through linker sections.
- All the variable that have to be stored in NVM and kept between power cycles are stored in a consecutive RAM section.
- This RAM section is ended with a CRC16.
- This RAM section is copied as a unique block inside NVM.
- At Power-up, the configuration is loaded from NVM and copied inside this RAM section.
- Any command will update the RAM version of the configuration. The RAM version is considered as volatile.
- Sending the SAVE command will save by copy the RAM configuration block into Flash.

- Sending RESTORE command will restore default configuration.

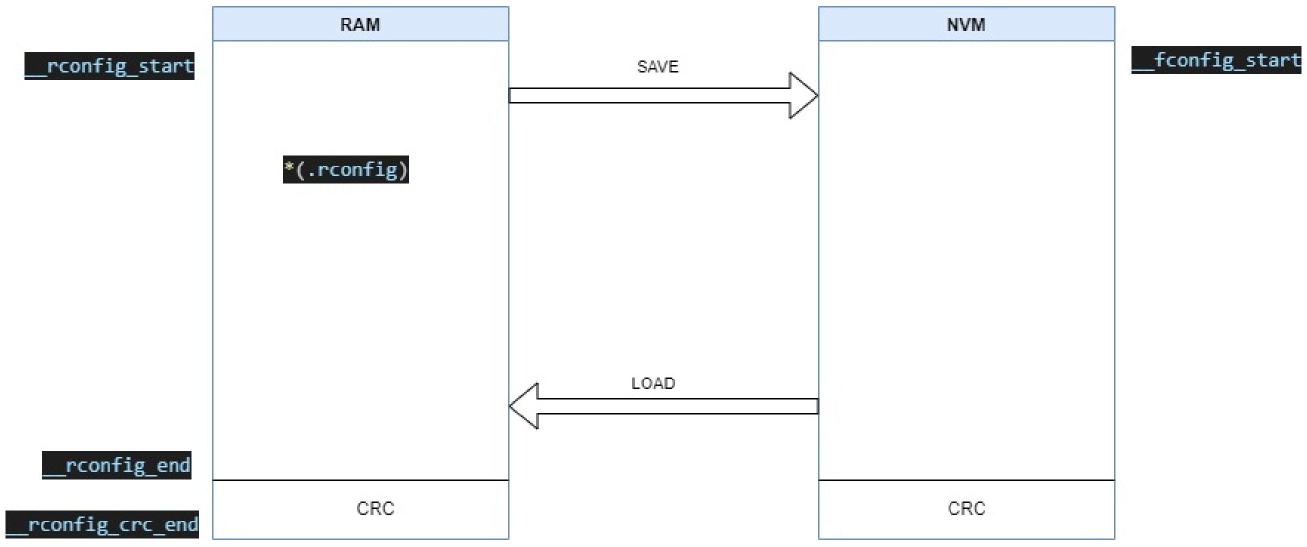


Fig. 7.2: NVM configuration

7.5.2 Initialization sequence during powerup

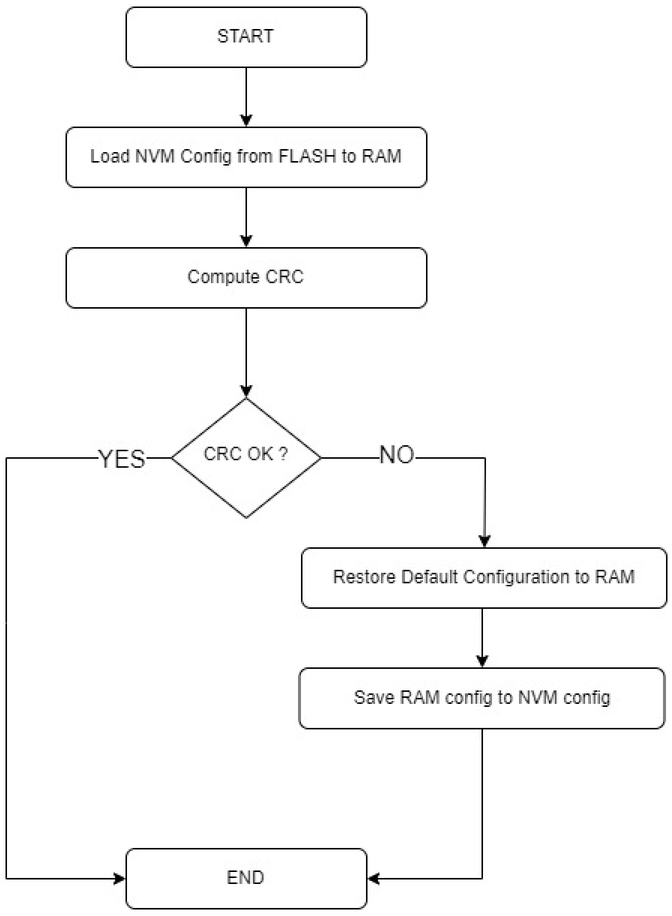

START
Load NVM Config from FLASH to RAM
Compute CRC
CRC OK ?
YES
NO
Restore Default Configuration to RAM
Save RAM config to NVM config
END

Fig. 7.3: Initialization sequence

8 SDK Runtime

8.1 Always running Threads

Once RTOS kernel has been started, the following threads are always running:

- Default Task
- Ctrl Task
- Flush Task

8.1.1 Default Task

Default Task is responsible for starting one of the top-level applications. It receives events from the Control task to switch to a particular mode of operation and starts corresponding top-level application.

Default task waits for a global event from EventManager, with a MESSAGE_QUEUE_TIMEOUT_MS (200ms) time-out.

- If an event is received within the timeout, Default task stops all running top-level tasks and it starts the requested top-level application.
- If no event is received within the timeout, Default task executes the USB_VBUS driver, which checks whether the VBUS pin of the MCU is attached to a power source or not.

This task is executed endlessly.

8.1.2 CtrlTask

Control Task is responsible for reception and execution of commands from external IO interfaces USB and/or UART.

