NRC7292 Evaluation Kit User Guide

(Standalone SDK API)

Ultra-low power & Long-range Wi-Fi

Ver 3.7
Mar. 16. 2022

NEWRACOM, Inc.

© Copyright Newracom 2022. All rights reserved.
Confidential

NRC7292 Evaluation Kit User Guide (Standalone SDK API)
Ultra-low power & Long-range Wi-Fi

© 2022 NEWRACOM, Inc.

All right reserved. No part of this document may be reproduced in any form without written permission from Newracom.
Newracom reserves the right to change in its products or product specification to improve function or design at any time without notice.

Office

Newracom, Inc.
25361 Commercentre Drive, Lake Forest, CA 92630 USA
http://www.newracom.com

Contents

1 Overview 13
2 General 14
2.1.1 Error Type 14
3 Wi-Fi 15
3.1 Data Type 15
3.1.1 API Status Return Value 15
3.1.2 Device Mode 15
3.1.3 Wifi State 16
3.1.4 Country Code 16
3.1.5 Security Mode 17
3.1.6 Bandwidth 17
3.1.7 IP Mode 17
3.1.8 SCAN_RESULT 18
3.1.9 SCAN_RESULTS 18
3.2 Function Call 19
3.2.1 nrc_wifi_get_ip_mode 19
3.2.2 nrc_wifi_set_ip_mode 19
3.2.3 nrc_wifi_set_ip_address 19
3.2.4 nrc_wifi_add_network 20
3.2.5 nrc_wifi_remove_network 20
3.2.6 nrc_wifi_get_country 20
3.2.7 nrc_wifi_set_country 21
3.2.8 nrc_wifi_set_ssid 21
3.2.9 nrc_wifi_set_bssid 22
3.2.10 nrc_wifi_set_scan_freq 22
3.2.11 nrc_wifi_get_bssid 23
3.2.12 nrc_wifi_set_security 23
3.2.13 nrc_wifi_get_enable_key 23
3.2.14 nrc_wifi_scan 24
3.2.15 nrc_wifi_scan_results 24
3.2.16 nrc_wifi_abort_scan 24
3.2.17 nrc_wifi_disassociate 25
3.2.18 nrc_wifi_connect 25
3.2.19 nrc_wifi_disconnect 25

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2.20 nrc_wifi_get_ip_address 26
3.2.21 nrc_wifi_set_state 26
3.2.22 nrc_wifi_get_state 26
3.2.23 nrc_wifi_set_tx_power 27
3.2.24 nrc_wifi_get_tx_power 27
3.2.25 nrc_wifi_register_event_handler 28
3.2.26 nrc_wifi_get_rssi 28
3.2.27 nrc_wifi_get_snr 28
3.2.28 nrc_wifi_get_mac_address 29
3.2.29 nrc_wifi_get_device_mode 29
3.2.30 nrc_wifi_set_rate_control 29
3.2.31 nrc_wifi_get_rate_control 30
3.2.32 nrc_wifi_get_aid 30
3.2.33 nrc_wifi_get_network_index 30
3.2.34 nrc_wifi_country_from_string 31
3.2.35 nrc_wifi_country_to_string 31
3.2.36 register_vendor_ie_handler 32
3.2.37 unregister_vendor_ie_handler 32
3.2.38 nrc_wifi_set_s1g_config 32
3.2.39 nrc_wifi_wps_pbc 33
3.2.40 nrc_wifi_set_channel 33
3.2.41 nrc_wifi_get_channel 33
3.2.42 nrc_wifi_get_ch_bw 34
3.2.43 nrc_wifi_softap_set_ip 34
3.2.44 nrc_wifi_softap_set_conf 34
3.2.45 nrc_wifi_softap_start 35
3.2.46 nrc_wifi_softap_start_dhcp_server 35
3.2.47 nrc_wifi_set_tx_time 36
3.2.48 nrc_wifi_set_bss_max_idle 36
3.2.49 nrc_wifi_enable_duty_cycle 37
3.2.50 nrc_wifi_disable_duty_cycle 37
3.2.51 nrc_wifi_set_cca_threshold 38
3.2.52 nrc_wifi_set_mcs 38
3.3 Callback Functions & Events 39
4 System 40
4.1 Data Type 40
4.1.1 Trace Level 40
4.1.2 Trace Types 40

© Copyright Newracom 2022. All rights reserved.
Confidential

4.2 Function Call 42
4.2.1 nrc_wifi_set_bdf_use 42
4.2.2 nrc_wifi_get_bdf_use 42
4.2.3 nrc_wifi_set_cal_use 42
4.2.4 nrc_wifi_get_cal_use 43
4.2.5 nrc_wifi_set_log_level 43
4.2.6 nrc_wifi_get_log_level 44
4.2.7 nrc_get_rtc 44
4.2.8 nrc_reset_rtc 44
5 Timer 45
5.1 Data Type 45
5.1.1 Timer Information Type 45
5.1.2 Timer Struct 45
5.2 Function Call 45
5.2.1 nrc_hw_timer_init 45
5.2.2 nrc_hw_timer_deinit 46
5.2.3 nrc_hw_timer_start 46
5.2.4 nrc_hw_timer_stop 47
5.2.5 nrc_hw_timer_clear_irq 47
5.3 Callback Functions & Events 47
6 UART 48
6.1 Data Type 48
6.1.1 Channel 48
6.1.2 UART Data Bit 48
6.1.3 UART Stop Bit 48
6.1.4 UART Parity Bit 49
6.1.5 UART Hardware Flow Control 49
6.1.6 UART FIFO 49
6.1.7 UART Configuration 49
6.1.8 UART Interrupt Type 50
6.2 Function Call 50
6.2.1 nrc_uart_set_config 50
6.2.2 nrc_hw_set_channel 50
6.2.3 nrc_uart_get_interrupt_type 51
6.2.4 nrc_uart_set_interrupt 51
6.2.5 nrc_uart_clear_interrupt 52
6.2.6 nrc_uart_put 52
6.2.7 nrc_uart_get 53

© Copyright Newracom 2022. All rights reserved.
Confidential

6.2.8 nrc_uart_register_interrupt_handler 53
6.2.9 nrc_uart_console_enable 53
6.3 Callback Functions & Events 54
7 GPIO 55
7.1 Data Type 55
7.1.1 GPIO Pin 55
7.1.2 GPIO Direction 55
7.1.3 GPIO Mode 56
7.1.4 GPIO Level 56
7.1.5 GPIO Alternative Function 56
7.1.6 GPIO Configurations 56
7.2 Function Call 57
7.2.1 nrc_gpio_config 57
7.2.2 nrc_gpio_output 57
7.2.3 nrc_gpio_outputb 57
7.2.4 nrc_gpio_input 58
7.2.5 nrc_gpio_inputb 58
7.2.6 nrc_gpio_register_interrupt_handler 59
7.3 Callback Functions & Events 59
8 I2C 60
8.1 Data Type 60
8.2 Function Call 60
8.2.1 nrc_i2c_init 60
8.2.2 nrc_i2c_enable 60
8.2.3 nrc_i2c_reset 61
8.2.4 nrc_i2c_start 61
8.2.5 nrc_i2c_stop 61
8.2.6 nrc_i2c_writebyte 62
8.2.7 nrc_i2c_readbyte 62
9 ADC 63
9.1 Data Type 63
9.1.1 ADC Channel 63
9.2 Function Call 63
9.2.1 nrc_adc_init 63
9.2.2 nrc_adc_deinit 63
9.2.3 nrc_adc_get_data 64
10 PWM 65

© Copyright Newracom 2022. All rights reserved.
Confidential

10.1 Data Type 65
10.1.1 PWM Channel 65
10.2 Function Call 65
10.2.1 nrc_pwm_hw_init 65
10.2.2 nrc_pwm_set_config 66
10.2.3 nrc_pwm_set_enable 66
11 SPI 68
11.1 Data Type 68
11.1.1 SPI Mode 68
11.1.2 SPI Frame Bits 68
11.2 Function Call 69
11.2.1 nrc_spi_init 69
11.2.2 nrc_spi_enable 69
11.2.3 nrc_spi_start_xfer 70
11.2.4 nrc_spi_stop_xfer 70
11.2.5 nrc_spi_xfer 70
11.2.6 nrc_spi_writebyte_value 71
11.2.7 nrc_spi_readbyte_value 71
11.2.8 nrc_spi_write_values 71
11.2.9 nrc_spi_read_values 72
12 HTTP Client 73
12.1 Data Type 73
12.1.1 HTTP Client Return Types 73
12.1.2 Define values 73
12.1.3 HTTP Client Connection Handle 74
12.1.4 SSL Certificate Structure 74
12.1.5 HTTP Client Data Type 74
12.2 Function Call 74
12.2.1 nrc_httpc_get 74
12.2.2 nrc_httpc_post 75
12.2.3 nrc_httpc_put 76
12.2.4 nrc_httpc_delete 76
12.2.5 nrc_httpc_delete 77
12.2.6 nrc_httpc_recv_response 78
12.2.7 nrc_httpc_close 78
13 FOTA 79
13.1 Data Type 79
13.1.1 FOTA Information 79

© Copyright Newracom 2022. All rights reserved.
Confidential

13.2 Function Call 79
13.2.1 nrc_fota_is_support 79
13.2.2 nrc_fota_write 80
13.2.3 nrc_fota_erase 80
13.2.4 nrc_fota_set_info 80
13.2.5 nrc_fota_update_done 81
13.2.6 nrc_fota_update_done_bootloader 81
13.2.7 nrc_fota_cal_crc 82
14 Power save 83
14.1 Data Type 83
14.1.1 Power Save Sleep Mode 83
14.1.2 Power Save TIM Mode 83
14.1.3 Power Save Wakeup Source 83
14.2 Function Call 84
14.2.1 nrc_ps_set_sleep 84
14.2.2 nrc_ps_deep_sleep 84
14.2.3 nrc_ps_modem_sleep 84
14.2.4 nrc_ps_tim_sleep 85
14.2.5 nrc_ps_set_modemsleep_stop 85
14.2.6 nrc_ps_set_gpio_wakeup_pin 86
14.2.7 nrc_ps_set_gpio_wakeup_source 86
14.2.8 nrc_ps_wakeup_reason 86
15 PBC (Push Button) 88
15.1 Data Type 88
15.1.1 pbc_ops 88
15.2 Function Call 88
15.2.1 wps_pbc_fail_cb 88
15.2.2 wps_pbc_timeout_cb 89
15.2.3 wps_pbc_success_cb 89
15.2.4 wps_pbc_button_pressed_event 89
15.2.5 wps_pbc_set_fail_cb 90
15.2.6 wps_pbc_set_timeout_cb 90
15.2.7 wps_pbc_set_success_cb 90
15.2.8 wps_pbc_set_btn_pressed_cb 91
15.2.9 init_wps_pbc 91
16 Middleware API Reference 92
16.1 FreeRTOS 92
16.2 WPA_supplicant 92

© Copyright Newracom 2022. All rights reserved.
Confidential

16.3 lwIP 92
16.4 MbedTLS 93
16.5 NVS library 93
17 S1G Channel 94
17.1 Overview 94
17.2 Channel 94
17.2.1 US (United States) Channel 94
17.2.2 KR (Korea) Channel 96
17.2.3 JP (Japan) channel 96
17.2.4 TW (Taiwan) channel 97
17.2.5 EU (Europe) channel 97
17.2.6 CN (China) channel 98
17.2.7 NZ (New Zealand) channel 98
17.2.8 AU (Australia) channel 99
18 Abbreviations 100
19 Revision history 101

© Copyright Newracom 2022. All rights reserved.
Confidential

List of Tables

Table 2.1 Error Type 14
Table 3.1 tWIFI_STATUS 15
Table 3.2 tWIFI_DEVICE_MODE 15
Table 3.3 tWIFI_STATE_ID 16
Table 3.4 tWIFI_COUNTRY_CODE 16
Table 3.5 tWIFI_SECURITY 17
Table 3.6 tWIFI_BANDWIDTH 17
Table 3.7 tWIFI_IP_MODE 17
Table 3.8 SCAN_RESULT 18
Table 3.9 Security Flags 18
Table 3.10 SCAN_RESULTS 18
Table 3.11 tWIFI_EVENT_ID 39
Table 4.1 TRACE_LEVEL 40
Table 4.2 TRACE_TYPES 40
Table 5.1 TIMER_INFO 45
Table 5.2 TIMER_STRUCT 45
Table 6.1 NRC_UART_CHANNEL 48
Table 6.2 NRC_UART_DATA_BIT 48
Table 6.3 NRC_UART_STOP_BIT 48
Table 6.4 NRC_UART_PARITY_BIT 49
Table 6.5 NRC_UART_HW_FLOW_CTRL 49
Table 6.6 NRC_UART_FIFO 49
Table 6.7 NRC_UART_CONFIG 49
Table 6.8 NRC_UART_INT_TYPE 50
Table 7.1 NRC_GPIO_PIN 55
Table 7.2 NRC_GPIO_DIR 55
Table 7.3 NRC_GPIO_MODE 56
Table 7.4 NRC_GPIO_LEVEL 56
Table 7.5 NRC_GPIO_ALT 56
Table 7.6 NRC_GPIO_CONFIG 56
Table 9.1 ADC_CH 63
Table 10.1 PWM_CH 65
Table 11.1 SPI_MODE 68
Table 11.2 SPI_FRAME_BITS 68
Table 12.1 httpc_ret_e 73
Table 12.2 Default define values 73
Table 12.3 con_handle_t 74
Table 12.4 ssl_certs_t 74
Table 12.5 httpc_data_t 74
Table 13.1 FOTA_INFO 79

© Copyright Newracom 2022. All rights reserved.
Confidential

Table 14.1 POWER_SAVE_SLEEP_MODE 83
Table 14.2 POWER_SAVE_TIM_MODE 83
Table 14.3 power save wakeup source 83
Table 15.1 pbc_ops 88
Table 17.1 US Channel 94
Table 17.2 KR USN Channel 96
Table 17.3 KR MIC Channel 96
Table 17.4 JP Channel 96
Table 17.5 TW Channel 97
Table 17.6 EU Channel 97
Table 17.7 CN Channel 98
Table 17.8 NZ Channel 98
Table 17.9 AU Channel 99
Table 18.1 Abbreviations and acronyms 100

© Copyright Newracom 2022. All rights reserved.
Confidential

List of Figures

Figure 1.1 NRC7292 FreeRTOS Host Architecture 13

© Copyright Newracom 2022. All rights reserved.
Confidential

1 Overview

This document introduces the Application Programming Interface (API) for standalone NRC7292 Software Development Kit (SDK). These APIs are used for Wi-Fi operations and events and other peripherals on the NRC7292 Evaluation Boards (EVB).

