RT58x Thread SDK

User Guide

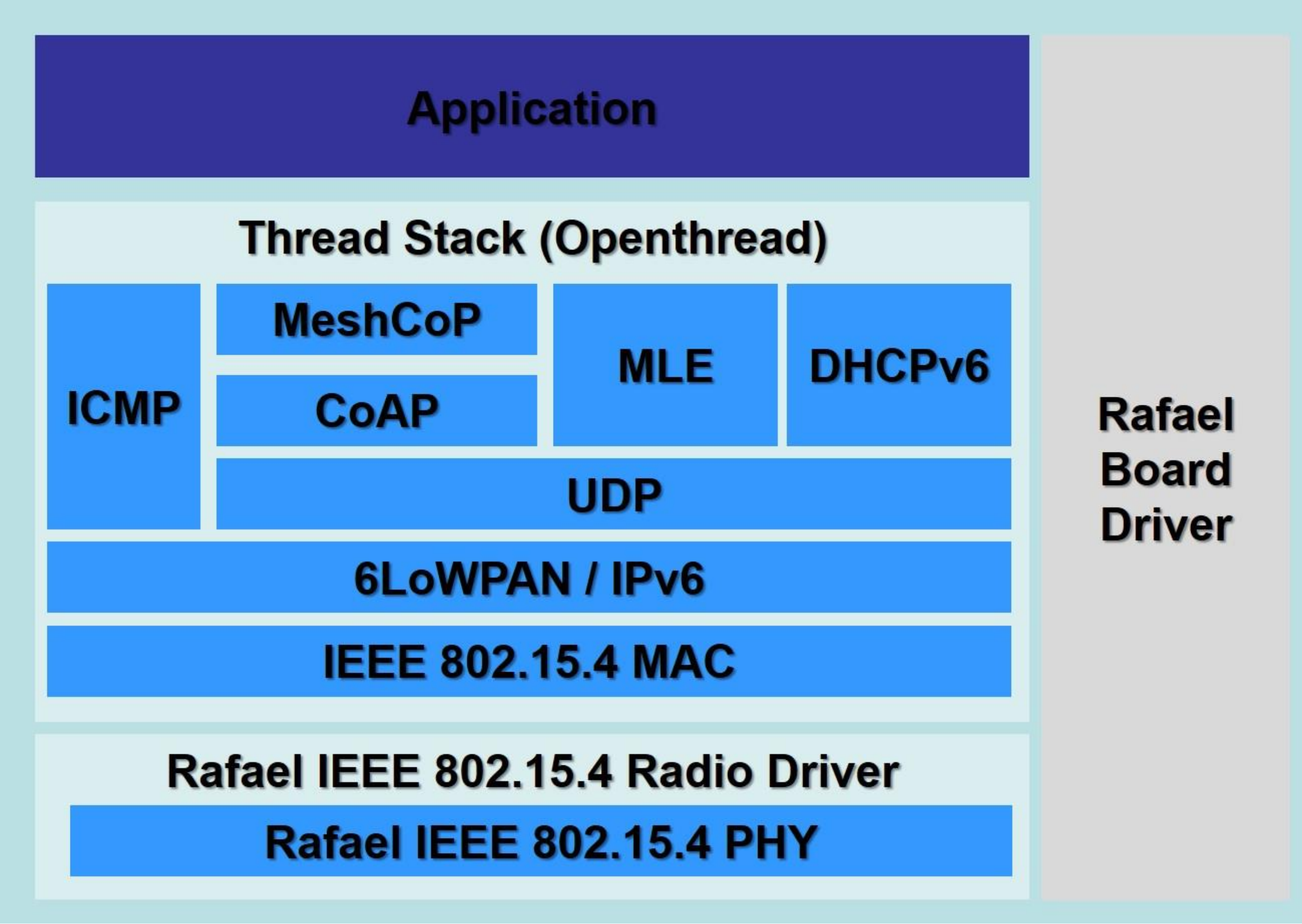
V1.0

Table of Contents

1. Introduction

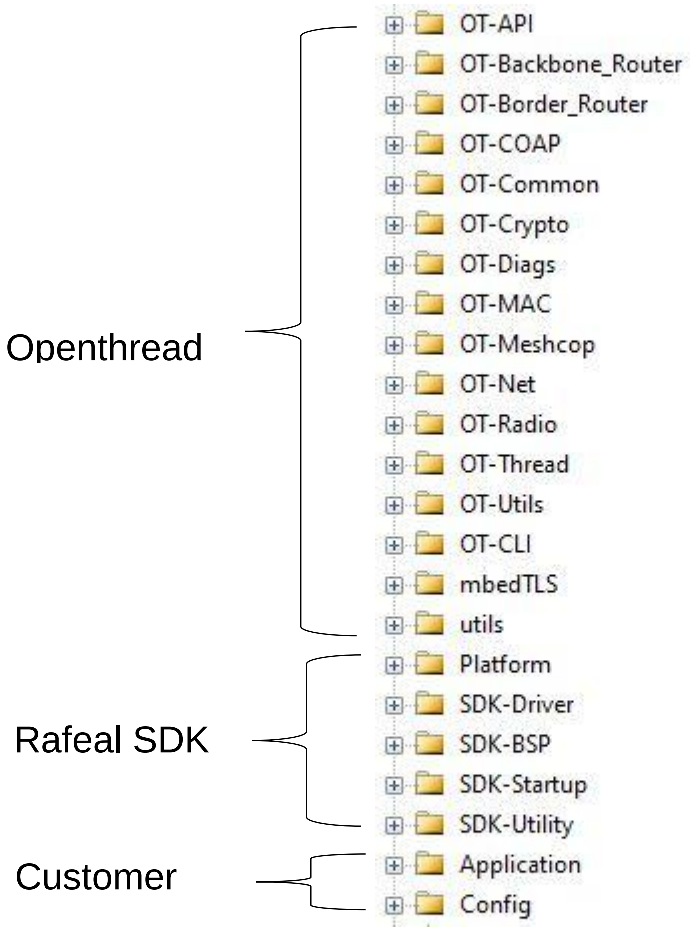
This document describes the interface provided by the Rafael Thread Stack Library. It includes a reference software and supporting library for the Rafael Thread SDK, which combines the open-source project Openthread with Rafael IEEE 802.15.4 and Rafael Driver.
The Rafael Thread Library is easy to integrate with Rafael RT58x series SDK.

2. System architecture



3. Thread SDK introduction

This section provides an overview of the file distribution within the Rafael Thread SDK specifically related to Keil operations.



OT-API
OT-Backbone_Router
OT-Border_Router
OT-COAP
OT-Common
OT-Crypto
OT-Diags
OT-MAC
OT-Meshcop
OT-Net
OT-Radio
OT-Thread
OT-Utils
OT-CLI
mbedTLS
utils
Platform
SDK-Driver
SDK-BSP
SDK-Startup
SDK-Utility
Application
Config

4. Thread Application introduction

This section introduces the examples and APIs included in the Rafael Thread SDK.

🗁 Project\Application\Thread\ftd
- 🗀 app.c
- 🗀 app.h
- 🗀 main_ftd.c
- 🗀 uart.cpp

4.1 Initaial

This function initializes the parameters of a thread network.

- static void _Set_Network_Configuration()

This function initializes the parameters of the sleep node.

- void _Sleep_Init()

This function configures the UDP settings.

- void _Udp_Init()

This function initializes the OTA.

- ota_init(g_app_instance)

4.2 Command register