On reception of a valid command by the Control Task, the command is processed and the corresponding reply is sent for output by triggering Flush task.

8.1.3 FlushTask

Flush task is responsible for transmitting any output data to the external I/O interfaces USB and/or UART. Any functions (including ISR functions) or tasks can request to send data to the non-blocking output (USB and/or UART).

When **port_tx_msg** function is called, the data is copied to an intermediate Report Buffer, which is a statically allocated circular buffer. The size of Report buffer is sufficient that any task/function can send a chunk of data for background output without delaying its throughput, even during an ISR.

The Flush task is then emptying the Report buffer onto the USB and/or the UART.

8.2 Application specific Threads

8.2.1 McpsTask

Mcps Task is used to handle the transmission and reception of data between the application layer and the MAC layer.

8.2.2 ReportTask

Report Task is used to handle the data reporting between from the MAC layer to the application layer.

8.3 Threads stack usage

8.3.1 FreeRTOS

The table below shows the threads stack usage of the tasks used in FreeRTOS in two different cases:

- Idle state (no applications running)

THREAD NAME	Stack usage
ctrlTask	396/2048
IDLE	164/512
defaultTask	148/4304
flushtask	184/512
Tmr Svc	156/512
Total HEAP	51200
Current HEAP used	9744
Max HEAP used	9936

- INIT application is running

THREAD NAME	Stack usage
ctrlTask	496/2048
IDLE	164/512
defaultTask	2668/4304
flushtask	184/512
reportTask	2440/4096
Tmr Svc	156/512
mcpsTask	1088/1600
Total HEAP	51200
Current HEAP used	27472
Max HEAP used	29568

9 Building SDK Firmware

9.1 nRF52840DK - Type 2AB EVB - DWM3001CDK

1. Install Segger Embedded Studio[9].

   A free SES license to use with Nordic products can be obtained here[10]

2. Open **.emProject** file provided in the package (e.g. Projects/DW3_QM33_SDK/FreeRTOS/nRF52840DK/ses/nRF52840D
   DW3_QM33_SDK_CLI-FreeRTOS.emProject)