The user application is implemented using SDK API, 3rd party libraries and system hal APIs. The lwIP is used for TCP/IP related codes. The mbedtls is related to encrption and decryption. stack is SDK API is used for Wi-Fi operations and getting configurations. The FreeRTOS is a real-time operating system kernel for embedded devices. It provides methods for multiple threads or tasks, mutexes, semaphores and software timers. Wifi API is implemented based on wpa_supplicant. It provides the general Wi-Fi operations such as scan, connect, set Wi-Fi configurations, and get system status information such as RSSI, SNR.

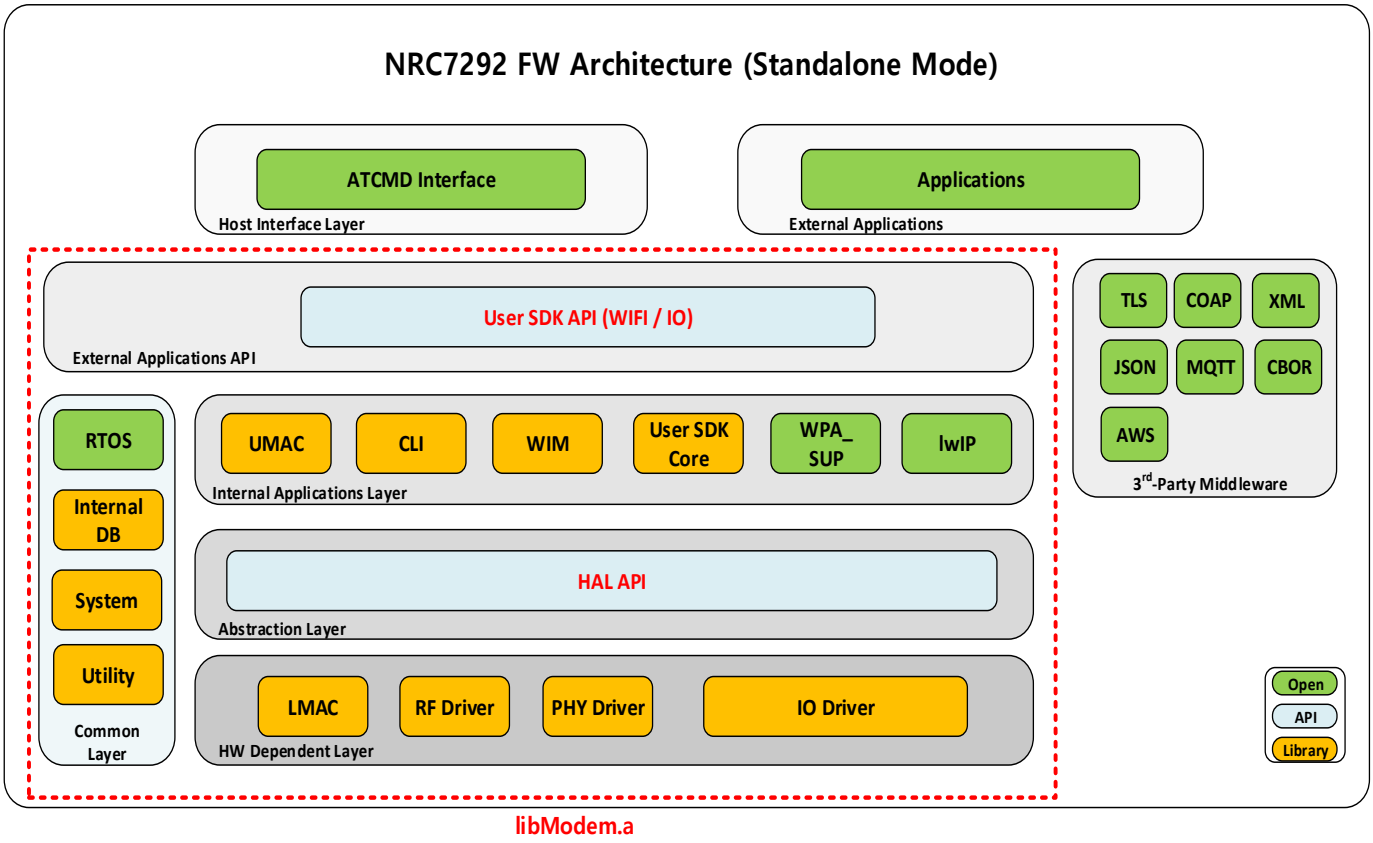

Figure 1.1 NRC7292 FreeRTOS Host Architecture

© Copyright Newracom 2022. All rights reserved.
Confidential

2 General

The general data types are defined at the “NRC7292/API/Inc/nrc_types.h”.

2.1.1 Error Type

nrc_err_t is an operation function return type. These types are defined at the “lib/sdk/inc/nrc_types.h”.

Table 2.1 Error Type

Name	Description
NRC_SUCCESS	Operation was successful
NRC_FAIL	Operation failed

© Copyright Newracom 2022. All rights reserved.
Confidential

3 Wi-Fi

The Wi-Fi API provides functions to:
• Scan & connect to AP
• Configuration the wifi settings
• Set and get the IP address

3.1 Data Type

These types are defined at the “NRC7292/API/Inc/api_wifi.h”.

3.1.1 API Status Return Value

tWIFI_STATUS is returned by API functions to indicate whether a function call succeeded or failed.

Table 3.1 tWIFI_STATUS

Name	Description
WIFI_SUCCESS	Operation succssful
WIFI_FAIL	Operation failed
WIFI_NOMEM	No memory
WIFI_INVALID	Invalid parameter
WIFI_INIT_FAIL	Wi-Fi initial is failed
WIFI_CONNECTION_FAIL	Wi-Fi connection is failed
WIFI_DHCP_FAIL	Get DHCP client is failed
WIFI_DHCP_TIMEOUT	Get IP address is timeout
WIFI_SET_IP_FAIL	Set IP address is failed
WIFI_SOFTAP_FAIL	SoftAP start is failed

3.1.2 Device Mode

tWIFI_DEVICE_MODE is the bandwidth.

Table 3.2 tWIFI_DEVICE_MODE

Name	Description
WIFI_MODE_STATION	Station
WIFI_MODE_AP	Access Point
WIFI_MODE_MESH_POINT	Mesh Point

© Copyright Newracom 2022. All rights reserved.
Confidential

3.1.3 Wifi State

tWIFI_STATE_ID is the wifi state.

Table 3.3 tWIFI_STATE_ID

Name	Description
WIFI_STATE_INIT	Initial
WIFI_STATE_READY	Wi-Fi ready
WIFI_STATE_TRY_CONNECT	Try to connect
WIFI_STATE_CONNECTED	Connected
WIFI_STATE_TRY_GET_IP	Try to get IP address
WIFI_STATE_GET_IP	Get IP address
WIFI_STATE_TRY_DISCONNECT	Try to disconnect
WIFI_STATE_DISCONNECTED	Disconnected
WIFI_STATE_SOFTAP_CONF	Set the softAP configuration
WIFI_STATE_SOFTAP_START	SoftAP is started
WIFI_STATE_DHCPS_START	DHCP server is started
WIFI_STATE_TRY_DISASSOC	Try to disassociate
WIFI_STATE_SCAN	Scan
WIFI_STATE_SCAN_DONE	Scan is finished

3.1.4 Country Code

tWIFI_COUNTRY_CODE is the country code.

Table 3.4 tWIFI_COUNTRY_CODE

Name	Description
WIFI_CC_UNKNOWN	Unknown value
WIFI_CC_JP	Japan
WIFI_CC_KR	Korea
WIFI_CC_TW	Taiwan
WIFI_CC_US	United States of America
WIFI_CC_EU	Europe
WIFI_CC_CN	China
WIFI_CC_NZ	New Zealand
WIFI_CC_AU	Australia

© Copyright Newracom 2022. All rights reserved.
Confidential

3.1.5 Security Mode

tWIFI_SECURITY is the security mode.

Table 3.5 tWIFI_SECURITY

Name	Description
WIFI_SEC_OPEN	Open
WIFI_SEC_WPA2	WPA2
WIFI_SEC_WPA3_OWE	WPA3 OWE
WIFI_SEC_WPA3_SAE	WPA3 SAE

3.1.6 Bandwidth

tWIFI_BANDWIDTH is the bandwidth.

Table 3.6 tWIFI_BANDWIDTH

Name	Description
WIFI_1M	1 Mhz bandwidth
WIFI_2M	2 Mhz bandwidth
WIFI_4M	4 Mhz bandwidth

3.1.7 IP Mode

tWIFI_IP_MODE is the IP mode.

Table 3.7 tWIFI_IP_MODE

Name	Description
WIFI_DYNAMIC_IP	Dynamic IP, which uses the DHCP client
WIFI_STATIC_IP	Static IP

© Copyright Newracom 2022. All rights reserved.
Confidential

3.1.8 SCAN_RESULT

This is a union of data types for SCAN_RESULTS.

Table 3.8 SCAN_RESULT

Type	Element	Description
char*	items[5]	This is union values. Each array entry points members. items[0] : BSSID items[1] : Frequency items[2] : Signal level items[3] : Flags items[4] : SSID
char*	bssid	BSSID, which is fixed-length, colon-separated hexadecimal ASCII string. (Ex. "84:25:3f:01:5e:50")
char*	freq	Frequency. The frequency is equivalent WiFi channel (2.4 / 5G frequency) (Ex. “5205”). See the “S1G Channel“
char*	sig_level	Numeric ASCII string of RSSI. (Ex. "-25"). The unit is dBm
char*	flags	ASCII string of the security model for the network. See the table 2.9.
char*	ssid	ASCII string of SSID.

Table 3.9 Security Flags

Name	Description
WPA2-EAP	Wi-Fi Protected Access 2 – Extensible Authentication Protocol
WPA2-PSK	Wi-Fi Protected Access 2 – Pre-Shared Key
WPA3-SAE	Wi-Fi Protected Access 3 - Simultaneous Authentication of Equals
WPA3-OWE	Wi-Fi Protected Access 3 - Opportunistic Wireless Encryption

3.1.9 SCAN_RESULTS

This is a structure for function nrc_wifi_scan_results().

Table 3.10SCAN_RESULTS

Type	Element	Description
int	n_result	number of scanned bssid
SCAN_RESULT	result[SCAN_RESULT_NUM]	scan results

※ SCAN_RESULT_NUM is a maximum scan results number, defaults is 10.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2 Function Call

These APIs are defined at the "sdk/api/api_wifi.h".

3.2.1 nrc_wifi_get_ip_mode

Get the IP mode.

Prototype :

tWIFI_STATUS nrc_wifi_get_ip_mode(tWIFI_IP_MODE* mode)

Input Parameters :

mode

Type: tWIFI_IP_MODE*

Purpose: Static IP or Dynamic IP.

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.2 nrc_wifi_set_ip_mode

Get the IP mode.

Prototype :

tWIFI_STATUS nrc_wifi_set_ip_mode(tWIFI_IP_MODE* mode, char* ip_addr)

Input Parameters :

mode

Type: tWIFI_IP_MODE*

Purpose: Static IP or Dynamic IP ip_addr

Type char*

Purpose: A pointer to set static IP which is ASCII string. (Ex. “192.168.200.23”)

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.3 nrc_wifi_set_ip_address

Set IP address. It requests a dynamic IP via DHCP or set a static IP.