This function registers the command

- static const otCliCommand kCommands[] = {name, callback}

1. *name* : registers command name
2. *callback*: command callback

4.3 application process action

This function includes the process of OpenThread tasks and the process of Rafael's driver.

- void _app_process_action()
  - otTaskletsProcess(g_app_instance)
  - otSysProcessDrivers(g_app_instance)

Customers can independently add their own task-processing procedures.

4.4 Software timer

Software timer create

- sw_timer_t *sw_timer_create(const char *name, uint32_t period, uint32_t auto_reload, uint32_t execute_mode, void *cb_param, sw_timer_cb cb_function)

1. *name* : A human readable text name that is assigned to the timer.
2. *period* : The period of the timer.
3. *auto_reload* : If auto_reload is set to TRUE, then the timer will expire repeatedly with a frequency set by the period parameter. If auto_reload is set to FALSE, then the timer will be a one-shot and enter the dormant state after it expires.
4. *execute_mode* : The execute mode of the timer.
5. *cb_param* : The input parameter of the call back function.
6. *cb_function* : The function to call when the timer expires

Software timer start

- sw_timer_err_t sw_timer_start(sw_timer_t *timer)

1. *timer* : The handle of the timer being started/restarted.

Software timer stop

- sw_timer_err_t sw_timer_stop(sw_timer_t *timer)

1. *timer* : The handle of the timer being stopped.