3. From top level menu, select Build - Project name (e.g. **nRF52840DK-DW3_QM33_SDK_CLI-FreeRTOS**)

[9] https://www.segger.com/downloads/embedded-studio/

[10] https://wiki.segger.com/Get_a_License_for_Nordic_Semiconductor_Devices

10 Customizing the firmware for your Project

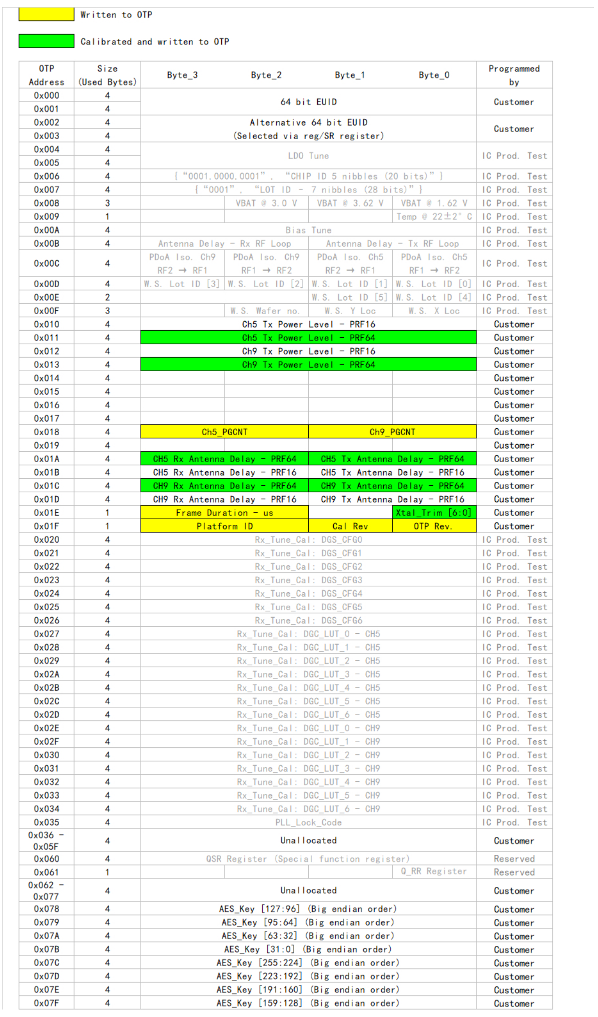
This chapter will be filled out in a future release.

10.1 OTP Memory Map

OTP memory map contains values calibrated after production.

The OTP memory locations are defined in the following tables. The OTP memory locations are each 32-bits wide; OTP addresses are word addresses, so each increment of address specifies a different 32-bit word. Memory allocation depends on the OTP Revision written at the address 0x1F.