Prototype :

tWIFI_STATUS nrc_wifi_set_ip_address(void)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

None

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.4 nrc_wifi_add_network

Add a network index associated with the Wi-Fi connection.

Prototype :

tWIFI_STATUS nrc_wifi_add_network(int *index)

Input Parameters :

index

Type: int*
Purpose: A pointer to receive assigned network ID. This will output a number, which is the network ID.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.5 nrc_wifi_remove_network

Remove a network index associated with the Wi-Fi connection.

Prototype :

tWIFI_STATUS nrc_wifi_remove_network(int index)

Input Parameters :

index

Type: int
Purpose: Network index.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.6 nrc_wifi_get_country

Get the country code.

Prototype :

tWIFI_STATUS nrc_wifi_get_country(tWIFI_COUNTRY_CODE *country_code)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

country_code
Type: char*
Purpose: country code. See "Country Code".

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.7 nrc_wifi_set_country

Set the country code.

Prototype :

tWIFI_STATUS nrc_wifi_set_set_country (char *country_code)

Input Parameters :

country_code
Type: char*
Purpose: country code. See "Country Code".

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.8 nrc_wifi_set_ssid

Set the SSID of the AP to connect. (*STA only)

Prototype :

tWIFI_STATUS nrc_wifi_set_ssid(int index, char * ssid)

Input Parameters :

index
Type: int
Purpose: Network index.

ssid
Type: char*
Purpose: A pointer to set ssid bssid which is ASCII string. . The maximum length of the name is 32 bytes.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2.9 nrc_wifi_set_bssid

Set the BSSID(Basic Service Set Identifier) of the AP to connect. (*STA only)

Prototype :

tWIFI_STATUS nrc_wifi_set_bssid(int index, char * ssid)

Input Parameters :

index

Type: int
Purpose: Network index.

bssid

Type: char*
Purpose: A pointer to set bssid which is colon-separated hexadecimal ASCII string. (Ex. "84:25:3f:01:5e:50"). The maximum length of the name is 17 bytes

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.10 nrc_wifi_set_scan_freq

Set the scan channel list for scanning AP

Prototype :

tWIFI_STATUS nrc_wifi_set_scan_freq(int index, uint16_t *freq, uint8_t num_freq)

nput Parameters :

index

Type: int
Purpose: Network index.

freq

Type: uint16_t*
Purpose: A pointer to the frequency list. The frequency should be assigned equivalent WiFi channel(2.4 / 5G frequency) (Ex. “5205 5200). See the “S1G Channel“

num_freq

Type: uint8_t
Purpose: number of frequencies.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2.11 nrc_wifi_get_bssid

Get the bssid.

Prototype :

tWIFI_STATUS nrc_wifi_get_bssid(char *bssid)

Input Parameters :

bssid

Type: char*

Purpose: A pointer to get bssid which is colon-separated hexadecimal ASCII string. (Ex. "84:25:3f:01:5e:50"). The maximum length of the name is 17 bytes.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.12 nrc_wifi_set_security

Set the security parameters for Wi-Fi connection.

Prototype :

void nrc_wifi_set_security (int index, int mode, char *password)

Input Parameters :

index

Type: int

Purpose: Network index.

mode

Type: int

Purpose: security mode, See “Security Mode”.

password

Type: char*

Purpose: A pointer to set password which is ASCII string. (Ex. "123ABDC"). The maximum length of the password is 30 bytes.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.13 nrc_wifi_get_enable_key

Get information on whether security is active.

Prototype :

bool nrc_wifi_get_enable_key(void)

© Copyright Newracom 2022. All rights reserved.
Confidential

Input Parameters :

None

Returns :

TRUE, if the security is enabled.
FALSE, if the security is disabled.

3.2.14 nrc_wifi_scan

Scan the AP

Prototype :

int nrc_wifi_scan (void)

Input Parameters :

None

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.15 nrc_wifi_scan_results

Get scan results.

Prototype :

tWIFI_STATUS nrc_wifi_scan_results(SCAN_RESULTS *results)

Input Parameters :

results
Type: SCAN_RESULTS*
Purpose: scan lists. See "SCAN_RESULTS".

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.16 nrc_wifi_abort_scan

Stop the scan procedure.

Prototype :

int nrc_wifi_abort_scan (void)

Input Parameters :

None

© Copyright Newracom 2022. All rights reserved.
Confidential

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.17 nrc_wifi_disassociate

Disassociate all stations or a specific station equal to mac address.

Prototype :

tWIFI_STATUS nrc_wifi_disassociate(char* mac_addr)

Input Parameters :

mac_addr

Type: char*

Purpose: A pointer to set broadcast(ff:ff:ff:ff:ff:ff) or single sta's MAC Address which is colon-separated hexadecimal ASCII string.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.18 nrc_wifi_connect

Connect to AP

※ The AP information such as ssid, security should be set befere calling this function.

Prototype :

tWIFI_STATUS nrc_wifi_connect (int index)

Input Parameters :

index

Type: int

Purpose: Network index.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.19 nrc_wifi_disconnect

Disconnect from the AP.

Prototype :

© Copyright Newracom 2022. All rights reserved.
Confidential

tWIFI_STATUS nrc_wifi_disconnect (int index)

Input Parameters :

index

Type: int

Purpose: Network index.

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.20 nrc_wifi_get_ip_address

Get the current IP address.

Prototype :

tWIFI_STATUS nrc_wifi_get_ip_address(char **ip_addr)

Input Parameters :

ip_addr

Type: char**

Purpose: A double pointer to get the address of IP address.

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.21 nrc_wifi_set_state

Set the current Wi-Fi connection state.

Prototype :

tWIFI_STATUS nrc_wifi_set_state(tWIFI_STATE_ID state)

Input Parameters :

state

Type: tWIFI_STATE_ID

Purpose: wifi state. See "Wifi_STATE".

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.22 nrc_wifi_get_state

Get the current Wi-Fi connection state.

© Copyright Newracom 2022. All rights reserved.
Confidential

Prototype :

tWIFI_STATUS nrc_wifi_get_state(tWIFI_STATE_ID* state)

Input Parameters :

state

Type: tWIFI_STATE_ID*

Purpose: wifi state. See "Wifi_STATE".

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.23 nrc_wifi_set_tx_power

Set TX power

Prototype :

tWIFI_STATUS nrc_wifi_set_tx_power(int txpower)

Input Parameters :

txpower

Type: int

Purpose: TX Power (in dBm) (8~18)

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.24 nrc_wifi_get_tx_power

Get the TX power.

Prototype :

tWIFI_STATUS nrc_wifi_set_tx_power(int txpower)

Input Parameters :

txpower

Type: int*

Purpose: TX Power

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2.25 nrc_wifi_register_event_handler

Register a Wi-Fi event handler callback function. The callback function will be called when a Wi-Fi event happens. See the "Callback Functions & Events"

Prototype :

tWIFI_STATUS nrc_wifi_register_event_handler(event_callback_fn fn)

Input Parameters :

fn

Type: event_callback_fn

Purpose: event handler for wifi connection and dhcp.

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.26 nrc_wifi_get_rssi

Get the RSSI value.

Prototype :

tWIFI_STATUS nrc_wifi_get_rssi(int8_t *rssi)

Input Parameters :

rssi

Type: int8_t*

Purpose: A pointer to get RSSI. The RSSI is signed binary number.

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.27 nrc_wifi_get_snr

Get the SNR value.

Prototype :

tWIFI_STATUS nrc_wifi_get_snr(uint8_t *snr)

Input Parameters :

snr

Type: uint8_t*

Purpose: A pointer to get SNR. The RSSI is unsigned binary number

Returns :

WIFI_SUCCESS, if the operation was successful.

© Copyright Newracom 2022. All rights reserved.
Confidential

Error code(tWIFI_STATUS), all other errors.

3.2.28 nrc_wifi_get_mac_address

Get the MAC address.

Prototype :

tWIFI_STATUS nrc_wifi_get_mac_address(char *addr)

Input Parameters :

addr

Type: char*

Purpose: A pointer to get MAC address which is colon-separated hexadecimal ASCII string. (Ex. "84:25:32:11:5e:50").

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.29 nrc_wifi_get_device_mode

Get the device mode.

Prototype :

tWIFI_STATUS nrc_wifi_get_device_mode(tWIFI_DEVICE_MODE *mode)

Input Parameters :

mode

Type: char*

Purpose: device mode. See “Device Mode”.

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.30 nrc_wifi_set_rate_control

Set the MCS rate control option.

Prototype :

tWIFI_STATUS nrc_wifi_set_rate_control(bool enable)

Input Parameters :

enable

Type: bool

© Copyright Newracom 2022. All rights reserved.
Confidential

Purpose: rate control enable / disable

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.31 nrc_wifi_get_rate_control

Get the MCS rate control option.

Prototype :

tWIFI_STATUS nrc_wifi_get_rate_control(bool *enable)

Input Parameters :

enable

Type: bool*
Purpose: A pointer to get rate control enable / disable

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.32 nrc_wifi_get_aid

Get the association ID, which is allocated by AP.

Prototype :

tWIFI_STATUS nrc_wifi_get_aid(int *aid)

Input Parameters :

aid

Type: int*
Purpose: A pointer to get association ID, which is signed binary number.

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.33 nrc_wifi_get_network_index

Get the current network index from global network index.

Prototype :

tWIFI_STATUS nrc_wifi_get_network_index(int *index)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

index
Type: int
Purpose: A pointer to get current network ID.

Returns :
WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.34 nrc_wifi_country_from_string

Get the country code from string.

Prototype :
tWIFI_COUNTRY_CODE nrc_wifi_country_from_string(const char *str)
Input Parameters :
str
Type: const char*
Purpose: A pointer to assign country code string which is ASCII string. See "Country Code".
Returns :
WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.35 nrc_wifi_country_to_string

Get string from country code index.

Prototype :
const char *nrc_wifi_country_to_string(tWIFI_COUNTRY_CODE cc)
Input Parameters :
cc
Type: tWIFI_COUNTRY_CODE
Purpose: country code. See "Country Code"
Returns :
WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2.36 register_vendor_ie_handler

Register vendor ie handler

Prototype :

tWIFI_STATUS register_vendor_ie_handler(int cmd, vendor_ie_event_callback_fn func)

Input Parameters :

cmd

Type: int

Purpose: command value (0xF0 ~ 0xF4)

cc

Type: vendor_ie_event_callback_fn

Purpose: event callback function

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.37 unregister_vendor_ie_handler

Unregister vendor ie handler

Prototype :

tWIFI_STATUS register_vendor_ie_handler(int cmd)

Input Parameters :

cmd

Type: int

Purpose: command value (0xF0 ~ 0xF4)

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.38 nrc_wifi_set_s1g_config

Set S1G channel

Prototype :

tWIFI_STATUS nrc_wifi_set_s1g_config(uint16_t s1g_channel)

Input Parameters :

s1g_channel

Type: uint16_t

Purpose: S1G channel.

© Copyright Newracom 2022. All rights reserved.
Confidential

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.39 nrc_wifi_wps_pbc

Set WPS Pushbutton

Prototype :

tWIFI_STATUS nrc_wifi_wps_pbc()

Input Parameters :

N/A

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.40 nrc_wifi_set_channel

Set frequency (Sub-1GHz)

Prototype :

tWIFI_STATUS nrc_wifi_set_channel(uint32_t s1g_freq)

Input Parameters :

s1g_freq
Type: uint32_t
Purpose: S1G channel frequency (MHz/10)

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.41 nrc_wifi_get_channel

Get frequency (Sub-1GHz)

Prototype :

tWIFI_STATUS nrc_wifi_get_channel(uint32_t *s1g_freq)

Input Parameters :

s1g_freq
Type: uint32_t *
Purpose: S1G channel frequency (MHz/10)

© Copyright Newracom 2022. All rights reserved.
Confidential

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.42 nrc_wifi_get_ch_bw

Get channel bandwidth

Prototype :

tWIFI_STATUS nrc_wifi_get_ch_bw(uint8_t *bandwidth)

Input Parameters :

bandwidth
Type: uint8_t *
Purpose: 0(1M BW) or 1(2M BW) or 2(4M BW)

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.43 nrc_wifi_softap_set_ip

Set IP for softap

Prototype :

tWIFI_STATUS nrc_wifi_softap_set_ip(char *ip_addr)

Input Parameters :

ip_addr
Type: char *
Purpose: Set IP address for softap

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.44 nrc_wifi_softap_set_conf

Set configuration for softap

Prototype :

tWIFI_STATUS nrc_wifi_softap_set_conf(int index, char *ssid, int channel, int sec_mode, char *password)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

index
Type: int
Purpose: network index
ssid
Type: char *
Purpose: SSID
channel
Type: int
Purpose: 11ah channel
sec_mode
Type: int
Purpose: security mode (tWIFI_SECURITY) See "Country Code"
password
Type: char *
Purpose: PASSWORD

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.45 nrc_wifi_softap_start

Start softap

Prototype :

tWIFI_STATUS nrc_wifi_softap_start(int index)

Input Parameters :

index
Type: int
Purpose: network index

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.46 nrc_wifi_softap_start_dhcp_server

Start DHCP Server

Prototype :

tWIFI_STATUS nrc_wifi_softap_start_dhcp_server(void)

Input Parameters :

N/A

© Copyright Newracom 2022. All rights reserved.
Confidential

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.47 nrc_wifi_set_tx_time

Set carrier sense time and pause time

Prototype :

tWIFI_STATUS nrc_wifi_set_tx_time(uint16_t cs_time, uint32_t pause_time)

Input Parameters :

cs_time
- Type: uint16_t
- Purpose: Carrier sensing time. Listen before talk (time unit: us) (0~12480)

pause_time
- Type: uint32_t
- Purpose: Tx pause time (time unit : us)

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.48 nrc_wifi_set_bss_max_idle

Set BSS MAX IDLE period and retry count. If you want to add BSS Max Idle IE, this API should be added.