Software timer reset

- sw_timer_err_t sw_timer_reset(sw_timer_t *timer)

1. *timer* : The handle of the timer being reset/started/restarted.

Software timer change period

- sw_timer_err_t sw_timer_change_period(sw_timer_t *timer, uint32_t period)

1. *timer* : The handle of the timer that is having its period changed.
2. *period* : The new period for the timer.

Software timer change execute mode

- sw_timer_err_t sw_timer_change_execute_mode(sw_timer_t *timer, uint32_t execute_mode)

1. *timer* : The handle of the timer that has its execute mode.
2. *execute_mode*: The new execute mode for the timer.

Software timer delete

- sw_timer_err_t sw_timer_delete(sw_timer_t *timer)

1. *timer* : The handle of the timer being deleted.

Software timer get running

- bool sw_timer_get_running(sw_timer_t *timer)

1. *timer* : The timer beging queried.
2. *return* : ture= is active. false = is dormant.

4.5 Memory Manage

Memory allocation

- void * mem_malloc (uint32_t u32_size)

1. *u32_size* : memory size in bytes.
2. *return* : NULL=allocate fail, memory pointer=allocate success.

Memory Free

- void mem_free (void * ptr)

1. *ptr* : allocate memory pointer.

Memory Copy

- void mem_memcpy (void *dest_ptr, void* src_ptr, uint32_t lens)

1. *dest_ptr* : target copy memory pointer.
2. *src_ptr* : original memory pointer that is being copied.
3. *lens* : copy size.

4.6 Openthread API

Please refer to the OpenThread API.

5. Commonly commands

5.1 Openthread Commands

state

Return state of current state.

```
> state
offline, disabled, detached, child, router or leader
```

```
Done
```

channel

Get the IEEE 802.15.4 Channel value.

Note: SUG-G (1-10), 2.4G(11-26)

```
> channel
11
Done
```

SUG-G	2.4G
1. 920000 KHz 2. 920500 KHz 3. 921000 KHz 4. 921500 KHz 5. 922000 KHz 6. 922500 KHz 7. 923000 KHz 8. 923500 KHz 9. 924000 KHz 10. 924500 KHz	11. 2405 MHz 12. 2410 MHz 13. 2415 MHz 14. 2420 MHz 15. 2425 MHz 16. 2430 MHz 17. 2435 MHz 18. 2440 MHz 19. 2445 MHz 20. 2450 MHz 21. 2455 MHz 22. 2460 MHz 23. 2465 MHz 24. 2470 MHz 25. 2475 MHz 26. 2480 MHZ

panid

Get the IEEE 802.15.4 PAN ID value.

```
> panid
0xdead
Done
```

networkkey

Get the Thread Network Key value.

```
> networkkey
00112233445566778899aabbccddeeff
Done
```

thread start

Enable Thread protocol operation and attach to a Thread network.

```
> thread start
Done
```

thread stop

Disable Thread protocol operation and detach from a Thread network.

```
> thread stop
Done
```

ipaddr

List all IPv6 addresses assigned to the Thread interface.

```
> ipaddr
fdde:ad00:beef:0:0:ff:fe00:0
fdde:ad00:beef:0:558:f56b:d688:799
fe80:0:0:0:f3d9:2a82:c8d8:fe43
Done
```

ping [async] [-I source] <ipaddr> [size] [count] [interval] [hoplimit] [timeout]

Send an ICMPv6 Echo Request.

async: Use the non-blocking mode. New commands are allowed before the ping process terminates.

source: The source IPv6 address of the echo request.

size: The number of data bytes to be sent ; Limit size: 1280 bytes.

count: The number of ICMPv6 Echo Requests to be sent.

interval: The interval between two consecutive ICMPv6 Echo Requests in seconds. The value may have fractional form, for example 0.5.

hoplimit: The hoplimit of ICMPv6 Echo Request to be sent.

timeout: Time in seconds to wait for the final ICMPv6 Echo Reply after sending out the request. The value may have fractional form, for example 3.5.

```
> ping fd00:db8:0:0:76b:6a05:3ae9:a61a
> 16 bytes from fd00:db8:0:0:76b:6a05:3ae9:a61a: icmp_seq=5 hlim=64
time=0ms
1 packets transmitted, 1 packets received. Packet loss = 0.0%. Round-trip
min/avg/max = 0/0.0/0 ms.
Done

> ping -I fd00:db8:0:0:76b:6a05:3ae9:a61a ff02::1 100 1 1 1
> 108 bytes from fd00:db8:0:0:f605:fb4b:d429:d59a: icmp_seq=4 hlim=64
time=7ms
1 packets transmitted, 1 packets received. Round-trip min/avg/max = 7/7.0/7 ms.
Done
```

Udp send <ip> <port> <message>

Send a UDP message.
ip: the destination address.
port: the UDP destination port.
message: the message to send ; Limit size: 640 characters.

```
> udp send fdde:ad00:beef:0:bb1:ebd6:ad10:f33 1234 hello
Done
```

For more details about OpenThread commands, please refer.

