rftest Documentation

espressif

2019 年 01 月 28 日

Contents

1 Quick Start Guide

1.1 Project Introduction

RFAPI:

- In eagletest/py_script/hal
- it is to store software interface APIs.

RFLIB:

- In eagletest/py_script/rftest/rflib
- it is to store common RF test functions.

TestCase:

- In eagletest/py_script/rftest/testcase
- it is to store RF test cases.

RFDATA:

- In eagletest/py_script/rftest/rfdata
- it is to store RF test data.

DOCS:

- In eagletest/py_script/rftest/docs
- it is to store document and testreport.

1.2 Start rftest

In “eagletest/py_script/” run commands:

```
ipython -i autotest.py rftest
chip=RFTCS(com(6)) # 6 should change to your com port number
```

1.3 Common test commands

1. write/read memory

   ```
   chip.mem.wrm(0x600060a0, 11, 8, 0x0)
   chip.mem.rdm(0x600060a0, 11, 8)
   ```

2. write/read i2c

   ```
   chip.i2c.rfrx.dtest = 0
   chip.i2c.rfrx.dtest
   ```

3. write/read PBUS

   ```
   chip.pbus.pbus_debugmode()
   chip.pbus.pbus_wr(‘rfrx1’, ‘en1’, 0x184)
   chip.pbus.pbus_rd(‘rfrx1’, ‘en1’)
   ```

chip.pbus.pbus_workmode()

4. select RF channel

   chip.wifi.rfchsel(1) # 1~14

5. TX Tone

   # Debug mode TX tone

   chip.pbus.pbus_debugmode()

   chip.pbus.open_tx(pa=0x5f, bb=0x120) # set PA and BB gain

   chip.rfcal.tos() # calibrate TX DC

   chip.wifi.txtone(1, 2, 40) #enable, frequency(mHz), digital attenuation(0.24db)

   # Debug mode stop TX tone

   chip.wifi.stoptone() #stop TX tone

   chip.pbus.pbus_workmode() # exit PBUS debug mode

   # Work mode TX tone

   chip.wifi.force_txon(1)

   chip.wifi.txtone(1, 2, 40) #enable, frequency(mHz), digital attenuation(0.24db)

   # Work mode stop TX tone

   chip.wifi.stoptone()

   chip.wifi.force_txon(0)

6. RX dump test

   chip.dump.adcdumptest(dump_num=1024, plot_en=1)

1.4 Basic functions test cases

1. Write and read test

   basic=BasicTest(chip.comport, chip.chipv)

   basic.mem_test()

   basic.i2c_test()

   basic.pbus_test()

2. RFPLL CAP Sweep

   basic=BasicTest(chip.comport, chip.chipv)

   basic.rfpll_sweep()

1.5 Performance oprimization cases

1.6 TX/RX performance test

1. open instrument server

   instru_server('iqx') #input: 'iqv', 'iqx', 'wt'

2. TX Power & EVM & MASK Test

   test=WIFI_TXRX_TEST(chip.comport, chip.chipv)

   test.WIFI_TX_PWR_EVM(cable_lose=2, channel=[14], data_rate=['mcs7'], iqv_no=2, iqv_num=10)

3. RX Sensitivity Test

   test.WIFI_RX_sens(cable_lose=2, chan_in=[14], data_rate=['mcs7'],iqv_no=1)

4. RX Maximum input level Test

   test.WIFI_RX_maxlevel(cable_lose=2, chan_in=[14], data_rate=['mcs7'], iqv_no=1)

5. RX Dynamic Range Test

   test.WIFI_RX_range(cable_lose=2, chan_in=[14], data_rate=['mcs7'], rx_range=['[-75, 0]'], iqv_no=1)

1.7 Current test

1.8 Generate docomments

In "eagletest/py_script/rftest/docs" run commands:

ipython doc_gen.py rftest

2 RFAPI

2.1 common

class common.**CHIP_ID**(*channel*, *chipv='AUTO'*)

基类: object

docstring for CHIP_ID

chip_mac()

class common.**CHIP_INFO**(*channel*, *INchipv='AUTO'*)

基类: object

docstring for CHIP_INFO

get_chipv()

Use UART Date register to determine the chip version

class common.**I2C**(*channel*, *chipv='ESP32'*)