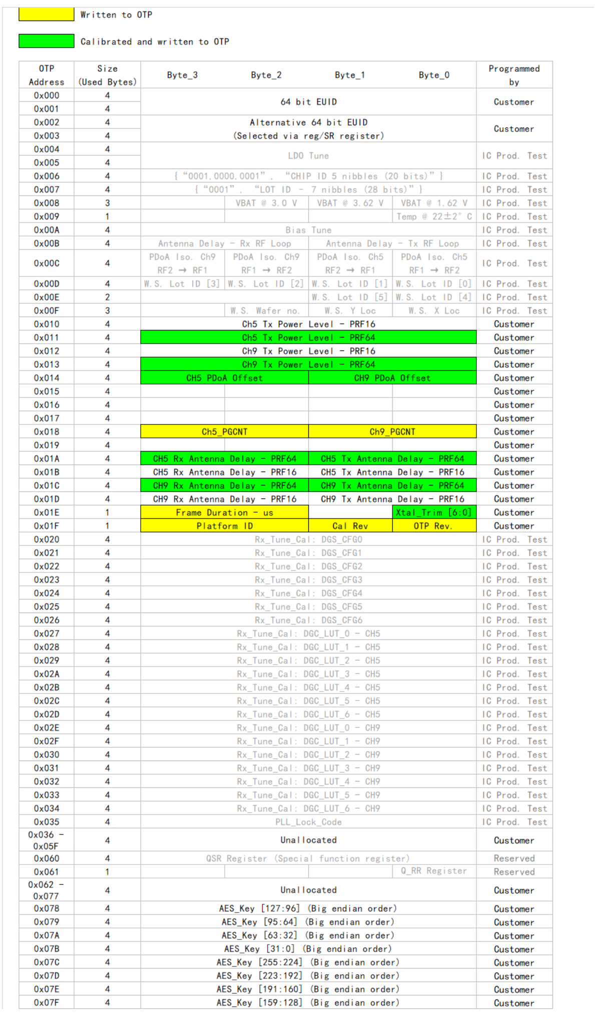
10.1.1 OTP Revision 1



🟨	Written to OTP
🟩	Calibrated and written to OTP

OTP Address	Size (Used Bytes)	Byte_3	Byte_2	Byte_1	Byte_0	Programmed by
0x000	4	64 bit EUID				Customer
0x001	4					
0x002	4	Alternative 64 bit EUID (Selected via reg/SR register)				Customer
0x003	4					
0x004	4	LDO Tune				IC Prod. Test
0x005	4					
0x006	4	{ "0001.0000.0001" , "CHIP ID 5 nibbles (20 bits)" }				IC Prod. Test
0x007	4	{ "0001" , "LOT ID – 7 nibbles (28 bits)" }				IC Prod. Test
0x008	3		VBAT @ 3.0 V	VBAT @ 3.62 V	VBAT @ 1.62 V	IC Prod. Test
0x009	1				Temp @ 22±2° C	IC Prod. Test
0x00A	4	Bias Tune				IC Prod. Test
0x00B	4	Antenna Delay – Rx RF Loop		Antenna Delay – Tx RF Loop		IC Prod. Test
0x00C	4	PDoA Iso. Ch9 RF2 → RF1	PDoA Iso. Ch9 RF1 → RF2	PDoA Iso. Ch5 RF2 → RF1	PDoA Iso. Ch5 RF1 → RF2	IC Prod. Test
0x00D	4	W.S. Lot ID [3]	W.S. Lot ID [2]	W.S. Lot ID [1]	W.S. Lot ID [0]	IC Prod. Test
0x00E	2			W.S. Lot ID [5]	W.S. Lot ID [4]	IC Prod. Test
0x00F	3		W.S. Wafer no.	W.S. Y Loc	W.S. X Loc	IC Prod. Test
0x010	4	Ch5 Tx Power Level – PRF16				Customer
0x011	4	Ch5 Tx Power Level – PRF64				Customer
0x012	4	Ch9 Tx Power Level – PRF16				Customer
0x013	4	Ch9 Tx Power Level – PRF64				Customer
0x014	4					Customer
0x015	4					Customer
0x016	4					Customer
0x017	4					Customer
0x018	4	Ch5_PGCNT		Ch9_PGCNT		Customer
0x019	4					Customer
0x01A	4	CH5 Rx Antenna Delay – PRF64		CH5 Tx Antenna Delay – PRF64		Customer
0x01B	4	CH5 Rx Antenna Delay – PRF16		CH5 Tx Antenna Delay – PRF16		Customer
0x01C	4	CH9 Rx Antenna Delay – PRF64		CH9 Tx Antenna Delay – PRF64		Customer
0x01D	4	CH9 Rx Antenna Delay – PRF16		CH9 Tx Antenna Delay – PRF16		Customer
0x01E	1	Frame Duration – us			Xtal_Trim [6:0]	Customer
0x01F	1	Platform ID		Cal Rev	OTP Rev.	Customer