Prototype :

tWIFI_STATUS nrc_wifi_set_bss_max_idle(int index, int period, int retry_cnt)

Input Parameters :

index
- Type: int
- Purpose: network index

period
- Type: int
- Purpose: bss max idle period. (0 ~ 2,147,483,647)

retry_cnt
- Type: int
- Purpose: retry count for receiving keep alive packet from STA. (1 ~ 100)

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2.49 nrc_wifi_enable_duty_cycle

Enable duty cycle

Prototype :

tWIFI_STATUS nrc_wifi_enable_duty_cycle(uint32_t window, uint32_t duration, uint32_t margin)

Input Parameters :

window

Type: uint32_t
Purpose: duty cycle window (time unit : us)

duration

Type: uint32_t
Purpose: specify allowed tx duration within duty cycle window (time unit : us)

cs_time

Type: uint32_t
Purpose: duty margin (time unit : us)

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

3.2.50 nrc_wifi_disable_duty_cycle

Disable duty cycle

Prototype :

tWIFI_STATUS nrc_wifi_disable_duty_cycle(void)

Input Parameters :

None

Returns :

WIFI_SUCCESS, if the operation was successful.
Error code(tWIFI_STATUS), all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.2.51 nrc_wifi_set_cca_threshold

Set CCA(Clear Channel Assessment) threshold

Prototype :

tWIFI_STATUS nrc_wifi_set_cca_threshold(int cca_threshold)

Input Parameters :

cca_threshold

Type: int

Purpose: CCA threshold.(unit: dBm) (-100 ~ -70)

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

3.2.52 nrc_wifi_set_mcs

Set MCS. It is applied when rate control is disabled

Prototype :

tWIFI_STATUS nrc_wifi_set_mcs(uint8_t mcs)

Input Parameters :

mcs

Type: uint8_t

Purpose: Modulation Coding Scheme (0 ~ 10)

Returns :

WIFI_SUCCESS, if the operation was successful.

Error code(tWIFI_STATUS), all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

3.3 Callback Functions & Events

Prototype :

void (*event_callback_fn)(tWIFI_EVENT_ID event)

Input Parameters :

event

Type: tWIFI_EVENT_ID

Purpose: Wi-Fi Event

Table 3.11tWIFI_EVENT_ID

Name	Description
WIFI_EVT_CONNECT_SUCCESS	Connection
WIFI_EVT_CONNECT_FAIL	Connection is failed
WIFI_EVT_GET_IP	Get IP address is success
WIFI_EVT_GET_IP_FAIL	Get IP address is failed
WIFI_EVT_DISCONNECT	Disconnection
WIFI_EVT_START_SOFT_AP	SoftAP is started
WIFI_EVT_SET_SOFT_AP_IP	IP address is assigned to SoftAP
WIFI_EVT_START_DHCP_SERVER	DHCP server is started
WIFI_EVT_SCAN	Scan is started
WIFI_EVT_SCAN_DONE	Scan is finished
WIFI_EVT_VENDOR_IE	Vendor IE

© Copyright Newracom 2022. All rights reserved.
Confidential

4 System

The system API provides functions to:
• Set and get the system configuration values
• Set the debug log level

4.1 Data Type

4.1.1 Trace Level

TRACE_LEVEL is a system log level. These types are defined at the “lib/modem/inc/util/util_trach.h”.

Table 4.1 TRACE_LEVEL

Name	Description
TL_VB	All messages logged with Trace level and Workflow Tracking logs
TL_INFO	All messages logged with Information or higher
TL_ERR	All messages logged with Error level or higher

4.1.2 Trace Types

TRACE_TYPE is the module name for trace log. These types are defined at the “lib/modem/inc/util/util_trach.h”.

Table 4.2 TRACE_TYPES

Name	Description	Default trace level
TT_QM	Queue manager	TL_ERR
TT_HIF	Host interface	TL_INFO
TT_WIM	Wireless information message	TL_INFO
TT_API	HAL API	TL_INFO
TT_MSG	Message	TL_INFO
TT_RX	Receive	TL_INFO
TT_TX	Transmit	TL_INFO
TT_DL	Downlink	TL_INFO
TT_UL	Uplink	TL_INFO
TT_PHY	Physical Layer	TL_INFO
TT_RF	Radio frequency	TL_ERR
TT_UMAC	Upper MAC	TL_INFO
TT_PS	Power save	TL_INFO
TT_TWT	Target wake time	TL_INFO
TT_HALOW	Halow certificate test	TL_INFO

© Copyright Newracom 2022. All rights reserved.
Confidential

TT_WPAS	Wpa supplicant	TL_INFO
TT_RC	Rate control	TL_INFO
TT_NET	Network	TL_INFO
TT_CMD	Command	TL_INFO
TT_MM	Memory manager	TL_INFO
TT_BA	Block ACK	TL_ERR
TT_FRAG	Fragmentation attack	TL_INFO
TT_SDK_GPIO	SDK GPIO API	TL_INFO
TT_SDK_HTTPC	SDK http client API	TL_INFO
TT_SDK_HTTPD	SDK http server daemon API	TL_INFO
TT_SDK_FOTA	SDK FOTA API	TL_INFO
TT_SDK_PS	SDK power save API	TL_INFO
TT_SDK_I2C	SDK I2C API	TL_ERR
TT_SDK_UART	SDK UART API	TL_ERR
TT_SDK_ADC	SDK ADC API	TL_INFO
TT_SDK_PWM	SDK PWM API	TL_INFO
TT_SDK_SPI	SDK SPI API	TL_INFO
TT_SDK_TIMER	SDK timer API	TL_INFO
TT_SDK_WIFI	SDK wifi API	TL_INFO
TT_SDK_WLAN_MANAGER	SDK wlan manager	TL_INFO
TT_RCV	Recovery	TL_INFO
TT_BMT	Beacon monitor	TL_INFO
TT_TEMP_SENSOR	Temperature sensor	TL_ERR

© Copyright Newracom 2022. All rights reserved.
Confidential

4.2 Function Call

The header file for system APIs are defined at the "sdk/inc/api_system.h".

4.2.1 nrc_wifi_set_bdf_use

Set the board data usage. If the value is enabled, transmission power control(TPC) is applied.

(*currently, not used)

Prototype :

nrc_err_t nrc_wifi_set_bdf_use(bool enable)

Input Parameters :

enable

Type: bool

Purpose: Enable(true) / disable(false) the board data

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

4.2.2 nrc_wifi_get_bdf_use

Get the board data en. (*currently, not used)

Prototype :

nrc_err_t nrc_wifi_get_bdf_use(bool *enabled)

Input Parameters :

enable

Type: bool

Purpose: A pointer to get bdf_use value.

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

4.2.3 nrc_wifi_set_cal_use

Set the calibration usage. If this value is enabled and RF calibration table is existed in serial flash, the calibration table value will be applied during transmission.

Prototype :

nrc_err_t nrc_wifi_set_cal_use(bool enable)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

enable
Type: bool
Purpose: Enable(true) / disable(false) the calibration usage

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

4.2.4 nrc_wifi_get_cal_use

Get the board data enable / disable for transmission power control. (* currently, not used)

Prototype :

nrc_err_t nrc_wifi_get_cal_use(bool *enabled)

Input Parameters :

enable
Type: bool*
Purpose: A pointer to get calibration usage value.

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

4.2.5 nrc_wifi_set_log_level

Set the log level for type id

Prototype :

nrc_err_t nrc_wifi_set_log_level(TRACE_TYPES type_id, TRACE_LEVEL level)

Input Parameters :

type_id
Type: TRACE_TYPES
Purpose: trace module name. See the "Trace Types"
level
Type: TRACE_LEVEL
Purpose: log level. See the "Trace Level"

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

4.2.6 nrc_wifi_get_log_level

Get the log level for type id

Prototype :

nrc_err_t nrc_wifi_get_log_level(TRACE_TYPES type_id, TRACE_LEVEL *level)

Input Parameters :

type_id

Type: TRACE_TYPES

Purpose: trace module name. See the "Trace Types"

level

Type: TRACE_LEVEL*

Purpose: A pointer to get log level. See the "Trace Level"

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

4.2.7 nrc_get_rtc

Retrieve the real time clock value since cold boot

Prototype :

nrc_err_t nrc_get_rtc(uint64_t* rtc_time)

Input Parameters :

rtc_time

Type: uint64_t*

Purpose: A pointer to get RTC time.

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

4.2.8 nrc_reset_rtc

Reset the real time clock to 0

Prototype :

void nrc_reset_rtc(void)

Input Parameters :

None

Returns :

None

© Copyright Newracom 2022. All rights reserved.
Confidential

5 Timer

The timer API provides functions to:
• Start and stop the timer

5.1 Data Type

These types are defined at the “lib/sdk/inc/api_timer.h”.

5.1.1 Timer Information Type

TIMER_INFO is an information about timer.

Table 5.1 TIMER_INFO

Name	Description
initialized	Timer is initialized
ch	Timer channel
cb	Timer callback function

※ Maximum 2 timers are supported in NRC7292. (32bits timer (channel 0), 64bits timer (channel 3))

5.1.2 Timer Struct

TIMER Struct is an array of TIMER_INFO_T values.

Table 5.2 TIMER_STRUCT

Name	Description
Timer[TIMER_MAX]	A list of timers

※ TIMER_MAX is 2 in NRC7292.

5.2 Function Call

The header file for system APIs are defined at the "sdk/inc/api_timer.h".

5.2.1 nrc_hw_timer_init

Initialize the hardware timer and register callback function.

Prototype :

nrc_err_t nrc_hw_timer_init(int ch, timer_callback isr_cb)

© Copyright Newracom 2022. All rights reserved.
Confidential

Input Parameters :

ch

Type: int

Purpose: timer channel

isr_cb

Type: timer_callback

Purpose: callback handler function when the timer expired

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

5.2.2 nrc_hw_timer_deinit

De-initialize the hardware timer.

Prototype :

nrc_err_t nrc_hw_timer_deinit(int ch)

Input Parameters :

ch

Type: int

Purpose: timer channel

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

5.2.3 nrc_hw_timer_start

Start the hardware timer.

Prototype :

nrc_err_t nrc_hw_timer_start(int ch, uint64_t time)

Input Parameters :

ch

Type: int

Purpose: timer channel

time

Type: time

Purpose: time duration

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

5.2.4 nrc_hw_timer_stop

Stop the hardware timer with timer channel ID.

Prototype :

nrc_err_t nrc_hw_timer_stop(int ch)

Input Parameters :

ch

Type: int

Purpose: timer channel

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

5.2.5 nrc_hw_timer_clear_irq

Clear interrupt request (IRQ) with timer channel ID

※ The IRQ should be cleared in the timer ISR callback function.

Prototype :

nrc_err_t nrc_hw_timer_clear_irq(int ch)

Input Parameters :

ch

Type: int

Purpose: timer channel

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

5.3 Callback Functions & Events

Prototype :

typedef void (*timer_callback)(int vector)

Input Parameters :

vector

Type: int

Purpose: input vector

© Copyright Newracom 2022. All rights reserved.
Confidential

6 UART

The UART API provides functions to:
• Set the UART channel, configurations, interrupt handler and interrupt type
• Get and put a character and print strings

6.1 Data Type

These types are defined at the “lib/sdk/inc/api_uart.h”.

6.1.1 Channel

NRC_UART_CHANNEL is an UART channel.

Table 6.1 NRC_UART_CHANNEL

Name	Description
NRC_UART_CH0	Channel 0
NRC_UART_CH1	Channel 1
NRC_UART_CH2	Channel 2
NRC_UART_CH3	Channel 3

6.1.2 UART Data Bit

NRC_UART_DATA_BIT is a data bit size.

Table 6.2 NRC_UART_DATA_BIT

Name	Description
NRC_UART_DB5	Data bit 5
NRC_UART_DB6	Data bit 6
NRC_UART_DB7	Data bit 7
NRC_UART_DB8	Data bit 8

6.1.3 UART Stop Bit

NRC_UART_STOP_BIT is a data bit size.