基类: object

docstring for I2C

i2c_rd(*block*, *host_id*, *reg_addr*)

i2c_rdm(*block*, *host_id*, *reg_addr*, *msb*, *lsb*)

i2c_wr(*block*, *host_id*, *reg_addr*, *value*)

i2c_wrm(*block*, *host_id*, *reg_addr*, *msb*, *lsb*, *value*)

class common.**MEM**(*channel*, *chipv='ESP32'*)
基类: object

docstring for common

accumiq(*mem_addr*, *burst_len*)

clrmask(*reg_addr*, *mask*)

rd(*reg_addr*)

rdm(*reg_addr*, *msb*, *lsb*)

rdmem(*mem_addr*, *data_len*)

setmask(*reg_addr*, *mask*)

wr(*reg_addr*, *value*)

wrm(*reg_addr*, *msb*, *lsb*, *value*)

class common.**PBUS**(*channel*, *chipv='ESP8266'*)
基类: object

docstring for PBUS

pbus_debugmode()

pbus_rd(*pbus_sel*, *pbus_en_sel*)

pbus_rm(*pbus_sel*, *pbus_en_sel*, *msb*, *lsb*)

pbus_wm(*pbus_sel*, *pbus_en_sel*, *msb*, *lsb*, *value*)

pbus_workmode()

pbus_wr(*pbus_sel*, *pbus_en_sel*, *value*)

2.2 wifi_api

class wifi_api.**WIFIAPI**(*channel*, *chipv='ESP32'*)
基类: object

adctrig(*smp_num_aft_trig*, *trig_mode='sw'*, *sample_80m=0*, *trigcase=0*, *dump_trig=0*, *rx_gain_mode=0*, *rx_gain=0*, *rx_gain0=0*, *gain0_wait=0*)
return curr_ptr,wrap_flag, buff_addr, buff_size

bbinit()

Brief bb init

Param

- no param

返回

- no return

cbw40m_en(*en=0*)

Brief 40M bandwidth enable

Param

- en:
- 0: HT40 disable

- 1: HT40 enable

返回

- print the status

cmdstop()

Brief wifi TX/RX state stop

Param

- no param

返回

- no return

esp_origin_mac()

Brief origin mac read

Param

- no param

返回

- the value of origin mac

esp_rx(*chan=1*, *data_rate=23*)

Brief rx packect command

Param

- chan: channel number (1 to 14)
- **data_rate:**

11b		11g		11n	
param	rate	param	rate	param	rate
0x0	1M	0xb	6M	0x10	MCS0
0x1	2ML	0xf	9M	0x11	MCS1
0x2	5.5ML	0xa	12M	0x12	MCS2
0x3	11ML	0xe	18M	0x13	MCS3
0x4	–	0x9	24M	0x14	MCS4
0x5	2MS	0xd	36M	0x15	MCS5
0x6	5.5MS	0x8	48M	0x16	MCS6
0x7	11MS	0xc	54M	0x17	MCS7

返回

- print the status

esp_tx(*chan=1*, *data_rate=23*, *backoff=0*)

Brief tx packect command

Param

- chan: channel number (1 to 14)

- **data_rate:**

11b		11g		11n	
param	rate	param	rate	param	rate
0x0	1M	0xb	6M	0x10	MCS0
0x1	2ML	0xf	9M	0x11	MCS1
0x2	5.5ML	0xa	12M	0x12	MCS2
0x3	11ML	0xe	18M	0x13	MCS3
0x4	–	0x9	24M	0x14	MCS4
0x5	2MS	0xd	36M	0x15	MCS5
0x6	5.5MS	0x8	48M	0x16	MCS6
0x7	11MS	0xc	54M	0x17	MCS7

- backoff: tx power attenuation, 4 indicates an attenuation 1dB

返回

- print the status

filltxpacket(*PackLen*, *pdu0len*, *pdu1len*, *rate=0*, *key_no=0*, *bssid_no=0*, *lnkstartaddr=0*, *gi_type='long'*, *ap_mac_5=1*, *ap_mac_4=2*, *ap_mac_3=3*, *ap_mac_2=4*, *ap_mac_1=5*, *ap_mac_0=6*)

Brief wifi TX tone set

Param

- PackLen: include que_no and packeet_len,as (que_no<<16)+packlen
- pdu0len: include pdu0 and pdu2 length, as (pdu2len<<16)+pdu0len
- pdu1len: include pdu1 and pdu3 length, as (pdu3len<<16)+pdu1len
- for ampsdu and ampdu, pdu0len and pdu1len must set to zero
- lnkstartaddr:for ver5.0 above, it <<8+bssid_no as param to board

get_rx_tone_pwr(*rx_freq_cfg*)

init_print()