0x020	4	Rx_Tune_Cal: DGS_CFG0				IC Prod. Test
0x021	4	Rx_Tune_Cal: DGS_CFG1				IC Prod. Test
0x022	4	Rx_Tune_Cal: DGS_CFG2				IC Prod. Test
0x023	4	Rx_Tune_Cal: DGS_CFG3				IC Prod. Test
0x024	4	Rx_Tune_Cal: DGS_CFG4				IC Prod. Test
0x025	4	Rx_Tune_Cal: DGS_CFG5				IC Prod. Test
0x026	4	Rx_Tune_Cal: DGS_CFG6				IC Prod. Test
0x027	4	Rx_Tune_Cal: DGC_LUT_0 – CH5				IC Prod. Test
0x028	4	Rx_Tune_Cal: DGC_LUT_1 – CH5				IC Prod. Test
0x029	4	Rx_Tune_Cal: DGC_LUT_2 – CH5				IC Prod. Test
0x02A	4	Rx_Tune_Cal: DGC_LUT_3 – CH5				IC Prod. Test
0x02B	4	Rx_Tune_Cal: DGC_LUT_4 – CH5				IC Prod. Test
0x02C	4	Rx_Tune_Cal: DGC_LUT_5 – CH5				IC Prod. Test
0x02D	4	Rx_Tune_Cal: DGC_LUT_6 – CH5				IC Prod. Test
0x02E	4	Rx_Tune_Cal: DGC_LUT_0 – CH9				IC Prod. Test
0x02F	4	Rx_Tune_Cal: DGC_LUT_1 – CH9				IC Prod. Test
0x030	4	Rx_Tune_Cal: DGC_LUT_2 – CH9				IC Prod. Test
0x031	4	Rx_Tune_Cal: DGC_LUT_3 – CH9				IC Prod. Test
0x032	4	Rx_Tune_Cal: DGC_LUT_4 – CH9				IC Prod. Test
0x033	4	Rx_Tune_Cal: DGC_LUT_5 – CH9				IC Prod. Test
0x034	4	Rx_Tune_Cal: DGC_LUT_6 – CH9				IC Prod. Test
0x035	4	PLL_Lock_Code				IC Prod. Test
0x036 – 0x05F	4	Unallocated				Customer
0x060	4	QSR Register (Special function register)				Reserved
0x061	1				Q_RR Register	Reserved
0x062 – 0x077	4	Unallocated				Customer
0x078	4	AES_Key [127:96] (Big endian order)				Customer
0x079	4	AES_Key [95:64] (Big endian order)				Customer
0x07A	4	AES_Key [63:32] (Big endian order)				Customer
0x07B	4	AES_Key [31:0] (Big endian order)				Customer
0x07C	4	AES_Key [255:224] (Big endian order)				Customer
0x07D	4	AES_Key [223:192] (Big endian order)				Customer
0x07E	4	AES_Key [191:160] (Big endian order)				Customer
0x07F	4	AES_Key [159:128] (Big endian order)				Customer