Table 6.3 NRC_UART_STOP_BIT

Name	Description
NRC_UART_SB1	Stop bit 1
NRC_UART_SB2	Stop bit 2

© Copyright Newracom 2022. All rights reserved.
Confidential

6.1.4 UART Parity Bit

NRC_UART_PARITY_BIT is a type of parity.

Table 6.4 NRC_UART_PARITY_BIT

Name	Description
NRC_UART_PB_NONE	None
NRC_UART_PB_ODD	Odd parity bit
NRC_UART_PB_EVEN	Even parity bit

6.1.5 UART Hardware Flow Control

NRC_UART_HW_FLOW_CTRL indicate that a UART hardware flow control is enabled or disabled.

Table 6.5 NRC_UART_HW_FLOW_CTRL

Name	Description
NRC_UART_SB1	Stop bit 1
NRC_UART_SB2	Stop bit 2

6.1.6 UART FIFO

NRC_UART_FIFO indicate that a UART FIFO is enabled or disabled.

Table 6.6 NRC_UART_FIFO

Name	Description
NRC_UART_FIFO_DISABLE	Disable FIFO
NRC_UART_FIFO_ENABLE	Enable FIFO

6.1.7 UART Configuration

NRC_UART_CONFIG is a configuration about UART.

Table 6.7 NRC_UART_CONFIG

Name	Description
ch	Channel number
db	Data bit
br	Baudrate
stop_bit	Stop bit
parity_bit	Parity bit
hw_flow_ctrl	Enable or disable hardware flow control

© Copyright Newracom 2022. All rights reserved.
Confidential

fifo	Enable or disable FIFO

6.1.8 UART Interrupt Type

NRC_UART_INT_TYPE is an interrupt type.

Table 6.8 NRC_UART_INT_TYPE

Name	Description
NRC_UART_INT_NONE	None
NRC_UART_INT_ERROR	Error
NRC_UART_INT_TIMEOUT	Timeout
NRC_UART_INT_RX_DONE	Rx is done
NRC_UART_INT_TX_EMPTY	Tx is empty

6.2 Function Call

The header file for system APIs are defined at the "sdk/inc/api_uart.h".

6.2.1 nrc_uart_set_config

Set the UART configurations.

Prototype :

nrc_err_t nrc_uart_set_config(NRC_UART_CONFIG *conf)

Input Parameters :

conf

Type: NRC_UART_CONFIG*

Purpose: A pointer to set uart configurations. See “UART Configuration”

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

6.2.2 nrc_hw_set_channel

Set the UART channel

Prototype :

nrc_err_t nrc_uart_set_channel(int ch)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

ch
Type: int
Purpose: UART channel

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

6.2.3 nrc_uart_get_interrupt_type

Get the UART interrupt type.

Prototype :

nrc_err_t nrc_uart_get_interrupt_type(int ch, NRC_UART_INT_TYPE *type)

Input Parameters :

ch
Type: int
Purpose: UART channel
type
Type: NRC_UART_INT_TYPE *
Purpose: A pointer to set UART interrupt type. See "UART Interrupt Type"

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

6.2.4 nrc_uart_set_interrupt

Set the UART interrupt.

Prototype :

nrc_err_t nrc_uart_set_interrupt(int ch, bool tx_en, bool rx_en)

Input Parameters :

ch
Type: int
Purpose: UART channel
tx_en
Type: bool
Purpose: Tx enable flagh
rx_en
Type: bool
Purpose: Rx enable flag

Returns :

NRC_SUCCESS, if the operation was successful.

© Copyright Newracom 2022. All rights reserved.
Confidential

NRC_FAIL, all other errors.

6.2.5 nrc_uart_clear_interrupt

Clear the UART interrupt.

Prototype :

nrc_err_t nrc_uart_clear_interrupt(int ch, bool tx_int, bool rx_int , bool timeout_int)

Input Parameters :

ch
- Type: int
- Purpose: UART channel

tx_en
- Type: bool
- Purpose: Tx enable flagh

rx_en
- Type: bool
- Purpose: Rx enable flag

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

6.2.6 nrc_uart_put

Put the character data to UART.

Prototype :

nrc_err_t nrc_uart_put(int ch, char data)

Input Parameters :

ch
- Type: int
- Purpose: UART channel

data
- Type: char
- Purpose: data

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

6.2.7 nrc_uart_get

Get the character data from UART.

Prototype :

nrc_err_t nrc_uart_get(int ch, char *data)

Input Parameters :

ch

Type: int

Purpose: UART channel

data

Type: char*

Purpose: A pointer to get data

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

6.2.8 nrc_uart_register_interrupt_handler

Register user callback function for UART input.

Prototype :

nrc_err_t nrc_uart_register_interrupt_handler(int ch, intr_handler_fn cb)

Input Parameters :

ch

Type: int

Purpose: timer channel

cb

Type: intr_handler_fn

Purpose: callback function

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

6.2.9 nrc_uart_console_enable

Enable/disable uart print and console command.

Prototype :

nrc_err_t nrc_uart_console_enable(bool enabled)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

Enabled
Type: bool
Purpose: true or false to enable or disable console print and command.

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

6.3 Callback Functions & Events

The interrupt handler function pointer type is defined at the “sdk/inc/nrc_types.h”.

Prototype :

typedef void (*intr_handler_fn)(int vector)

Input Parameters :

vector
Type: int
Purpose: input vector

© Copyright Newracom 2022. All rights reserved.
Confidential

7 GPIO

The GPIO API provides functions to:
• Set the GPIO configurations and interrupt handler
• Get GPIO input values and set GPIO output values

7.1 Data Type

These types are defined at the “lib/sdk/inc/api_gpio.h”.

7.1.1 GPIO Pin

NRC_GPIO_PIN is a GPIO pin number.

Table 7.1 NRC_GPIO_PIN

Name	Description
GPIO_00	GPIO 0
GPIO_01	GPIO 1
GPIO_02	GPIO 2
GPIO_03	GPIO 3
GPIO_08	GPIO 8
GPIO_09	GPIO 9
GPIO_10	GPIO 10
GPIO_11	GPIO 11
GPIO_12	GPIO 12
GPIO_13	GPIO 13
GPIO_14	GPIO 14
GPIO_15	GPIO 15
GPIO_16	GPIO 16
GPIO_17	GPIO 17

※ These GPIOs are used for NRC7292. The supported GPIOs are different in each chip. Please reference the hardware guide ducouement.

7.1.2 GPIO Direction

NRC_GPIO_DIR is a GPIO direction.

Table 7.2 NRC_GPIO_DIR

Name	Description
GPIO_INPUT	Input direction
GPIO_OUTPUT	Output direction

© Copyright Newracom 2022. All rights reserved.
Confidential

7.1.3 GPIO Mode

NRC_GPIO_MODE is a GPIO mode.

Table 7.3 NRC_GPIO_MODE

Name	Description
GPIO_PULL_UP	Pull up
GPIO_PULL_DOWN	Pull down
GPIO_FLOATING	Floating

7.1.4 GPIO Level

NRC_GPIO_LEVEL is a GPIO level.

Table 7.4 NRC_GPIO_LEVEL

Name	Description
GPIO_LEVEL_LOW	0
GPIO_LEVEL_HIGH	1

7.1.5 GPIO Alternative Function

NRC_GPIO_ALT is a alternative function.

Table 7.5 NRC_GPIO_ALT

Name	Description
GPIO_FUNC	GPIO function
GPIO_NOMAL_OP	GPIO Normal operation

7.1.6 GPIO Configurations

NRC_GPIO_CONFIG is a GPIO configuration.

Table 7.6 NRC_GPIO_CONFIG

Name	Description
gpio_pin	Pin number
gpio_dir	Direction
gpio_alt	Alternative function
gpio_mode	Mode

© Copyright Newracom 2022. All rights reserved.
Confidential

7.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_gpio.h”.

7.2.1 nrc_gpio_config

Set the GPIO configuration.

Prototype :

nrc_err_t nrc_gpio_config(NRC_GPIO_CONFIG *conf)

Input Parameters :

conf

Type: NRC_GPIO_CONFIG*

Purpose: A pointer to set GPIO configurations. See “GPIO Configurations”

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

7.2.2 nrc_gpio_output

Set the GPIO data (32bits).

Prototype :

nrc_err_t nrc_gpio_output(uint32_t *word)

Input Parameters :

conf

Type: uint32_t *

Purpose: A pointer to set GPIO output value (32bits)

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

7.2.3 nrc_gpio_outputb

Set the GPIO data for a specified pin number.

Prototype :

nrc_err_t nrc_gpio_outputb(int pin, int level)

Input Parameters :

pin

Type: int

Purpose: GPIO pin number

© Copyright Newracom 2022. All rights reserved.
Confidential

level
Type: int
Purpose: output value level

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

7.2.4 nrc_gpio_input

Get the GPIO data (32bits).

Prototype :

nrc_err_t nrc_gpio_input(uint32_t *word)

Input Parameters :

conf
Type: uint32_t *
Purpose: A pointer to get GPIO output value (32bits)

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

7.2.5 nrc_gpio_inputb

Get the GPIO data for a specified pin number.

Prototype :

nrc_err_t nrc_gpio_inputb(int pin, int *level)

Input Parameters :

pin
Type: int
Purpose: GPIO pin number
level
Type: int
Purpose: A pointer to get GPIO input value

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

7.2.6 nrc_gpio_register_interrupt_handler

Register GPIO interrupt handler.

※NRC729 support level trigger.

Prototype :

nrc_gpio_register_interrupt_handler(int pin, intr_handler_fn cb)

Input Parameters :

pin

Type: int

Purpose: pin number

cb

Type: intr_handler_fn

Purpose: callback function

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

7.3 Callback Functions & Events

The interrupt handler function pointer type is defined at the “sdk/inc/nrc_types.h”.

Prototype :

typedef void (*intr_handler_fn)(int vector)

Input Parameters :

vector

Type: int

Purpose: input vector

© Copyright Newracom 2022. All rights reserved.
Confidential

8 I2C

The I2C API provides functions to:
• Set the I2C configurations
• I2C initialize, enable, reset
• Read and write byte via I2C

8.1 Data Type

These types are defined at the “lib/sdk/inc/api_i2c.h”.

8.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_i2c.h”.

8.2.1 nrc_i2c_init

Initialize the I2C controller.

Prototype :

nrc_err_t nrc_i2c_init(uint8_t scl, uint8_t sda, uint32_t clk)

Input Parameters :

scl
Type: uint8_t
Purpose: Serial clock line for I2C

sda
Type: uint8_t
Purpose: Serial data line for I2C

clk
Type: uint32_t
Purpose: Clock speed of I2C controller. It can be set up to 400,000Hz.

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

8.2.2 nrc_i2c_enable

Enable or disable the I2C controller.

※ Please disable I2C only after a transaction is stopped.

© Copyright Newracom 2022. All rights reserved.
Confidential

Prototype :

nrc_err_t nrc_i2c_enable(bool enable)

Input Parameters :

enable

Type: bool

Purpose: I2C controller enable or disable

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

8.2.3 nrc_i2c_reset

Reset the I2C controller.

Prototype :

nrc_err_t nrc_i2c_reset(void)

Input Parameters :

N/A

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

8.2.4 nrc_i2c_start

Start the I2C operation.

Prototype :

nrc_err_t nrc_i2c_start(void)

Input Parameters :

N/A

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

8.2.5 nrc_i2c_stop

Stop the I2C operation.

Prototype :

nrc_err_t nrc_i2c_stop(void)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

N/A

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

8.2.6 nrc_i2c_writebyte

Write data to the I2C controller.

Prototype :

nrc_err_t nrc_i2c_writebyte(uint8_t data)

Input Parameters :

data
Type: uint8_t
Purpose: data

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

8.2.7 nrc_i2c_readbyte

Read data from the I2C controller.

Prototype :

nrc_err_t nrc_i2c_readbyte(uint8_t *data, bool ack)

Input Parameters :

data
Type: uint8_t*
Purpose: A pointer to store the read data
ack
Type: bool
Purpose: ACK flag. If there's no further reading registers, then false. Otherwise, true

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

9 ADC

The ADC API provides functions to:
• Initialize / De-initialize the ADC controller
• Read the ADC controller data

9.1 Data Type

These types are defined at the “lib/sdk/inc/api_adc.h”.

9.1.1 ADC Channel

ADC_CH is an ADC channel.

Table 9.1 ADC_CH

Name	Description
ADC1	ADC channel 1
ADC2	ADC channel 2
ADC3	ADC channel 3

9.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_adc.h”.

9.2.1 nrc_adc_init

Initialize the ADC controller.

Prototype :

nrc_err_t nrc_adc_init(void)

Input Parameters :

N/A

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

9.2.2 nrc_adc_deinit

De-initialize the ADC controller.

Prototype :

© Copyright Newracom 2022. All rights reserved.
Confidential

nrc_err_t nrc_adc_deinit(void)

Input Parameters :

N/A

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

9.2.3 nrc_adc_get_data

Read the data from the ADC controller.