Brief init print

Param

- no param

返回

- no return

macinit()

Brief mac init

Param

- no param

返回

- no return

phyinit()

Brief phy init(include rfinit and bbinit and macinit)

Param

- no param

返回

- no return

read_hw_noisefloor()

Brief RF noisefloor read

Param

- no param

返回

- the value of hardware noisefloor

rfchsel(*chan*, *cbw2040_cfg=0*)

Brief wifi channel set

Param

- chan: channel number (1 to 14)
- **cbw2040_cfg:**
  - 1:HT40 enable
  - 0:HT40 disable

返回

- no return

rfinit()

Brief rf init

Param

- no param

返回

- no return

rxdc_cal()

rxstart(*rate_sym*)

Brief wifi RX state open

Param

- rate_sym: wifi rate (need to measure RX performance)

返回

- no return

set_noise_floor(*noise=380*)

Brief RF noisefloor set

Param

- noise: value of noisefloor

返回

- no return

set_tx_dig_gain (*force_en=1*, *dig_gain=25*)

Brief set tx digital gain command

Param

- **force_en:**
  - 0: disable;
  - 1: enable
- dig_gain:

返回

- print the status

set_tx_gain (*pa_gain=95*, *bb_gain=288*)

Brief set tx gain command

Param

- pa_gain: 0x1f, 0x2f,0x3f,0x4f,0x5f,0x6f,0x7f
- bb_gain: . . . , 0x100,0x140,0x20,0x60, . . .

返回

- print the status

stoptone (*tone_no=0*)

Brief wifi tx tone state close

Param

- tone_no: the number of tone

返回

- no return

target_power_backoff (*backoff_qdb*)

Brief wifi target power backoff

Param

- backoff_qdb: value of backoff

返回

- no return

test_txtone_pwr (*atten*, *loop_num*, *mode=0*, *step=0*, *delay_us=10*)

tx_cbw40m_en (*en=0*)

Brief Tx CBW40 enable

Param

- en:
- 0: Tx CBW40 disable

- 1: Tx CBW40 enable

返回

- print the status

tx_contin_en (*en=0*)

Brief Tx continuous enable

Param

- en:
- 0: Tx continuous disable
- 1: Tx continuous enable

返回

- print the status

txstart (*tx_rate*, *packnum*, *que_no=1*, *frm_delay=100*, *cbw40=0*, *ht_dup=0*, *dis_cca=1*)

txtone (*tone1_en=1*, *freq1_mhz=2*, *tone1_att=0*, *tone2_en=0*, *freq2_mhz=0*, *tone2_att=0*)

Brief wifi TX tone set

Param

- **tone1_en:**
  - 1: first tone enable
  - 0: fisrt tone disable
- freq1_mhz: first tone offset frequency
- tone1_att: first tone attenuation set
- **tone2_en:**
  - 1: second tone enable
  - 0: second tone disable
- freq2_mhz: second tone offset frequency
- tone2_att: second tone attenuation set

返回

- no return

txtone_step (*step1=0*, *att1=0*, *en2=0*, *step2=0*, *att2=0*)

wifiscwout (*en=1*, *chan=1*, *backoff=0*)

Brief SCW TX command

Param

- **en: SCW Tx enable**
  - 0: disable
  - 1: enable
- chan: channel number (1 to 14)
- backoff: tx power attenuation, 4 indicates an attenuation 1dB

返回

- print the status

wifitxout(*chan=1*, *data_rate=23*, *backoff=0*)

Brief tx packect command

Param

- chan: channel number (1 to 14)
- **data_rate:**

11b		11g		11n	
param	rate	param	rate	param	rate
0x0	1M	0xb	6M	0x10	MCS0
0x1	2ML	0xf	9M	0x11	MCS1
0x2	5.5ML	0xa	12M	0x12	MCS2
0x3	11ML	0xe	18M	0x13	MCS3
0x4	–	0x9	24M	0x14	MCS4
0x5	2MS	0xd	36M	0x15	MCS5
0x6	5.5MS	0x8	48M	0x16	MCS6
0x7	11MS	0xc	54M	0x17	MCS7

- backoff: tx power attenuation, 4 indicates an attenuation 1dB

返回

- print the status

2.3 bt_api

class bt_api.**BTAPI**(*channel*, *chipv='ESP32'*)