10.1.2 OTP Revision 2



🟨 Written to OTP

🟩 Calibrated and written to OTP

OTP Address	Size (Used Bytes)	Byte_3	Byte_2	Byte_1	Byte_0	Programmed by
0x000	4	64 bit EUID				Customer
0x001	4					
0x002	4	Alternative 64 bit EUID (Selected via reg/SR register)				Customer
0x003	4					
0x004	4	LDO Tune				IC Prod. Test
0x005	4					
0x006	4	{ "0001,0000,0001" , "CHIP ID 5 nibbles (20 bits)" }				IC Prod. Test
0x007	4	{ "0001" , "LOT ID - 7 nibbles (28 bits)" }				IC Prod. Test
0x008	3		VBAT @ 3.0 V	VBAT @ 3.62 V	VBAT @ 1.62 V	IC Prod. Test
0x009	1				Temp @ 22±2° C	IC Prod. Test
0x00A	4	Bias Tune				IC Prod. Test
0x00B	4	Antenna Delay - Rx RF Loop		Antenna Delay - Tx RF Loop		IC Prod. Test
0x00C	4	PDoA Iso. Ch9 RF2 → RF1	PDoA Iso. Ch9 RF1 → RF2	PDoA Iso. Ch5 RF2 → RF1	PDoA Iso. Ch5 RF1 → RF2	IC Prod. Test
0x00D	4	W.S. Lot ID [3]	W.S. Lot ID [2]	W.S. Lot ID [1]	W.S. Lot ID [0]	IC Prod. Test
0x00E	2			W.S. Lot ID [5]	W.S. Lot ID [4]	IC Prod. Test
0x00F	3		W.S. Wafer no.	W.S. Y Loc	W.S. X Loc	IC Prod. Test
0x010	4	Ch5 Tx Power Level - PRF16				Customer
0x011	4	🟩 Ch5 Tx Power Level - PRF64				Customer
0x012	4	Ch9 Tx Power Level - PRF16				Customer
0x013	4	🟩 Ch9 Tx Power Level - PRF64				Customer
0x014	4	🟩 CH5 PDoA Offset		🟩 CH9 PDoA Offset		Customer
0x015	4					Customer
0x016	4					Customer
0x017	4					Customer
0x018	4	🟨 Ch5_PGCNT		🟨 Ch9_PGCNT		Customer
0x019	4					Customer
0x01A	4	🟩 CH5 Rx Antenna Delay - PRF64		🟩 CH5 Tx Antenna Delay - PRF64		Customer
0x01B	4	CH5 Rx Antenna Delay - PRF16		CH5 Tx Antenna Delay - PRF16		Customer
0x01C	4	🟩 CH9 Rx Antenna Delay - PRF64		🟩 CH9 Tx Antenna Delay - PRF64		Customer
0x01D	4	CH9 Rx Antenna Delay - PRF16		CH9 Tx Antenna Delay - PRF16		Customer
0x01E	1	🟨 Frame Duration - us			🟩 Xtal_Trim [6:0]	Customer
0x01F	1	🟨 Platform ID		🟨 Cal Rev	🟨 OTP Rev.	Customer
0x020	4	Rx_Tune_Cal: DGS_CFG0				IC Prod. Test
0x021	4	Rx_Tune_Cal: DGS_CFG1				IC Prod. Test
0x022	4	Rx_Tune_Cal: DGS_CFG2				IC Prod. Test
0x023	4	Rx_Tune_Cal: DGS_CFG3				IC Prod. Test
0x024	4	Rx_Tune_Cal: DGS_CFG4				IC Prod. Test
0x025	4	Rx_Tune_Cal: DGS_CFG5				IC Prod. Test
0x026	4	Rx_Tune_Cal: DGS_CFG6				IC Prod. Test
0x027	4	Rx_Tune_Cal: DGC_LUT_0 - CH5				IC Prod. Test
0x028	4	Rx_Tune_Cal: DGC_LUT_1 - CH5				IC Prod. Test
0x029	4	Rx_Tune_Cal: DGC_LUT_2 - CH5				IC Prod. Test
0x02A	4	Rx_Tune_Cal: DGC_LUT_3 - CH5				IC Prod. Test
0x02B	4	Rx_Tune_Cal: DGC_LUT_4 - CH5				IC Prod. Test
0x02C	4	Rx_Tune_Cal: DGC_LUT_5 - CH5				IC Prod. Test
0x02D	4	Rx_Tune_Cal: DGC_LUT_6 - CH5				IC Prod. Test
0x02E	4	Rx_Tune_Cal: DGC_LUT_0 - CH9				IC Prod. Test
0x02F	4	Rx_Tune_Cal: DGC_LUT_1 - CH9				IC Prod. Test
0x030	4	Rx_Tune_Cal: DGC_LUT_2 - CH9				IC Prod. Test
0x031	4	Rx_Tune_Cal: DGC_LUT_3 - CH9				IC Prod. Test
0x032	4	Rx_Tune_Cal: DGC_LUT_4 - CH9				IC Prod. Test
0x033	4	Rx_Tune_Cal: DGC_LUT_5 - CH9				IC Prod. Test
0x034	4	Rx_Tune_Cal: DGC_LUT_6 - CH9				IC Prod. Test
0x035	4	PLL_Lock_Code				IC Prod. Test
0x036 - 0x05F	4	Unallocated				Customer
0x060	4	QSR Register (Special function register)				Reserved
0x061	1				Q_RR Register	Reserved
0x062 - 0x077	4	Unallocated				Customer
0x078	4	AES_Key [127:96] (Big endian order)				Customer
0x079	4	AES_Key [95:64] (Big endian order)				Customer
0x07A	4	AES_Key [63:32] (Big endian order)				Customer
0x07B	4	AES_Key [31:0] (Big endian order)				Customer
0x07C	4	AES_Key [255:224] (Big endian order)				Customer
0x07D	4	AES_Key [223:192] (Big endian order)				Customer
0x07E	4	AES_Key [191:160] (Big endian order)				Customer
0x07F	4	AES_Key [159:128] (Big endian order)				Customer

10.2 Adapting to your board

This chapter will be filled out in a future release.

10.3 Select which Apps you need

This chapter will be filled out in a future release.

11 Licenses in this SDK

Read the header of each file to know more about the license concerning this file.

Following licenses are used in the SDK :

- Apache-2.0 license
- Qorvo license
- FreeRTOS license
- Nordic Semiconductor ASA license
- Garmin Canada license
- ARM Limited license
- Third-party licenses: All third-party code contained in SDK_BSP/external (respective licenses included in each of the imported projects)

The provided HEX files were compiled using the projects located in the folders. For license and copyright information, see the individual .c and .h files that are included in the projects.