Prototype :

nrc_err_t nrc_adc_get_data(uint32_t id, uint16_t *data)

Input Parameters :

id
- Type: uint32_t
- Purpose: Channel ID

data
- Type: uint16_t *
- Purpose: A pointer for of data(Max value : 0x1FF)

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.
NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

10 PWM

The PWM API provides functions to:
• Initialize the PWM controller
• Set configuration and enable for PWM

10.1 Data Type

These types are defined at the “lib/sdk/inc/api_pwm.h”.

10.1.1 PWM Channel

PWM_CH is an PWM channel.

Table 10.1PWM_CH

Name	Description
PWM_CH0	PWM channel 0
PWM_CH1	PWM channel 1
PWM_CH2	PWM channel 2
PWM_CH3	PWM channel 3

※ These PWM channel are used for NRC7292. The supported PWM channels are different in each chip. Please reference the hardware guide document.

10.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_pwm.h”.

10.2.1 nrc_pwm_hw_init

Initialize the ADC controller.

Prototype :

nrc_err_t nrc_pwm_hw_init(uint8_t ch, uint8_t gpio_num, uint8_t use_high_clk)

Input Parameters :

ch
Type: uint8_t
Purpose: PWM channel ID. See “PWM Channel”

gpio_num
Type: uint8_t
Purpose: GPIO number assigned for PWM

use_high_clk

© Copyright Newracom 2022. All rights reserved.
Confidential

Type: uint8_t
Purpose: If 0, then the pulse duration for 1-bit in each pattern is about 20.8us. Otherwise, about 10.4us

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

10.2.2 nrc_pwm_set_config

Set configuration parameters of PWM. One duty cycle consists of 4 pulse patterns(total 128-bit).

※ It starts with the MSB of pattern1 and ends with the LSB of pattern4.

Prototype :

nrc_err_t nrc_pwm_set_config(uint8_t ch, uint32_t pattern1, uint32_t pattern2, uint32_t pattern3, uint32_t pattern4)

Input Parameters :

ch
Type: uint8_t
Purpose: PWM channel ID. See "PWM Channel"

pattern1
Type: uint32_t
Purpose: 1st pulse pattern(Pattern bits 0~31)

pattern2
Type: uint32_t
Purpose: 2nd pulse pattern(Pattern bits 32~63)

pattern3
Type: uint32_t
Purpose: 3rd pulse pattern(Pattern bits 64~95)

pattern4
Type: uint32_t
Purpose: 4th pulse pattern(Pattern bits 96~127)

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

10.2.3 nrc_pwm_set_enable

Enable the specified PWM channel.

Prototype :

nrc_err_t nrc_pwm_set_enable(uint32_t ch, bool enable)

© Copyright Newracom 2022. All rights reserved.
Confidential

Input Parameters :

ch

Type: uint32_t

Purpose: PWM channel ID. See “PWM Channel”

enable

Type: bool

Purpose: Enable / disable

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

11 SPI

The SPI API provides functions to:
• Initialize and enable the SPI controller
• Write and read byte via SPI

11.1 Data Type

These types are defined at the “lib/sdk/inc/api_spi.h”.

11.1.1 SPI Mode

SPI_MODE is a SPI mode, which is related to CPOL and CPHA values.

※ Refer the Serial Peripheral Interface. (https://en.wikipedia.org/wiki/Serial_Peripheral_Interface)

Table 11.1 SPI_MODE

Name	Description
SPI_MODE0	SPI mode 0 (CPOL=0, CPHA=0)
SPI_MODE1	SPI mode 1 (CPOL=0, CPHA=1)
SPI_MODE2	SPI mode 2 (CPOL=1, CPHA=0)
SPI_MODE3	SPI mode 3 (CPOL=1, CPHA=1)

11.1.2 SPI Frame Bits

SPI_FRAME_BITS is a number of frame bits.

Table 11.2 SPI_FRAME_BITS

Name	Description
SPI_BIT4	SPI 4-bit frame
SPI_BIT5	SPI 5-bit frame
SPI_BIT6	SPI 6-bit frame
SPI_BIT7	SPI 7-bit frame
SPI_BIT8	SPI 8-bit frame
SPI_BIT9	SPI 9-bit frame
SPI_BIT10	SPI 10-bit frame
SPI_BIT11	SPI 11-bit frame
SPI_BIT12	SPI 12-bit frame
SPI_BIT13	SPI 13-bit frame
SPI_BIT14	SPI 14-bit frame
SPI_BIT15	SPI 15-bit frame
SPI_BIT16	SPI 16-bit frame

© Copyright Newracom 2022. All rights reserved.
Confidential

11.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_spi.h”.

11.2.1 nrc_spi_init

Initialize the SPI controller with the specified mode and bits

Prototype :

nrc_err_t nrc_spi_init(SPI_MODE mode, SPI_FRAME_BITS bits, uint32_t clock)

Input Parameters :

mode

Type: SPI_MODE

Purpose: SPI mode. See “SPI Mode”

bits

Type: SPI_FRAME_BITS

Purpose: SPI frame bits. See “SPI Frame Bits”

clock

Type: uint32_t

Purpose: SPI clock frequency. The unit is Hz.

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

11.2.2 nrc_spi_enable

Enable / disable the SPI controller.

Prototype :

nrc_err_t nrc_spi_enable(bool enable)

Input Parameters :

enable

Type: bool

Purpose: Enable / disable

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

11.2.3 nrc_spi_start_xfer

Enable CS to continuously transfer data.

Prototype :

nrc_err_t nrc_spi_start_xfer(void)

Input Parameters :

N/A

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

11.2.4 nrc_spi_stop_xfer

Disable CS to continuously transfer data.

Prototype :

nrc_err_t nrc_spi_stop_xfer(void)

Input Parameters :

N/A

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

11.2.5 nrc_spi_xfer

Transfer the data between master and slave. User can call nrc_spi_xfer multiple times to transmit data.

※ This function should run inside nrc_spi_start_xfer() and nrc_spi_stop_xfer().

Prototype :

nrc_err_t nrc_spi_xfer(uint8_t *wbuffer, uint8_t *rbuffer, uint32_t size)

Input Parameters :

wbuffer
- Type: uint8_t*
- Purpose: A pointer to write data

rbuffer
- Type: uint8_t*
- Purpose: A pointer to read data

size
- Type: uint32_t
- Purpose: Number of bytes to transfer

Returns :

© Copyright Newracom 2022. All rights reserved.
Confidential

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

11.2.6 nrc_spi_writebyte_value

Write one-byte data to the specified register address.

Prototype :

nrc_err_t nrc_spi_writebyte_value(uint8_t addr, uint8_t data);

Input Parameters :

addr
- Type: uint8_t
- Purpose: register address to write data

data
- Type: uint8_t
- Purpose: data to write

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

11.2.7 nrc_spi_readbyte_value

Read one-byte data to the specified register address.

Prototype :

nrc_err_t nrc_spi_readbyte_value(uint8_t addr, uint8_t data);

Input Parameters :

addr
- Type: uint8_t
- Purpose: register address to read data

data
- Type: uint8_t*
- Purpose: A pointer to read data

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

11.2.8 nrc_spi_write_values

Write bytes data to the specified register address.

Prototype :

© Copyright Newracom 2022. All rights reserved.
Confidential

nrc_err_t nrc_spi_write_values(uint8_t addr, uint8_t *data, int size)

Input Parameters :

addr
- Type: uint8_t
- Purpose: register address to write data

data
- Type: uint8_t*
- Purpose: A pointer to write data

size
- Type: int
- Purpose: write data size. The unit is bytes.

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

11.2.9 nrc_spi_read_values

Read bytes data to the specified register address.

Prototype :

nrc_err_t nrc_spi_read_values(uint8_t addr, uint8_t *data, int size)

Input Parameters :

addr
- Type: uint8_t
- Purpose: register address to read data

data
- Type: uint8_t*
- Purpose: A pointer to read data

size
- Type: int
- Purpose: read data size. The unit is bytes.

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

12 HTTP Client

The HTTP client API provides functions to:
- HTTP request method (GET, PUT, POST, DELETE)
- Retrieves the response data about request function

12.1 Data Type

These types are defined at the “lib/sdk/inc/api_httpc.h”.

12.1.1 HTTP Client Return Types

httpc_ret_e is a return type for HTTP client.

Table 12.1 httpc_ret_e

Name	Description
HTTPC_RET_ERROR_TLS_CONNECTION	TLS connection fail
HTTPC_RET_ERROR_PK_LOADING_FAIL	Private key loading fail
HTTPC_RET_ERROR_CERT_LOADING_FAIL	Certificate loading fail
HTTPC_RET_ERROR_SEED_FAIL	Seed creation fail
HTTPC_RET_ERROR_BODY_SEND_FAIL	Request body send fail
HTTPC_RET_ERROR_HEADER_SEND_FAIL	Request Header send fail
HTTPC_RET_ERROR_INVALID_HANDLE	Invalid handle
HTTPC_RET_ERROR_ALLOC_FAIL	Memory allocation fail
HTTPC_RET_ERROR_SCHEME_NOT_FOUND	Scheme(http:// or https://) not found
HTTPC_RET_ERROR_SOCKET_FAIL	Socket creation fail
HTTPC_RET_ERROR_RESOLVING_DNS	Cannot resolve the hostname
HTTPC_RET_ERROR_CONNECTION	Connection fail
HTTPC_RET_ERROR_UNKNOWN	Unknown error
HTTPC_RET_CON_CLOSED	Connection closed by remote
HTTPC_RET_OK	Success

12.1.2 Define values

Table 12.2 Default define values

Define	Value
HTTP_PORT	80
HTTPS_PORT	443
INVALID_HANDLE	0xFFFFFFFF

© Copyright Newracom 2022. All rights reserved.
Confidential

12.1.3 HTTP Client Connection Handle

con_handle_t is a connection handle type for HTTP client.

Table 12.3 con_handle_t

Name	Description
con_handle_t	Connection handle

12.1.4 SSL Certificate Structure

ssl_certs_t is a SSL certificate structure type.

Table 12.4 ssl_certs_t

Name	Description
server_cert	Server certification
client_cert	Client certification
client_pk	Client private key
server_cert_length	Server certification I, server_cert buffer size
client_cert_length	Client certification I, client_cert buffer size
client_pk_length	Client private key I, client_pk buffer size

12.1.5 HTTP Client Data Type

httpc_data_t is a data type for HTTP client.

Table 12.5 httpc_data_t

Name	Description
data_out	Connection handle
data_out_length	Output buffer length
data_in	Pointer of the input buffer for data receiving
data_in_length	Input buffer length
recved_size	Actuall received data size

12.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_httpc.h”.

12.2.1 nrc_httpc_get

Executes a GET request on a given URL.

Prototype :

httpc_ret_e nrc_httpc_get(con_handle_t *handle, const char *url, const char *custom_header, httpc_data_t *data, ssl_certs_t *certs)

© Copyright Newracom 2022. All rights reserved.
Confidential

Input Parameters :

handle
- Type: con_handle_t*
- Purpose: Connection handle”

url
- Type: const char *
- Purpose: URL for the request

custom_header
- Type: const char *
- Purpose: Customized request header. The request-line(“<method> <uri> HTTP/1.1”) and “Host: <host-name>” will be sent in default internally. Other headers can be set as null-terminated string format.

Data
- Type: httpc_data_t *
- Purpose: A pointer to the #httpc_data_t to manage the data sending and receiving

certs
- Type: ssl_certs_t *
- Purpose: A pointer to the #ssl_certs_t for the certificates

Returns :

HTTPC_RET_OK, if the operation was successful.
Negative error value, all other errors.

12.2.2 nrc_httpc_post

Executes a POST request on a given URL.

Prototype :

httpc_ret_e nrc_httpc_post(con_handle_t *handle, const char *url, const char *custom_header, httpc_data_t *data, ssl_certs_t *certs)

Input Parameters :

handle
- Type: con_handle_t*
- Purpose: Connection handle”

url
- Type: const char *
- Purpose: URL for the request

custom_header
- Type: const char *
- Purpose: Customized request header. The request-line(“<method> <uri> HTTP/1.1”) and “Host: <host-name>” will be sent in default internally. Other headers can be set as null-terminated string format.

Data

© Copyright Newracom 2022. All rights reserved.
Confidential

Type: httpc_data_t *
Purpose: A pointer to the #httpc_data_t to manage the data sending and receiving
certs
Type: ssl_certs_t *
Purpose: A pointer to the #ssl_certs_t for the certificates

Returns :

HTTPC_RET_OK, if the operation was successful.
Negative error value, all other errors.

12.2.3 nrc_httpc_put

Executes a PUT request on a given URL.