基类: object

bt_tx_tone(*en=0*, *chan=1*, *backoff=0*)

Brief BT tx tone open

Param

- **en:**
  - 1: BT tx tone enable
  - 0: BT tx tone disable
- chan: BT tx channel set (0 to 78)
- backoff: tone power attenuation set,step is 1(0.25dbm)

返回

- no return

fcc_bt_tx(*pwr_level*, *FH_en*, *tx_chan*, *rate*, *DH*, *datatype*)

Brief BR/EDR tx open

Param

- pwr_level: TX power level,range 0 to 7,step 3dbm

- **FH_en:**
  - 1: frequency hopping enable
  - 0: frequency hopping disable
- tx_chan: BT tx channel set (0 to 78)
- rate: tx rate set,1=1Mbps,2=2Mbps,3=3Mbps
- DH: 1=DH1.3=DH3,5=DH5
- datatype: 0: 01010101, 1: 00001111, 2: prbs9

返回

- no return

fcc_le_tx(*pwr_level*, *tx_chan*, *payload_len*, *datatype*)

Brief LE tx open

Param

- pwr_level: TX power level,range 0 to 9,step 2dbm,deafult 4
- tx_chan: BT tx channel set (0 to 39)
- payload_len: payload length,range 0 to 255,deafult 250
- datatype: 0: 01010101, 1: 00001111, 2: prbs9

返回

- no return

fcc_le_tx_syncw(*pwr_level*, *tx_chan*, *payload_len*, *datatype*, *syncw*)

Brief le tx (add synchronization of DC offset compensation and identification)open

Param

- pwr_level: TX power level,range 0 to 9,step 2dbm,deafult 4
- tx_chan: BT tx channel set (0 to 39)
- payload_len: payload length,range 0 to 255,deafult 250
- datatype: 0: 01010101, 1: 00001111, 2: prbs9
- syncw: deafult syncw=0x71764129

返回

- no return

rw_le_rx_per(*rx_chan*, *syncw*)

Brief LE Rx open

Param

- **rx_chan: rx channel set (0 to 39)**
  - channel 0、1、2–10 is corresponding frequency 2404MHz、2406MHz、2408MHz–2424MHz
  - channel 11、12、13–36 is corresponding frequency 2428MHz、2430MHz、2432MHz–2478MHz

- channel 37、38、39 is corresponding frequency 2402MHz、2426MHz、2480MHz

- syncw: synchronization of DC offset compensation and identification，is decide for instrument

返回

- no return

rw_rx_per(*modetype*, *rx_chan*, *ulap*, *ltaddr*)

Brief BR/EDR Rx open

Param

- **modetype:**
  - 0: BR
  - 1: EDR
- rx_chan: rx channel set (0 to 78),even number channel is from 0 to 39,uneven number is from 40 to 78, for example: 1 is channel 2, 40 is channel 1
- ulap: BT MAC,size is 32bit,include UAP(8bit) + LAP(24bit),the param is decide for instrument
- ltaddr: logical transport address,is decide for instrument,range 0 to 7

返回

- no return

3 RFLIB

3.1 pbus

class pbus.**pbus**(*comport*, *chipv='ESP32'*)

基类: object

all_pbus()

open_rx()

open_tx(*pa=95*, *bb=288*)

pbus_debugmode()

pbus_rd(*pbus_sel*, *pbus_en_sel*)

pbus_workmode()

pbus_wr(*pbus_sel*, *pbus_en_sel*, *value*)

read_dco()

set_dco(*i1=256*, *q1=256*, *i2=256*, *q2=256*)

3.2 rfpll

class rfpll.**rfpll**(*comport*, *chipv='ESP32'*)

基类：object

read_rfpll_reg()

reset()

restart_cal()

set_freq(*frf*, *cry_freq=40*)

set_freq_outband()

3.3 wifi_lib

class wifi_lib.**WIFILIB**(*comport*, *chipv='ESP32'*)

基类：object

GetDesirePackNum(*result*)

GetEntryAddr(*result*)

GetFcsErr(*result*)

GetFreqoff(*result*)

GetFrmCount(*result*)

GetGain(*result*)

GetGoodData(*result*)

GetGoodPackNum(*result*)

GetKeyMatch(*result*)

GetNoise(*result*)

GetRssi(*result*)

GetRxHung(*result*)

GetRxHung_status(*result*)

GetRxState(*result*)

GetTotErr(*result*)