5.2 Rafael User Commands

euiset

- Set the device eui64 value.

```
> euiset 1122334455667
Done
```

6. Start Thread network

1. Prepare two boards with the flashed Example (Thread_2P4G.bin or Thread_SubG).
2. Open a terminal (Tera Term)
3. Connect to the used COM port with the following direct UART settings:
   - Baud rate: 115200
   - 8 data bits
   - 1 stop bit
   - No parity
   - Flow control: none
4. Check the board 1 role

```
> state
leader
Done
```

5. Check the board 2 role

```
> state
child
Done
```

6. Check the board 2 IP

```
> ipaddr
fdde:ad00:beef:0:0:ff:fe00:0
fdde:ad00:beef:0:558:f56b:d688:799
fe80:0:0:0:f3d9:2a82:c8d8:fe43
Done
```

7. Use board 1 to ping board 2

```
> ping fdde:ad00:beef:0:558:f56b:d688:799
> 16 bytes from ping fdde:ad00:beef:0:558:f56b:d688:799: icmp_seq=5 hlim=64
time=0ms
1 packets transmitted, 1 packets received. Packet loss = 0.0%. Round-trip
min/avg/max = 0/0.0/0 ms.
Done
```

8. To modify Thread network parameters, you can either use code to programmatically modify the parameters or use commands through the command line or configuration interface.

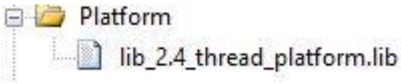
7. Configuration

7.1 Project Configuration

- PLAFFORM_CONFIG_ENABLE_SUBG

  If defined, enable SUG-G; otherwise, default to using 2.4GHz.

  Note: Use the corresponding library for the platform.



```
Platform
  lib_2.4_thread_platform.lib
```

7.2 Main Configuration

- RFB_DATA_RATE

  Set SUG-G data rate value; supported Value: [FSK_50K; FSK_100K; FSK_150K; FSK_200K; FSK_300K].

- RFB_CCA_THRESHOLD

  Set clear channel assessment (CCA) threshold value; Default: 75 (-75 dBm)

Revision History

Revision	Description	Owner	Date
V1.0	Initial version	Jiemin	2023/06/26

Information in this document is provided in connection with **Rafael Microelectronics, Inc.** (“**Rafael Micro**”) products. These materials are provided by **Rafael Micro** as a service to its customers and may be used for informational purposes only. **Rafael Micro** assumes no responsibility for errors or omissions in these materials. **Rafael Micro** may make changes to this document at any time, without notice. **Rafael Micro** advises all customers to ensure that they have the latest version of this document and to verify, before placing orders, that information being relied on is current and complete. **Rafael Micro** makes no commitment to update the information and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to its specifications and product descriptions.

THESE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, RELATING TO SALE AND/OR USE OF **RAFAEL MICRO** PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, CONSEQUENTIAL OR INCIDENTAL DAMAGES, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. **RAFAEL MICRO** FURTHER DOES NOT WARRANT THE ACCURACY OR COMPLETENESS OF THE INFORMATION, TEXT, GRAPHICS OR OTHER ITEMS CONTAINED WITHIN THESE MATERIALS. **RAFAEL MICRO** SHALL NOT BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUES OR LOST PROFITS, WHICH MAY RESULT FROM THE USE OF THESE MATERIALS.

Rafael Micro products are not intended for use in medical, lifesaving or life sustaining applications. **Rafael Micro** customers using or selling **Rafael Micro** products for use in such applications do so at their own risk and agree to fully indemnify **Rafael Micro** for any damages resulting from such improper use or sale. **Rafael Micro, logos** and **RT568** are **Trademarks** of **Rafael Microelectronics, Inc**. Product names or services listed in this publication are for identification purposes only, and may be trademarks of third parties. Third-party brands and names are the property of their respective owners.