Prototype :

httpc_ret_e nrc_httpc_put(con_handle_t *handle, const char *url, const char *custom_header, httpc_data_t *data, ssl_certs_t *certs)

Input Parameters :

handle
Type: con_handle_t*
Purpose: Connection handle”
url
Type: const char *
Purpose: URL for the request
custom_header
Type: const char *
Purpose: Customized request header. The request-line(“<method> <uri> HTTP/1.1”) and “Host: <host-name>” will be sent in default internally. Other headers can be set as null-terminated string format.
Data
Type: httpc_data_t *
Purpose: A pointer to the #httpc_data_t to manage the data sending and receiving
certs
Type: ssl_certs_t *
Purpose: A pointer to the #ssl_certs_t for the certificates

Returns :

HTTPC_RET_OK, if the operation was successful.
Negative error value, all other errors.

12.2.4 nrc_httpc_delete

Executes a DELETE request on a given URL.

© Copyright Newracom 2022. All rights reserved.
Confidential

Prototype :

httpc_ret_e nrc_httpc_delete(con_handle_t *handle, const char *url, const char *custom_header, httpc_data_t *data, ssl_certs_t *certs)

Input Parameters :

handle

Type: con_handle_t*

Purpose: Connection handle”

url

Type: const char *

Purpose: URL for the request

custom_header

Type: const char *

Purpose: Customized request header. The request-line(“<method> <uri> HTTP/1.1”) and “Host: <host-name>” will be sent in default internally. Other headers can be set as null-terminated string format.

Data

Type: httpc_data_t *

Purpose: A pointer to the #httpc_data_t to manage the data sending and receiving

certs

Type: ssl_certs_t *

Purpose: A pointer to the #ssl_certs_t for the certificates

Returns :

HTTPC_RET_OK, if the operation was successful.

Negative error value, all other errors.

12.2.5 nrc_httpc_delete

Executes a DELETE request on a given URL.

Prototype :

httpc_ret_e nrc_httpc_delete(con_handle_t *handle, const char *url, const char *custom_header, httpc_data_t *data, ssl_certs_t *certs)

Input Parameters :

handle

Type: con_handle_t*

Purpose: Connection handle”

url

Type: const char *

Purpose: URL for the request

custom_header

Type: const char *

© Copyright Newracom 2022. All rights reserved.
Confidential

Purpose: Customized request header. The request-line("<method> <uri> HTTP/1.1") and "Host: <host-name>" will be sent in default internally. Other headers can be set as null-terminated string format.

Data
Type: httpc_data_t *
Purpose: A pointer to the #httpc_data_t to manage the data sending and receiving

certs
Type: ssl_certs_t *
Purpose: A pointer to the #ssl_certs_t for the certificates

Returns :

HTTPC_RET_OK, if the operation was successful.
Negative error value, all other errors.

12.2.6 nrc_httpc_recv_response

Retrieves the response data when there are remains after executing the request functions.

Prototype :

httpc_ret_e nrc_httpc_recv_response(con_handle_t *handle, httpc_data_t *data);

Input Parameters :

handle
Type: con_handle_t*
Purpose: Connection handle"

data
Type: httpc_data_t *
Purpose: A pointer to the #httpc_data_t to manage the data sending and receiving

Returns :

HTTPC_RET_OK, if the operation was successful.
Negative error value, all other errors.

12.2.7 nrc_httpc_close

Close connection. Connecionit is included in each request method function.

Prototype :

void nrc_httpc_close(con_handle_t *handle)

Input Parameters :

handle
Type: bool
Purpose: Enable / disable

Returns :

N/A

© Copyright Newracom 2022. All rights reserved.
Confidential

13 FOTA

The FOTA API provides functions to:
- Check the support of FOTA and set FOTA information
- Erase and write FOTA area.
- Firmware and bootloader FOTA update done function.
- CRC32 calculation.

13.1 Data Type

These types are defined at the “lib/sdk/inc/api_fota.h”.

13.1.1 FOTA Information

FOTA_INFO is an information about FOTA firmware.

Table 13.1 FOTA_INFO

Name	Description
fw_length	Firmware length
crc	CRC32 value
ready	ready flag (Not used)

13.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_fota.h”.

13.2.1 nrc_fota_is_support

Check the flash is able to support FOTA

Prototype :

bool nrc_fota_is_support(void)

Input Parameters :

N/A

Returns :

True, if it supports FOTA.

False, if it does not support FOTA.

© Copyright Newracom 2022. All rights reserved.
Confidential

13.2.2 nrc_fota_write

Write len size from src to dst in fota memory area.

Prototype :

nrc_err_t nrc_fota_write(uint32_t dst, uint8_t *src, uint32_t len)

Input Parameters :

dst
- Type: uint32_t
- Purpose: offset from fota_memory start address

src
- Type: uint8_t*
- Purpose: source address

len
- Type: uint32_t
- Purpose: source data length

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

13.2.3 nrc_fota_erase

Erase len size from (fota start address + dst).

Prototype :

nrc_err_t nrc_fota_erase(uint32_t dst, uint32_t len)

Input Parameters :

dst
- Type: uint32_t
- Purpose: offset from fota_memory start address

len
- Type: uint32_t
- Purpose: source data length

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

13.2.4 nrc_fota_set_info

Set fota binary information (binary length and crc)

Prototype :

© Copyright Newracom 2022. All rights reserved.
Confidential

nrc_err_t nrc_fota_set_info(uint32_t len, uint32_t crc)

Input Parameters :

len
- Type: uint32_t
- Purpose: binary size

crc
- Type: uint32_t
- Purpose: crc value for binary

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

13.2.5 nrc_fota_update_done

Updated firmware and reboot.

Prototype :

nrc_err_t nrc_fota_update_done(FOTA_INFO* fw_info)

Input Parameters :

fw_info
- Type: FOTA_INFO*
- Purpose: fota binary information (binary length and crc)

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

13.2.6 nrc_fota_update_done_bootloader

Updated bootloader and reboot.

Prototype :

nrc_err_t nrc_fota_update_done_bootloader(FOTA_INFO* fw_info)

Input Parameters :

fw_info
- Type: FOTA_INFO*
- Purpose: fota binary information (binary length and crc)

Returns :

NRC_SUCCESS, if the operation was successful.
NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

13.2.7 nrc_fota_cal_crc

Calculate crc32 value.

Prototype :

nrc_err_t nrc_fota_cal_crc(uint8_t* data, uint32_t len, uint32_t *crc)

Input Parameters :

data

Type: uint8_t*

Purpose: A pointer for data

len

Type: uint32_t

Purpose: length for CRC

crc

Type: uint32_t

Purpose: A pointer to store the calculated crc value

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

N/A

© Copyright Newracom 2022. All rights reserved.
Confidential

14 Power save

The power save memory API provides functions to:
• Set power save mode
• Set wakeup pin and source

14.1 Data Type

These types are defined at the “lib/sdk/inc/api_ps.h”.

14.1.1 Power Save Sleep Mode

POWER_SAVE_SLEEP_MODE is a power save sleep mode.

Table 14.1 POWER_SAVE_SLEEP_MODE

Name	Description
POWER_SAVE_MODEM_SLEEP_MODE	Modem sleep
POWER_SAVE_DEEP_SLEEP_MODE	Deep sleep

14.1.2 Power Save TIM Mode

POWER_SAVE_TIM_MODE is a power save TIM mode.

Table 14.2 POWER_SAVE_TIM_MODE

Name	Description
POWER_SAVE_NON_TIM	Non TIM mode
POWER_SAVE_TIM	TIM mode

14.1.3 Power Save Wakeup Source

These 83efine are related to wakeup source.

Table 14.3 power save wakeup source

Define	Value
WAKEUP_SOURCE_NO_SLEEP	0x0
WAKEUP_SOURCE_RTC	0x00000001L << 0
WAKEUP_SOURCE_GPIO	0x00000001L << 1

© Copyright Newracom 2022. All rights reserved.
Confidential

14.2 Function Call

The header file for system APIs are defined at the “sdk/inc/api_ps.h”.

14.2.1 nrc_ps_set_sleep

Set the power save mode & power save protocol.

Prototype :

nrc_err_t nrc_ps_set_sleep(uint8_t sleep_mode, uint64_t interval, uint32_t timeout)

Input Parameters :

sleep_mode

Type: POWER_SAVE_SLEEP_MODE

Purpose: power save sleep mode. See “Power Save Sleep Mode”

interval

Type: uint64_t

Purpose: 0 (TIM mode) or non-zero (Non-TIM mode, interval >= 1000ms). The unit is ms.

Timeout

Type: uint32_t

Purpose: timeout. Only works in TIM mode with modem sleep. The unit is ms.

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

14.2.2 nrc_ps_deep_sleep

Command the device to go to deep sleep.

Prototype :

nrc_err_t nrc_ps_deep_sleep(uint64_t interval)

Input Parameters :

interval

Type: uint64_t

Purpose: The duration (interval >= 1000ms) for sleep. The unit is ms.

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

14.2.3 nrc_ps_modem_sleep

Command the device WIFI to go to sleep.

© Copyright Newracom 2022. All rights reserved.
Confidential

Prototype :

nrc_err_t nrc_ps_modem_sleep(uint64_t interval, uint32_t timeout)

Input Parameters :

interval

Type: uint64_t

Purpose: The duration (interval >= 1000ms) for WiFi sleep. The unit is ms.

Timeout

Type: uint32_t

Purpose: timeout. The unit is ms.

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

14.2.4 nrc_ps_tim_sleep

The function commands device WiFi to sleep.
The WiFi wakes up if Traffic Indication Map signal received.

Prototype :

nrc_err_t nrc_ps_tim_sleep(uint32_t timeout)

Input Parameters :

timeout

Type: uint32_t

Purpose: timeout. The unit is ms.

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

14.2.5 nrc_ps_set_modemsleep_stop

Stop the modem sleep

Prototype :

nrc_err_t nrc_ps_set_modemsleep_stop(void)

Input Parameters :

s

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

14.2.6 nrc_ps_set_gpio_wakeup_pin

Configure a wakeup-gpio-pin when system state is uCode or deepsleep.

※ This function should be called before deepsleep, if user want to config the wakeup-gpio-pin.

Prototype :

nrc_err_t nrc_ps_set_gpio_wakeup_pin(bool check_debounce, int pin_number)

Input Parameters :

check_debounce

Type: bool

Purpose: check mechanical vibration of a switch

pin_number

Type: int

Purpose: GPIO pin number for wakeup when GPIO is enabled for wakeup source

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

14.2.7 nrc_ps_set_gpio_wakeup_source

Configure wakeup sources when system state is deepsleep.

※ This function should be called before deepsleep, if user want to config the wakeup source.

Prototype :

nrc_err_t nrc_ps_set_wakeup_source(uint8_t wakeup_source)

Input Parameters :

wakeup_source

Type: uint8_t

Purpose: wakeup source. See "Power Save Wakeup Source"

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

14.2.8 nrc_ps_wakeup_reason

Get the wakeup reason.

Prototype :

nrc_err_t nrc_ps_wakeup_reason(uint8_t *reason)

Input Parameters :

© Copyright Newracom 2022. All rights reserved.
Confidential

reason

Type: uint8_t*

Purpose: A pointer to get wakeup reason. See "Power Save Wakeup Source"

Returns :

NRC_SUCCESS, if the operation was successful.

NRC_FAIL, all other errors.

© Copyright Newracom 2022. All rights reserved.
Confidential

15 PBC (Push Button)

WPS-PBC for simple network configuration

15.1 Data Type

These types are defined at the “sdk/inc/api_pbc.h”.

15.1.1 pbc_ops

pbc_ops are a structure type.

Table 15.1 pbc_ops

Name	Description
GPIO_PushButton	wps pbc GPIO for push button
nrc_wifi_wps_pbc_fail	wps pbc operation fail
nrc_wifi_wps_pbc_timeout	wps pbc operation timout
nrc_wifi_wps_pbc_success	wps pbc operation success
nrc_wifi_wps_pbc_pressed	wps pbc operation press

15.2 Function Call

The header file for PBC APIs is defined at the “sdk/inc/api_pbc.h”.

15.2.1 wps_pbc_fail_cb

This callback is called when wps pbc operation fail

Prototype :

void wps_pbc_fail_cb(void)

Input Parameters :

N/A

Returns :

N/A

© Copyright Newracom 2022. All rights reserved.
Confidential

15.2.2 wps_pbc_timeout_cb

This callback is called when there is no connection attempt for 120 second and timeout occurs.