GetValidCount(*result*)

cbw40m_en(*en=0*)

chan2freq(*chan*)

cmdstop()

esp_rx(*chan*, *rate_sym*)

esp_tx(*chan=1*, *ratenum=23*, *backoff=0*)

force_tx_gain_init(*chan=1*, *rate='mcs7'*, *pa_gain=95*, *bb_gain=288*, *dig_gain=244*)

force_txon(*en=0*)

get_filename(*folder*, *file_name*, *sub_folder=''*)

Folder file store folder

File_name file name

Sub_folder if not need, it may be default “”

get_length_delay_duty(*rate=23*, *duty=0.1*)
set the duty cycle of tx packet,duty range(0,1]

get_length_delay_mean(*rate=23*)
set the diffrent duty cycle to meet average current at 140mA

get_ratelst(*ratelst*, *rx_rate_option=”*)

get_rx_cable_lost(*iqv_unit_no=1, cable_att=30, chan_m=[14], noise_ref=-95.2*)

get_rx_tone_pwr(*rx_freq_mhz=5*, *rx_freq_cmp_mult20=0*, *gain_force=0*, *gain=40*)

get_rx_tone_pwr_scan(*rx_freq_mhz=5*, *gain_force=0*, *gain=40*, *scan_range=10*)

i2c_ric(*block*, *ctrl_name*)

i2c_wic(*block*, *ctrl_name*, *value*)

rate2ht40(*rate*)

ratecheck(*rate_sym*)

read_mac()

rf_tune()

rfchsel(*chan*, *cbw2040_cfg=0*)

rxdc_cal()

rxresult(*result_data*)

rxstart(*rate_sym*)

save_init_print(*folder=”*)

stoptone()

test_txtone_pwr(*atten*, *loop_num*, *mode=0*, *step=0*, *delay_us=10*)

tx_cbw40m_en(*en=0*)

tx_contin_en(*en=0*)

txout(*rate_sym*, *PackNum=0*, *PackLen=1024*, *cbw40=0*, *ht_dup=0*, *backoff_qdb=0*, *frm_delay=2000*)

txpacket(*txchan=1*, *rate_sym='mcs7'*, *PackNum=0*, *cbw40=0*, *ht_dup=0*, *backoff_qdb=0*, *duty=0*)

txtone(*tone1_en=1*, *freq1_mhz=2*, *tone1_att=0*, *tone2_en=0*, *freq2_mhz=0*, *tone2_att=0*)

wifiscwout(*en=0*, *chan=1*, *backoff=0*)

3.4 adc_dump

class adc_dump.**DUMP**(*comport*, *chipv='ESP32'*)
基类: object

adcdump(*logdir*, *start_addr*, *byte_len*, *burst_len*, *buff_start*, *buff_size*, *trig_pos*, *adc_version='10bit'*, *chan_no=1*, *dump_13bit=0*)

adcdumptest (*logdir='dump'*, *dump_num=1024*, *trig_mode='sw'*, *adc_version='10bit'*, *sample_80m=0*, *plot_en=0*, *chan_en=0*, *chan=14*, *force_gain_en=0*, *force_gain=70*, *force_gain0=70*, *gain0_wait=0*, *rxgain_offset=40*, *trigcase=0*, *dump_trig=0*, *subplot_en=0*, *subplot_row=3*, *subplot_col=3*, *index=0*, *plot_save=0*, *dump_13bit=0*)

dump_trig: 1 is dump first, then trig, 0 is trig first, then dump.

get_dump_accm (*accum_i*, *accum_q*, *rssi*, *mem_addr*, *burst_len*)

get_dump_data (*dump_13bit*, *adc_data*)

sampledeal (*sample*, *adc_version='8bit'*)

set_dump_mode (*dump_13bit=0*)

write_dump_data (*accum_i*, *accum_q*, *accum_lna_sat*, *accum_vga_sat*, *rssi*, *mem_addr*, *burst_len*, *csvwrite*, *dump_13bit=0*)

4 TESTCASE

4.1 rf_debug

Store RF submodule test scripts

for example: PA, I2C, TX Gain, RX Gain test ....

4.2 performance

Store RF performance scripts

for example: wifiTX WIFIRX BTTX BTRX performance ...

4.3 current

Store TX/RX current measurement scripts

4.4 temprature

Store multiple chips testing scripts in temperature chamber

5 TestReport

Python 模块索引

索引

R

S

T

W