Prototype :

void wps_pbc_timeout_cb(void)

Input Parameters :

N/A

Returns :

N/A

15.2.3 wps_pbc_success_cb

This callback is called when wps pbc operation succeeds

Prototype :

void wps_pbc_success_cb(void)

Input Parameters :

N/A

Returns :

N/A

15.2.4 wps_pbc_button_pressed_event

This callback is called when user push the button which is connected with GPIO we register for interrupt.
*

Prototype :

void wps_pbc_button_pressed_event(int vector)

Input Parameters :

vector

Type: int

Purpose: GPIO pin number for wakeup when GPIO is enabled for wakeup source

Returns :

N/A

© Copyright Newracom 2022. All rights reserved.
Confidential

15.2.5 wps_pbc_set_fail_cb

This callback is called when wps pbc set is failed

Prototype :

void wps_pbc_set_fail_cb(func_void cb)

Input Parameters :

cb

Type: func_void

Purpose: callback to register

Returns :

N/A

15.2.6 wps_pbc_set_timeout_cb

This callback is called when wps pbc timeout is set

Prototype :

void wps_pbc_set_timeout_cb(func_void cb)

Input Parameters :

cb

Type: func_void

Purpose: callback to register

Returns :

N/A

15.2.7 wps_pbc_set_success_cb

This callback is called when wps pbc set is succeed

Prototype :

void wps_pbc_set_timeout_cb(func_void cb)

Input Parameters :

cb

Type: func_void

Purpose: callback to register

Returns :

N/A

© Copyright Newracom 2022. All rights reserved.
Confidential

15.2.8 wps_pbc_set_btn_pressed_cb

This callback is called when wps pbc button is pressed

Prototype :

void wps_pbc_set_btn_pressed_cb(func_int cb)

Input Parameters :

cb

Type: func_int

Purpose: callback to register

Returns :

N/A

15.2.9 init_wps_pbc

Initialize wps pbc function

Prototype :

void init_wps_pbc(struct pbc_ops *ops)

Input Parameters :

ops

Type: struct pbc_ops *

Purpose: structure contains GPIO and callbacks

Returns :

N/A

© Copyright Newracom 2022. All rights reserved.
Confidential

16 Middleware API Reference

16.1 FreeRTOS

FreeRTOS is a market-leading real–time operating system (RTOS) for microcontrollers and small microprocessors.

- Official Website:
  - https://www.freertos.org/RTOS.html
- Online Documentation:
  - https://www.freertos.org/features.html
- Git Repository:
  - https://github.com/FreeRTOS/FreeRTOS

16.2 WPA_supplicant

Wpa_supplicant is a WPA Supplicant for Linux, BSD, Mac OS X, and Windows with support for WPA and WPA2 (IEEE 802.11i / RSN). Supplicant is the IEEE 802.1X/WPA component that is used in the client stations. It implements key negotiation with a WPA authenticator, and it controls the roaming and IEEE 802.11 authentication/association of the wlan driver.

- Official website:
  - https://w1.fi/wpa_supplicant/
- Online Documentation:
  - https://w1.fi/wpa_supplicant/devel/
- GitHub Page:
  - git clone git://w1.fi/srv/git/hostap.git

16.3 lwIP

lwIP (lightweight IP) is a widely used open-source TCP/IP stack designed for embedded systems.

- Official Website:
  - http://savannah.nongnu.org/projects/lwip
- Online Documentation:
  - http://www.nongnu.org/lwip
- Git Repository:
  - https://git.savannah.nongnu.org/git/lwip.git

© Copyright Newracom 2022. All rights reserved.
Confidential

16.4 MbedTLS

MbedTLS is an implementation of the TLS and SSL protocols and the respective cryptographic algorithms and support code required.

- Official Website:
  - https://tls.mbed.org
- Online API Reference:
  - https://tls.mbed.org/api
- GitHub Page:
  - https://github.com/ARMmbed/mbedtls

16.5 NVS library

NVS library used for storing data values in the flash memory. Data are stored in a non-volatile manner, so it is remaining in the memory after power-out or reboot. This lib is inspired and based on TridentTD_ESP32NVS work.

The NVS stored data in the form of key-value. Keys are ASCII strings, up to 15 characters. Values can have one of the following types:

- integer types: uint8_t, int8_t, uint16_t, int16_t, uint32_t, int32_t, uint64_t, int64_t
- zero-terminated string
- variable length binary data (blob)

Refer to the NVS ESP32 lib original documentation for a details about internal NVS lib organization.

© Copyright Newracom 2022. All rights reserved.
Confidential

17 S1G Channel

17.1 Overview

The wpa_supplicant is required for the NRC7292 STA operation. As the wpa_supplicant does not natively support S1G (Sub-1 GHz) channels, an internal channel mapping table is used within the firmware to uniquely associate each S1G channel to its pre-determined channel in the traditional 2.4/5GHz band.

As the set of available S1G channels is region-specific, a separate wpa_supplicant configuration file is used to handle the usage for each supported region.

17.2 Channel

17.2.1 US (United States) Channel

Table 17.1 US Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
1	1	902.5	2412
3	1	903.5	2422
5	1	904.5	2432
7	1	905.5	2442
9	1	906.5	2452
11	1	907.5	2462
36	1	908.5	5180
37	1	909.5	5185
38	1	910.5	5190
39	1	911.5	5195
40	1	912.5	5200
41	1	913.5	5205
42	1	914.5	5210
43	1	915.5	5215
44	1	916.5	5220
45	1	917.5	5225
46	1	918.5	5230
47	1	919.5	5235
48	1	920.5	5240
149	1	921.5	5745
150	1	922.5	5750

© Copyright Newracom 2022. All rights reserved.
Confidential

151	1	923.5	5755
152	1	924.5	5760
100	1	925.5	5500
104	1	926.5	5520
108	1	927.5	5540
2	2	903	2417
6	2	905	2437
10	2	907	2457
153	2	909	5765
154	2	911	5770
155	2	913	5775
156	2	915	5780
157	2	917	5785
158	2	919	5790
159	2	921	5795
160	2	923	5800
161	2	925	5805
112	2	927	5560
8	4	906	2447
162	4	910	5810
163	4	914	5815
164	4	918	5820
165	4	922	5825
116	4	926	5580

© Copyright Newracom 2022. All rights reserved.
Confidential

17.2.2 KR (Korea) Channel

There are two kinds of packages for KR. The developer could see the 'KR MIC (922.5Mhz-929.0Mhz)' log for KR (Korea) Channel (925.5Mhz – 929.0Mhz).

17.2.2.1 KR USN Package (921.5Mhz – 922.5Mhz)

Table 17.2 KR USN Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
44	1	921.5	5220
45	1	922.5	5225

17.2.2.2 KR MIC Package (925.5Mhz – 929.0Mhz)

Table 17.3 KR MIC Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
37	1	926.5	5185
38	1	927.5	5190
39	1	928.5	5195
42	2	927.0	5210
43	2	929.0	5215

17.2.3 JP (Japan) channel

Table 17.4 JP Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
36	1	917.0	5180
37	1	918.0	5185
38	1	919.0	5190
39	1	920.0	5195
40	1	921.0	5200
41	1	922.0	5205
42	1	923.0	5210
43	1	924.0	5215
44	1	925.0	5220
45	1	926.0	5725
46	2	927.0	5230

© Copyright Newracom 2022. All rights reserved.
Confidential

17.2.4 TW (Taiwan) channel

Table 17.5 TW Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
36	1	839.0	5180
37	1	840.0	5185
38	1	841.0	5190
39	1	842.0	5195
40	1	843.0	5200
41	1	844.0	5205
42	1	845.0	5210
43	1	846.0	5215
44	1	847.0	5220
45	1	848.0	5225
46	1	849.0	5230
47	1	850.0	5235
48	1	851.0	5240
149	2	839.5	5745
150	2	841.5	5750
151	2	843.5	5755
152	2	845.5	5760
153	2	847.5	5765
154	2	849.5	5770
155	4	840.5	5775
156	4	844.5	5780
157	4	848.5	5785

17.2.5 EU (Europe) channel

Table 17.6 EU Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
36	1	863.5	5180
37	1	864.5	5185
38	1	865.5	5190
39	1	866.5	5195
40	1	867.5	5200
41	2	864.0	5205
42	2	866.0	5210

© Copyright Newracom 2022. All rights reserved.
Confidential

17.2.6 CN (China) channel

Table 17.7 CN Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
36	1	755.5	5180
37	1	756.5	5185
38	1	757.0	5190
39	1	758.5	5195
40	1	759.5	5200
41	1	760.5	5205
42	1	761.5	5210
43	1	762.5	5215
44	1	763.5	5220
45	1	764.5	5225
46	1	765.5	5230
47	1	766.5	5235
48	1	767.5	5240
149	1	768.5	5745
150	1	769.5	5750
151	1	770.5	5755
152	1	779.5	5760
153	1	780.5	5765
154	1	781.5	5770

17.2.7 NZ (New Zealand) channel

Table 17.8 NZ Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
36 (Default)	1	915.5	5180
37	1	916.5	5185
38	1	917.5	5190
39	1	918.5	5195
40	1	919.5	5200
41	1	920.5	5205
42	1	921.5	5210
43	1	922.5	5215
44	1	923.5	5220
45	1	924.5	5225
46	1	925.5	5230

© Copyright Newracom 2022. All rights reserved.
Confidential

47	1	926.5	5235
48	1	927.5	5240
153	2	916.0	5765
154	2	918.0	5770
155	2	920.0	5775
156	2	922.0	5780
157	2	925.0	5785
158	2	927.0	5790
162	4	917.0	5810
163	4	921.0	5815
164	4	926.0	5820

17.2.8 AU (Australia) channel

Table 17.9 AU Channel

Available frequency band index	Bandwidth (MHz)	Sub-1GHz frequency	2.4 / 5G frequency
36 (Default)	1	915.5	5180
37	1	916.5	5185
38	1	917.5	5190
39	1	918.5	5195
40	1	919.5	5200
41	1	920.5	5205
42	1	921.5	5210
43	1	922.5	5215
44	1	923.5	5220
45	1	924.5	5225
46	1	925.5	5230
47	1	926.5	5235
48	1	927.5	5240
153	2	916.0	5765
154	2	918.0	5770
155	2	921.0	5775
156	2	923.0	5780
157	2	925.0	5785
158	2	927.0	5790
162	4	917.0	5810
163	4	922.0	5815
164	4	926.0	5820

© Copyright Newracom 2022. All rights reserved.
Confidential

18 Abbreviations

Table 18.1 Abbreviations and acronyms

Name	Description
IP	Internet Protocol
LwIP	Lightweight Internet Protocol
SDK	Software Development Kit
SDK	Software Development Kit
API	Application Programming Interface
EVB	Evaluation Board
AP	Access Point
STA	Station
SSID	Service Set Identifier
BSSID	Basic Service Set Identifier
RSSI	Received Signal Strength Indication
SNR	Signal-to-noise ratio
WPA2	Wi-Fi Protected Access 2
WPA3-SAE	Wi-Fi Protected Access 3 – Simultaneous Authentication of Equals
WPA3-OWE	Wi-Fi Protected Access 3 – Opportunistic Wireless Encryption
EAP	Extensible Authentication Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
AID	Association ID
MAC	Medium Access Control
dBm	decibel-Milliwatts
S1G	Sub 1 Ghz
HAL	Hardware Abstract Layer
ADC	Analog-to-Digital Converter
UART	Universal Asynchronous Receiver-Transmitter
PWM	Pulse-Width Modulation
SPI	Serial Peripheral Interface
TPC	Transmission Power Control
GPIO	General-purpose input/output
CPOL	Clock Polarity
CPHA	Clock Phase
TIM	Traffic Indication Map
NVS	Non-Volatile Storage

© Copyright Newracom 2022. All rights reserved.
Confidential

19 Revision history

Revision No	Date	Comments
Ver 1.0	11/01/2018	Initial version for customer release created
Ver 1.1	03/25/2019	APIs for Wi-Fi, Timer, ADC, and SPI updated
Ver 1.2	04/05/2019	APIs for Wi-Fi, Connection, Timer, I2C, ADC, SPI, PWM updated
Ver 1.3	07/02/2019	APIs for Wi-Fi, Connection updated
Ver 1.4	11/07/2019	Add HTTP Client API and FOTA API
Ver 1.5	07/05/2020	Add Serial Flash API
Ver 1.6	07/23/2020	Add Power save
Ver 1.7	08/20/2020	Add Wi-Fi BSSID setting. Change name ‘nrc_wifi_add_network’ and added ‘nrc_wifi_remove_network’
Ver 1.8	11/20/2020	Add APIs for WIFI (GET: aid, bssid, country, channel, bw, security, device_mode, info, disassoc)
Ver 1.9	12/30/2020	Add APIs for System (bdf, cal, log_level)
Ver 2.0	01/14/2021	Update api_timer and api_i2c. Added wpa_supplicant and AWS IoT SDK in middleware api
Ver 2.1	03/17/2021	Add API for WIFI; nrc_wifi_abort_scan
Ver 3.0	07/26/2021	Update SDK return values
Ver 3.1	09/21/2021	Update pwm api
Ver 3.2	09/30/2021	Remove sflash api, Added NZ, AU channel table and NVS library
Ver 3.3	10/15/2021	Add APIs for WPS PBC
Ver 3.4	02/25/2022	Update power save and system api
Ver 3.5	03/01/2022	Add set tx_time api
Ver 3.6	03/01/2022	Update nrc_uart_console_enable() and remove nrc_uart_printf()
Ver 3.7	03/16/2022	Added wifi apis - nrc_wifi_set_bss_max_idle(), nrc_wifi_set_mcs(), nrc_wifi_enable_duty_cycle(), nrc_wifi_disable_duty_cycle(), nrc_wifi_set_cca_threshold() Remove nrc_i2c_waitack()

© Copyright Newracom 2022. All rights reserved.
Confidential